A RISC-V based hardware accelerator designed for Yolo object detection system

Guohe Zhang[1], Kepeng Zhao[1], Bin Wu[1], Yiqun Sun[2], Li Sun[3] and Feng Liang[1]

1. School of Microelectronics, Xi'an Jiaotong University. Xi'an, Shaanxi, 710049, China;
E-mail: zhangguohe@mail.xjtu.edu.cn (G. Z.); zkp549204742@stu.xjtu.edu.cn (K. Z.); newwubin19950404@stu.xjtu.edu.cn (B. W.); fengliang@xjtu.edu.cn (F. L.)
2. UC TECH IP Co., Ltd. Shenzhen, Guangdong, 518057, China;
E-mail: yiqun.sun@uctechip.com (Y. S.)
3. Airforce Engineering University. Xi'an, Shaanxi, 710077, China
E-mail: sl_lxa@mail.nwpu.edu.cn (L. S.)

Abstract

You only look once (YOLO) is a state-of-the-art, real-time object detection system. A hardware accelerator of YOLO is presented using the open source RISC-V core ROCKET as its controller. Extended customized instructions based on RISC-V were proposed for this accelerator. The hardware design was verified using Xilinx Virtex-7 FPGA VC709 and the results show that the accelerator costs about 400ms to finish the Yolo algorithm and is supposed to have higher speed with more computation modules.

Key words: accelerator, Yolo, RISC-V, object detection

I. Introduction

Nowadays, Convolutional Neural Network (CNN) has been widely researched for its good performance in a wide range of applications such as image classification, object detection and tracking. And this technique also shows its huge potential in the industry like surveillance camera, unmanned supermarket, auto driving which can really change our life. Since the AlexNet [1] in 2012, the demand of computation power is increased rapidly. As we all know, CNN consists of many matrix multiplications which are not effective enough to be operated on CPU. So, what is the suitable hardware to do this computation has also been researched.

Though the hardware accelerator can work by itself, [2] designs an accelerator on FPGA+Xeon platform. We design a system of CPU + accelerator for the idea of configuration. So, the accelerator can be controlled by introductions from CPU and the area and scale of the accelerator will be configured before used. As for the CPU, RISC-V [3] core is chosen for its opensource and extensibility, we design our own custom introductions for the accelerator used as a co-processor.

You only look once [4] (YOLO) is a state-of-the-art, real-time object detection system which is extremely fast and accurate. Our accelerator is designed for the inference part of this algorithm. Thinking about that this algorithm can be used in surveillance camera in which power consumption and chip area is firstly cared. So, the network structure of YOLO is researched carefully and the computation module is designed according to it. Generality has been abandoned to achieve the high energy-efficient. The idea of configuration has also been thought to allow users to choose how many computation modules they want to use.

To increase the performance of accelerator, [5] deceases the movement of data from on-chip and off-chip by a reconfigurable architecture. [6] adopts a recently proposed binary weight method to converts the CNN computation to multiplication-free processing, [7] designs an accelerator using depthwise separable convolution and ping-pong on-chip buffers are used to reduce the bandwidth limitation of off-chip memory.

In this paper, we design a multi-level memory hierarchy to make sure that the whole system can work in pipeline with the limited bandwidth. The hardware implementation of YOLOv2 algorithm is realized and verified by using Xilinx Virtex-7 FPGA VC709.

II. **Algorithm used and CPU platform based**

A. YOLO algorithm

YOLO is a state-of-the-art, real-time object detection system, the class of the objects will be shown and bounding boxes will be used to show where they are in the pictures. Object detection is a very practical application and YOLO has too big network to be implemented on embedded CPUs, and that's why we want to design an accelerator for it. Darknet-19 is used as the inference network in YOLO, in which we find that the size of filters is always 3*3 or 1*1 and the stride of convolution is always 1. The size of output is always the multiples of 7. So, the computation module is designed to adapt these sizes which causes the result that our accelerator is much smaller than others and can finish this algorithm in a small area of chip.

B. RISC-V

RISC-V is a RISC instruction set architecture which is proposed by University of California, Berkeley, and has been promoted for its good performance and opensource. The RISC-V foundation wants to start a new time of hardware designing when engineers can design a chip like a software using the opensource ISA and toolchains which will reduce the difficulty to develop a chip. The custom introductions have been held to let users to extend their own introductions for their designs. In RISC-V architecture, ROCC interface is set for co-processors, introductions for co-processors will be distinguished by the decoder of CPU and be passed through ROCC interface to the accelerator. So, our accelerator and CPU form a system, 8 introductions have been designed to control the accelerator

III. Architecture

A. Architecture of the accelerator

The architecture of this accelerator is shown as Fig. 1. As we mentioned in section II, this accelerator works based on a RISC-V core. The instructions and data given from CPU core through the ROCC interface are stored in the instruction FIFO and data FIFO. Finite-state machine (FSM) works as the controller, after configuration by the signals decoded from the instructions, FSM starts the accelerator. The input data is read from the DDR SDRAM off the chip and stored in the buffer. With the weight they are then fed into the computation module for convolution, pooling, activation function and the output result is stored in another buffer which works as the input buffer for the next layer. The data will be stored to DDR SDRAM when all the layers are finished and the FSM will stop until the next instruction.

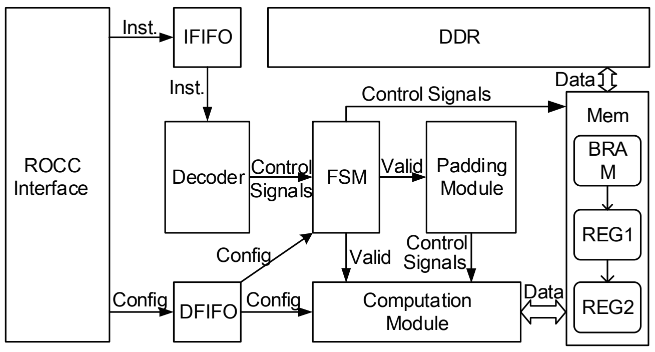

Fig. 1 Architecture of the whole accelerator

B. *Design of the computation module*

The computation module is designed for the computation part of the YOLO algorithm consisting of convolution, pooling and activation function.

1) Convolution

The convolution part owns 9 multipliers and 7 adders and a FIFO. The concrete architecture is shown in Fig. 2, in which the idea of pipeline is used to accelerate the computation. The first level is 9 multipliers, the input data from the shift registers (we will introduce them in the next part) and the weights from the padding module are fed into two input ports of the multiplier. Adders with saturation function and a FIFO compose the next levels. The weights are fixed while the input data is updated by the shift registers every clock cycle when a convolution computation is implemented. Weights will be changed after a row of input data computation is finished.

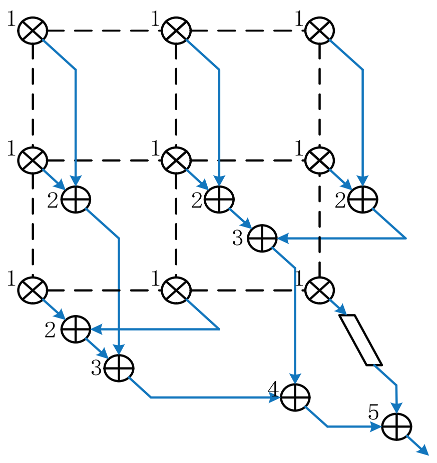

Fig. 2 The struction of the convolution part

2) Pooling

We only do max pooling because in YOLO algorithm there is only one average pooling, so it's better to do that in the RISC-V CPU core. As for the max pooling part we design, there are three comparators to get the max element among four input data. Whether the pooling calculation is needed is decided by the signal decoded from the instruction from the RISC-V core. When it's enabled, the valid signal to start this part will come when the accumulation of one raw data in an output channel is finished. And if the instruction tells this layer doesn't need pooling, the data from the accumulator will directly store into the buffer.

3) Activation

The activation function we implement here is RELU which is easy to realize in circuit, but for the future configurable design, we will give more functions which can be chose by the configuration information given from the instruction.

C. *Design of the memory hierarchy*

Memory hierarchy is a very important part of a system, it will indeed influence the area and performance of a circuit especially when it is implemented on a FPFA. The memory off the chip like DDR SDRAM can store a large amount of data but the bandwidth of the interface is limited which can be the bottleneck of the performance. So, a multi-level memory hierarchy is designed to solve this problem which is shown in Fig. 3.

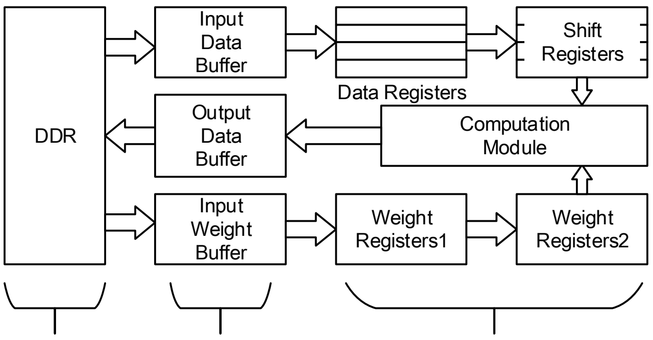

Fig. 3 Design of the multi-level memory hierarchy

1) DDR

The first level is the DDR off the chip to store the input data and weight which is seldom used because of its low bandwidth and high latency.

2) Local buffer

The second level is the local buffer which is the main memory for our accelerator. There are three buffers to store data and weight, two of which are data buffer to store the input data and the result after a layer computation is finished. Data buffer must be implemented big enough for the input image because in the YOLO algorithm the greatest demand of memory occurs in the input layer. The necessity to design two of them is that the role of one is the input buffer and the other is output buffer and they will be exchanged when a layer computation is finished. Compared with data buffer, weight buffer is much smaller because the amount of weight YOLO need is too large to store in chip. So, the content must be updated from DDR after half of it in weight buffer is used to ensure the computation will continue with no delay.

3) Double-registers group

The last level is a double-registers group for data and weight,

ISBN: 978-1-5386-8139-8

in which a 9*8bits register group is for weight and a 226*4*8bits shift register group is for data. The reason why we need double of it is that the first register group exists for data and weight which are being transported from the buffer while the next register group is the space for the content which is being computed.

D. Design of padding module

Controller is an important module of a chip, in an accelerator FSM used to be the controller while in our design we decide to implement a padding module in charge of transporting data and weight from buffer to double-registers group. This module reduces so much complexity of FSM that it will just give a valid signal to start the computation state and then just wait the stop signal from computation module to change to next state. As I mentioned before, the bandwidth of the memory has been the bottleneck of the performance, since the bandwidth and speed of data transporting cannot be improved, we do our best to ensure that the data flows into the computation module with no delay. That's why we design the double-registers group, the data for next computation is stored in the second level group, and once the computation is over, it will be fed into the computation module in one clock cycle. The data in first level register group will than flows into the second level register group.

E. Configurable design in the accelerator

Configurable design is an important part of this accelerator. The number of computation modules can be decided by users. The power consumption and chip area are the lowest when working with only one computation module, more of which can be implemented if users want higher performance.

IV. Results and discussion

Our work is implemented on the Xilinx Virtex-7 FPGA VC709, the resource costs and utilization when only one computation module is implemented are shown in TABLE I.

TABLE I RESOURCE COSTS AND UTILIZATION

	used	available	Resource Utilization (%)
LUT	13798	433200	3.19
FF	17514	866400	2.02
BRAM	83	1470	5.65
DSP	161	3600	4.47

TABLE II COMPARSION WITH OTHER WORKS

	Platform	Power (W)	Performance (GOP/s)	Energy Efficiency (GOPS/W)
Ref.[8]	Virtex-7 VXT458t	18.61	61.62	3.31
Ref.[9]	Zynq-7000	3.32	2.13	0.64
This work (1CM)	VC709	2.38	3.5	1.47
This work (7CMs)	VC709	3.85	24.5	6.36

Two versions of our work (one computation module and seven computation modules) are compared with other accelerators, the result of performance and power consumption is shown in TABLE II. About 2.7s is cost to finish the YOLO algorithm in our accelerator with only one computation module and 400ms will be cost when seven computation modules are implemented.

V. Conclusion

A YOLO accelerator is presented with custom designed RISC-V introductions extended based on a ROCC interface of an opensource core specially. A multi-level memory architecture and a configurable computation module are presented to reduce the power consumption and hardware resources. The results according to the verification of Xilinx Virtex-7 FPGA VC709 show that the accelerator costs about 400ms to finish the Yolo algorithm and is supposed to have higher speed with more computation modules.

Acknowledgments

This research was partly supported by the National Natural Science Foundation of China under Grant No.61701531 and No. 61474093 and the Core Electronic Devices, High-end General Chips and Basic Software Products Projects of China (2017ZX01030204).

References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. (2014). ImageNet classification with deep convolutional neural. In *Neural Information Processing Systems* (pp. 1-9).

[2] Moss, D. J., Nurvitadhi, E., Sim, J., Mishra, A., Marr, D., Subhaschandra, S., & Leong, P. H. (2017, September). High performance binary neural networks on the Xeon+ FPGA™ platform. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 1-4). IEEE.

[3] Lee, Y., Waterman, A., Avizienis, R., Cook, H., Sun, C., Stojanović, V., & Asanović, K. (2014, September). A 45nm 1.3 GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators. In *ESSCIRC 2014-40th European Solid State Circuits Conference (ESSCIRC)* (pp. 199-202). IEEE.

[4] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

[5] Chen, Y. H., Krishna, T., Emer, J. S., & Sze, V. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, *52*(1), 127-138.

[6] Duan, Y., Li, S., Zhang, R., Wang, Q., Chen, J., & Sobelman, G. E. (2018, November). Energy-Efficient Architecture for FPGA-based Deep Convolutional Neural Networks with Binary Weights. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)* (pp. 1-5). IEEE.

[7] 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*, *65*(10), 1415-1419.

[8] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015, February). Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 161-170). ACM.

[9] Feng, G., Hu, Z., Chen, S., & Wu, F. (2016, October). Energy-efficient and high-throughput FPGA-based accelerator for Convolutional Neural Networks. In *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)* (pp. 624-626). IEEE.

ISBN: 978-1-5386-8139-8