High-Speed FPGA-to-FPGA Interface for a Multi-Chip CNN Accelerator

Gitae Park, Thaising Taing, Hyungwon Kim

Dept. of Electronics Engineering, Chungbuk National University, Cheongju, Korea

{pgt1756, thaisingtaing, hwkim}@chungbuk.ac.kr

2023 20th International SoC Design Conference (ISOCC) | 979-8-3503-2703-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISOCC59558.2023.10396076

Abstract— **The size of Convolutional Neural Network (CNN) hardware accelerators is increasing year by year due to the rapidly growing complexity of CNN models. Due to its intensive growth, the prototyping and deployment of CNN accelerators on FPGA devices are becoming challenging. To solve this problem, several researchers have proposed multi-chip CNN accelerator architectures. These architectures utilize a multi-FPGA testing platform that employs high-speed FPGA-to-FPGA communication interfaces. To deal with communication issues in multi-FPGA CNN accelerators, we present an implementation of a high-speed FPGA-to-FPGA interface based on the Aurora protocol. Aurora is a scalable, lightweight, link-layer protocol for high-speed serial communication. To demonstrate the advantages of the FPGA-to-FPGA interface, we designed a multi-chip CNN accelerator using the Aurora interface, which is implemented on Xilinx ZCU102 and Xilinx VU19P FPGAs.**

Keywords; CNN Accelerator, Aurora Interface, FPGA, PCIe, MicroBlaze

I. INTRODUCTION

FPGAs are widely used as a means of validating SoC designs for deep neural networks (DNNs) accelerators. DNN is utilized for a variety of image recognition such as image classification, object detection, and segmentation aimed at autonomous vehicles and factory automation.

Among DNNs, CNN recognizes feature patterns by learning the geometry or textures of training image data based on convolutional filter operations. Convolution layers, each of which is a high-order convolution operator, make up the majority of CNN. There are various benefits to implementing CNN accelerator SoC designs utilizing an FPGA. Most FPGA-based CNN accelerators focus on a single FPGA design since using an FPGA offers the advantage of shorter latency and lower power consumption than using a CPU. Since the target CNN model tends to grow rapidly over time, however, the hardware resource in the FPGA may not be sufficient to accommodate the increasing size of the accelerator SoC design [1]. While it is not practical to keep replacing the FPGA with a larger FPGA, even the largest available FPGA may not be sufficient to implement the recent CNN accelerator designs.

An effective approach to overcoming this problem is using multiple FPGAs with high-speed interconnects. In this work, we present a method of implementing a CNN accelerator SoC on multiple FPGAs consisting of a master FPGA and slave FPGAs connected via Aurora interface IPs.

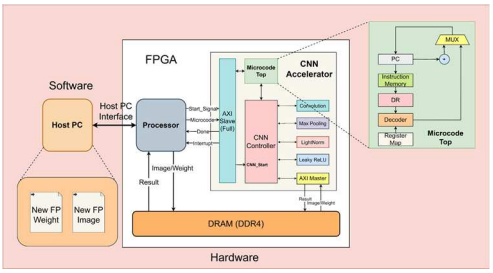

Figure 1. The Conventional CNN Accelerator Architecture

II. SYSTEM ARCHITECTURE

In most of the previous FPGA platforms for CNN Accelerator SoCs, all the components including a CPU core, a DRAM controller, a Neural Network Processing Unit (NPU) IP, and peripheral interfaces are all implemented in a single FPGA board as shown in Figure 1.

In the proposed system architecture, the components of the CNN accelerator can be partitioned into a set of FPGAs. Figure 2 illustrates an example FPGA platform consisting of 4 FPGAs. In this scenario, the four FPGAs connected to each other through Aurora Interfaces to transfer and receive the Instructions. Moreover, one of the FPGAs which has a CPU Core is connected to the host PC by Aurora Interface. One more FPGA is connected to external DDR by PCIe and serves as a Master for another FPGA in a linear sequence. It works as Daisy Chain to process each CNN Accelerator consecutively. Microcode is sent from the Master FPGA to the first Slave FPGA through Aurora, and this FPGA transmits the microcode to the next adjacent FPGA meaning when each FPGA receives a start signal and microcode, it begins to operate the CNN Accelerator, and the result of each operation is stored in DDR sequentially.

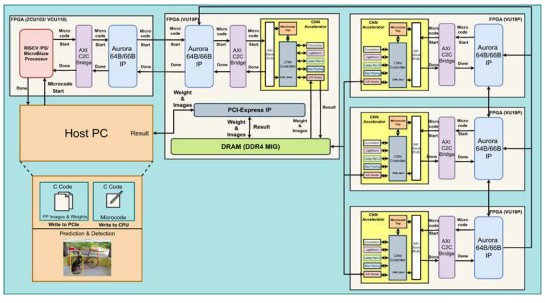

Figure 2. System Architecture for One Master And Four Slave FPGAs Communication via Aurora Interface and PCIe for CNN

In our current experiment demonstrated in this work, we only show our preliminary implementation using 2 FPGAs which are the Processor and CNN Accelerator. The first FPGA is a master FPGA (ZCU102 or VCU118) which is controlled by the Host PC. The master FPGA includes the CPU core based on either hard-core or soft-core IP, an AXI bus, a Chip2Chip Bridge, and an Aurora 64B/66B IP. The second FPGA is a slave FPGA (VU19p) which contains the DDR4 DRAM memory controller, PCI-express IP, and a CNN Accelerator as illustrated in Fig. 3

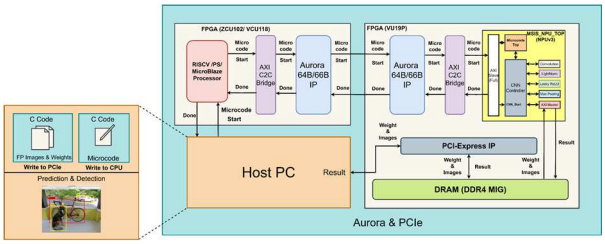

Figure 3. System Architecture for Two FPGAs Communication via Aurora Interface and PCIe for CNN Accelerator

III. Operation of Multi-FPGA CNN Accelerator

The CNN accelerator implemented on two FPGA platform is tested using a common CNN model called YOLOv2-Tiny, which is high-speed object detector aimed at mobile applications. The operation of the CNN accelerator is described by the following steps.

- Step1: (Control program download) Host-PC downloads to the CPU core's program memory in the master FPGA. The control program controls the start, finish, interrupt, extra computation of each layer of the CNN model.
- Step2: (Microcode download) Host-PC downloads to the microcode memory in the master FPGA the microcode. The microcode contains the instructions to control all the layers of the target CNN model, YOLOv2-tiny.
- Step3: (Weight and Image download) Host-PC downloads the weight parameters and test image data, respectively, to the weight memory region and the image region of the DDR DRAM attached to the slave FPGA.
- Step4: (Computing Each Layer) Control program of the CPU in the master FPGA initiates the microcode of the CNN accelerator. The accelerator IP loads up the weights, input feature data (or image), calculates the current layer's operations, and writes the resulting output feature data to the DDR DRAM. The control program repeats Step4 for all layers of the CNN.
- Step5: (Computing Each Layer) Once all layers' computations are finished, the control program transfers the final output feature data to Host PC. Host PC displays the detected objects with bounding boxes and calculates the average precision.

IV. Result And Conculsions

According to the results of the experiment, the data transfer from FPGA-to-FPGA for multi-chip design through PCIe and Aurora is roughly 2000 times faster than the data transmission over the old Host-PC Interface like UART. In the future, we plan to expand this data transmission for utilization in our CNN Accelerator purposes with 4 FPGAs.

Our analysis shows the fact that the maximum implementable size of CNN accelerator varies depending on the number of FPGAs used in the multi-FPGA platform. Table 1 compares the size of multiply-and-accumulate (MAC) arrays that can be implemented in one FPGA, two FPGAs, three FPGAs, and four FPGAs, respectively. The number of processing elements (PEs) in Table 1 indicates the size of MAC arrays. For the one-FPGA platform, the MAC array size is very limited, since one FPGA should accommodate all components of the accelerator SoC. On the other hand, for the multi-FPGA platforms, the implementable MAC array rapidly increases along with the number of FPGAs.

TABLE I. RESOURCE UTILIZATION AND PERFORMANCE

# of Device	1 VU19p	2 VU19p	3 VU19p	4 VU19p
Total LUTs in all FPGAs	4,085,760	8,171,520	12,257,280	16,343,040
LUTs used for Accelerator	2,722,783	5,434,634	8,146,485	10,858,336
Flipflops used for Accelerator	470,000	920,172	1,370,344	1,820,516
BRAM size used for Accelerator (Kb)	13,752	26,946	40,860	54,414

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant for RLRC funded by the Korea government (MSIT) (No. 2022R1A5A8026986, RLRC) and was also supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology). It was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2023-2020-0-01462, Grand-ICT) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This research was supported by National R&D Program through the National Research Foundation of Korea (NRF) funded by Ministry of Science and ICT (No. 2020M3H2A1076786, System Semiconductor specialist nurturing). The corresponding Author is HyungWon Kim.

References

[1] X. Inc, "Xilinx SP011 Aurora 64B/66B Protocol Specification, standards specification," 2014.

[2] Xilinx and Inc, "7 Series FPGAs Integrated Block for PCI Express v3.3 LogiCORE IP Product Guide."