PC process migration using FPGAs in ring networks

Keisuke Takano¹,a), Tetsuya Oda², Ryo Ozaki², Akira Uejima² and Masaki Kohata²
¹ Graduate School of Engineering, Okayama University of Science
Ridai-cho 1-1, Kita-ku, Okayama-shi, Okayama 700-0005, Japan
² Faculty of Engineering, Okayama University of Science
a) takano@kpc1.ice.ous.ac.jp

Abstract: This paper describes a method of process migration of PCs by FPGA-based ring networks. Here the original system configuration is given on the premise that the proposed complex cluster control circuit is used, and the communication protocol used on them is determined. The experiment results show that this system has the capability to perform PC process migration with FPGAs. The primary advantage of this system is that it is suitable for low-cost operations in which the PC is shut down after the process is entrusted to the FPGAs, and the result is obtained later.

Keywords: Hybrid Cluster, Interconnect Network, Dynamic Partial Reconfiguration

Classification: Network System

References

[1] K. Sano, and S. Yamamoto, “FPGA-Based Scalable and Power-Efficient Fluid Simulation using Floating-Point DSP Blocks,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 10, pp. 2823–2837, April 2017. DOI:10.1109/TPDS.2017.2691770
[2] K. Nagar and J. Bakos, “A Sparse Matrix Personality for the Convey HC-1,” 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines, May 2011. DOI:10.1109/FCCM.2011.60
[3] R. Ozaki, A. Uejima, and M. Kohata, “Implementation of a PC-FPGA Hybrid Cluster and Its Evaluation,” IEICE Transaction on Information and Systems, Vol. J96-D, No. 5, pp. 1313-1320, May 2013[in Japanese]
[4] D. Duolikun, A. Aikebaier, T. Enoikido, and M. Takizawa, “A process migration approach to energy-efficient computation in a cluster of servers,” Proc. of the BWCCA-2014, 2014. DOI:10.1109/BWCCA.2014.63

1 Introduction

A cluster system is regarded as a high-performance computing system, and there are many cases and various forms of cluster systems in which FPGAs are partially utilized [1] [2]. We propose a PC-FPGA complex cluster that
includes these various forms and connects PCs (CPUs and GPUs) and FPGAs equally [3].

On the other hand, process migration [4] is one of the key technologies used to improve cluster performance and availability. Here, process migration refers to the technology that moves the software processes running on one PC to another PC without interruption of any processes on either PC. In this paper, we propose a system in which FPGAs perform process migration of PCs used in laboratories (hereinafter referred to as “this system”). These FPGAs are connected to PCs and form a ring network. In this system, when PC processes are executed for a long time, they are entrusted to FPGAs in the middle of the execution. Meanwhile, the PC itself is shut down and is restarted the next day, and the execution result is received from the FPGA.

In this paper, we report the basic configuration and communication protocol for this system, and an image processing experiment conducted using them.

2 PC-FPGA Hybrid Cluster

2.1 Basic Configuration

Fig. 1 shows the overall system configuration and the FPGA circuit configuration. FPGA boards are installed in the PCI Express slot of each PC, and these FPGA boards constitute a ring network. Each connected PC-FPGA pair is called a node. Any user circuit (corresponding to PC application software) can be reconfigured dynamically in the user module using HWICAP. In addition, control register operation from a PC or user module and the DMA controller can be activated, and data can be transferred to any address. There are two address spaces for this purpose: one mapped to PCI Express and the other mapped to the internal bus. Table I (a) and (b) lists each address space. By specifying the internal bus address space in Table I (a), the PC can access the address space in Table I (b), i.e., the components of the FPGA.
### Table I. Memory Map

| Usage                  | Start Address | End Address  | Size  |
|------------------------|---------------|--------------|-------|
| (a) PCI Express Address|               |              |       |
| DMA                    | 0x00000000    | 0x000003FF   | 1 KB  |
| System Control         | 0x00000000    | 0x000003FF   | 1 KB  |
| Internal Bus           | 0x00000000    | 0x07FFFFFF   | 128 MB|
| (b) Internal Bus Address|             |              |       |
| Module(Unit)           |               |              |       |
| ICAP                   | 0x00000000    | 0x00000FFF   | 4 KB  |
| BAR2 Register          | 0x00100000    | 0x0010FFFF   | 64 KB |
| Router_PC              | 0x00200000    | 0x0020FFFF   | 64 KB |
| Router_UserModule      | 0x00400000    | 0x0040FFFF   | 64 KB |
| UserModule(Reg)        | 0x00600000    | 0x0060FFFF   | 64 KB |
| UserModule(User)       | 0x00610000    | 0x0067FFFF   | 448 KB|
| DRAM                   | 0x40000000    | 0x7FFFFFFF   | 1 GB  |

2.2 Physical Layer

The FPGA Mezzanine Card (FMC) on the FPGA is connected by a coaxial cable. The FMC is used in many FPGA boards such as Xilinx and Intel. The high-speed transceiver in the FPGA is used, and 4b5b code is used for encoding. The data transfer rate of the physical layer is set to 4 Gbps (line rate is 5 Gbps).

2.3 Data Link Layer

A packet consisting of a 1-word (32bit) header and 32-word data is used as the communication unit. The header contains the sender/receiver node address, sender/receiver port, and type. A router relays packets using store-and-forward. The 4b5b code has some control signals called K characters. After assigning XON/XOFF of the flow control to these K characters, the receiver controls the flow by sending XON/XOFF to the sender according to the state of the receiving buffer.

2.4 Network Layer

There are five types of packets: connection request, connection permission, DMA control, DMA data, and register control. The control information which includes these types of packets is specified in a header comprising 10 fields. The details are (22:20) Packet Type, (19:16) Source Port Address and Destination Port Address, (15) Virtual Channel, (14:12) Type, (11:8) Source Port and Destination Port, and (7:0) Source Address and Destination Address.

In Fig. 1, Source Address and Destination Address specify the source and destination node numbers, respectively. Source Port and Destination Port specify the source and destination within the node, which is either the PC or FPGA. In addition, the source and destination modules are individually specified using the Source Port Address and Destination Port Address. These header configurations enable flexible data transfer in a mixed PC and FPGA environment.
Fig. 2. Procedure for data transmission (PC0 to FPGA1).

To use this system for process migration, the communication procedure is determined so that reliable communication can be achieved in all possible communication paths using the router. For example, Fig. 2 shows a procedure for data transmission from PC0 to the DRAM of FPGA1. In this example, as the number of iterations of the data packet is described in the request packet, no termination processing is required. In this system, such a communication procedure is defined between PC, DRAM, HWICAP, and user module (including the same elements of different nodes). As described above, HWICAP is mapped on the internal bus address space and can be accessed from PCI Express, so that (for example) dynamic partial reconfiguration of FPGA1 can be performed from PC0 by executing this communication procedure with the write address as HWICAP and write data as partial reconfiguration data.

3 Experiment Results

3.1 Experiment Overview

The CPU used in each PC was an Intel Core i7-3820, and Ubuntu 18.04 was used as the OS. The Xilinx KC705 evaluation board was adopted for the FPGA. DDR3-1600 was used as the DRAM on the FPGA board. SYSTEC’s SYPCIE (runs on Gen2x4) was used for the PCI Express interface. The AXI interconnect was implemented to operate at 128 bit/160 MHz, and each FPGA module was implemented to operate at 160 MHz.

As a preliminary preparation, software to apply Laplacian filters to 1000 HD images (filter software) and circuits for the user module (filter circuit) were individually configured by using C language for the PC and Verilog for the FPGA. In the filter circuit, four FPGAs cooperatively apply Laplacian filters. Pre-filtered and post-filtered HD images on the FPGA side are placed in DRAM on FPGA0.

First, PC0 and FPGA0-3 were started, and PC1-3 were shut down. Next, after the filter software was started in PC0 and filter processing of 500 images
was conducted on PC0, the execution data and the filter circuit (dynamic reconfiguration data) were transferred to the DRAM of FPGA0 and HWICAP of FPGA0-3, and after the execution of the filter circuit was instructed, PC0 was shut down. Finally, PC0 was activated and the execution results were transferred from the DRAM of FPGA0 to the memory of PC0 by the interrupt from the user module of FPGA0.

3.2 Results
The experiment described in Section 3.1 showed that all images were correctly filtered. The number of used slices was 39479 when the filter circuit was transferred. The communication speed between PC and FPGA (Data size 1 GB) was 332 [MB/s] for transfer from PC0 to FPGA0 and 305 [MB/s] for transfer to FPGA1. On the dynamic partial reconfiguration of the filter circuit, the reconfiguration was completed in 31 [ms] in both FPGAs. The power consumption of PC1 + FPGA1 and FPGA1 in the experiment was 60.0 [W] and 21.0 [W], respectively.

This system was evaluated using the results described above. As for the number of slices used, approximately 22.5 [%] of the area went unused in the experiment environment. Although the user module needs to be secured as a rectangular region for dynamic partial reconstruction, the largest rectangular region was investigated, and it was found that approximately 34.6% of the region could be utilized. It was proven that power consumption could be reduced by 65.0% by turning off the PCs after the processing was requested from PC to FPGA. The speed was sufficient for the transfer of images in the process migration.

Overall, it was found that the intended behavior of process migration was realized. As demonstrated in the experiment, it is possible to shut down PCs temporarily after the process is entrusted to the FPGA, and the result is obtained later; this feature is not found in conventional systems.

4 Conclusion
This paper describes an FPGA circuit configuration and communication protocol for creating a system that can realize process migration of PCs using an FPGA circuit. Moreover, it was confirmed by the experiment that the intended operation of process migration was realized.

Currently, the network shape is limited to a ring, but in future work, we want to return to the design philosophy of a composite cluster so that the system can support various network forms.

Acknowledgments
This study was conducted with the support of the Xilinx University Program.