4J-07

System Software Of A Future Computer System With Mesh Interconnect

†Tomohiro Misono, ‡Kenji Kise

†Department of Computer Science, Tokyo Institute of Technology
‡Graduate School of Information Science and Engineering, Tokyo Institute of Technology

†misono@arch.cs.titech.ac.jp, ‡kise@cs.titech.ac.jp

1 Introduction

In conventional computers, it is hard for users to freely change the number of computer components like memories, because the expandability depends on the specification of the motherboard. This is because that conventional computers are interconnected by buses and the maximum number of components which can be attached is already determined. Moreover, an arbitration scheme becomes complex as the number of connections increases.

On a separate note, accelerators such as GPGPU have been used for high-performance computing along with CPU in recent year. Furthermore, recent studies have shown the effectiveness of an application-specific accelerator or processor implemented on FPGA. These facts show that we can enhance the entire computer system performance by flexibly using accelerators.

Based on these backgrounds, we predict that a new type of computer system will appear in the future which enables its users to easily change hardware configuration (add memories or I/O ports etc.) and uses various kinds of accelerators. Therefore we have proposed a new computer system composed of multiple FPGA boards which fulfills this characteristic. This project is currently under implementation.

In this system, each FPGA board has mainly one function such as a processing core, memory or an accelerator. We aim to create computer systems with high scalability and expandability by connecting these boards as two-dimensional mesh network. We adopt x86 ISA for the prototype architecture because of its popularity in commercial PC and variety of application binaries. In this paper, we discuss the software function required to fully utilize the proposed computer system and show the steps of the development of its software and system simulator.

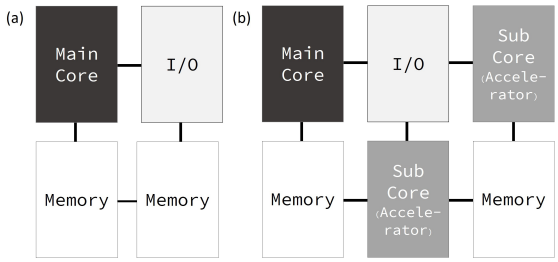

Figure 1: Examples of hardware configuration of the proposed computer system (a) four-board connection, (b) six-board connection

2 System configuration

Fig.1 shows the examples of hardware configuration block diagram of the proposed computer system. Each block corresponds to an FPGA board. These FPGA boards are connected in two-dimensional mesh network. Inter-FPGA communication is realized by exchanging packets through NIC (Network Interface Controller) on each board.

In Fig.1 (a), we use four FPGA boards and in Fig.1 (b), we connect six boards. Each board's function is as written in the block. There exists only one Main core board in computer system on which OS runs and controls other Sub cores as devices. Other boards (Sub core, memory, I/O) can be placed any places and any numbers according to users' demand. We believe this flexibility makes the whole systems scalable.

Note that we don't consider at the moment to connect several processing core boards as a multi core processor and run OS nor process switching in Sub cores (once a program starts in a Sub core, we have to wait until the end of the program).

3 Required software function

In order to run an operating system (mainly we consider Linux) on the proposed hardware configuration, we think the following firmware processing is

needed before operating system begin:

- recognize the whole hardware configuration
- detect the memory map (initialize NIC memory mapping table)

Main core is responsible for these initialization by communicating every board one by one using relative offset and checking if board exists when powered on. After initializing memory map, NIC uses memory mapping table to handle every memory access request (including access to the I/O region) to determine the position of the destination board.

In order to use Sub cores, we need device drivers because Main core recognizes Sub cores as devices. There is small amount of private memory in a Sub core. Main core sends an application program to that memory and instructs the start of execution. If more memories are needed, Sub cores allocate/free external memories through Main core. In order to realize efficient memory access, Main core (OS) should allocate memories in a memory board close to the Sub core which uses those memories.

4 Development framework

First, we use a function-level simulator to develop software such as device drivers or an efficient memory allocation scheme. After that we plan to create a cycle-accurate simulator to validate mesh-interconnect network and develop firmware program.

We use existing full system simulator. There are several open source full system simulators which support x86 ISA[1][2][3]. QEMU[3] doesn't do cycle-accurate simulation but is fast. Therefore, we use QEMU to develop our system.

There is an example of full system simulator using QEMU called MARSS[4]. MARSS supports dynamically switching between a cycle-accurate simulation mode and a fast emulation mode using QEMU.

We plan to use QEMU as a base full system simulator and modify or enhance its function to create simulation environment for the proposed mesh interconnected computer system.

5 Current Development

Currently, we have been developing a device driver which controls Sub core. As stated in Section 4, we currently focus on only device driver function and don't consider interconnections of system.

We use SimMips[5] because it is simple and has sufficient ability to simulate a Sub core function. SimMips is a MIPS software simulator written in C++. It can load a MIPS ELF binary and executes it.

We connected SimMips to QEMU as a virtual PCI device, and write simple device driver for Linux to control it. This time, we use SimMips internal memory and don't use memory outside the device. Because QEMU is written in C, we make a necessary modification to SimMips in order to connect it as a device. We checked that the implemented device driver successfully sends MIPS ELF binary to SimMips device via DMA transfer and the device executes the program correctly.

6 Conclusion

We have been developing a new computer system with mesh interconnect. In this paper, we have stated the needed software for that system. We plan to create firmware, device driver for Sub core (accelerator) or memory allocation scheme by using QEMU simulation framework. First we develop programs in function level and then validate it with low layer simulation.

To date we have completed a simple device driver for Sub core. Our next step is to enable Sub cores to use external memories in addition to their internal memory.

References

[1] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[2] bochs: The open source ia-32 emulation project (home page). http://bochs.sourceforge.net/.

[3] Qemu. http://wiki.qemu.org/.

[4] Avadh Patel, Furat Afram, Shunfei Chen, and Kanad Ghose. MARSSx86: A Full System Simulator for x86 CPUs. In *Design Automation Conference 2011 (DAC'11)*, 2011.

[5] Naoki Fujieda, Shimpei Watanabe, and Kenji Kise. A mips system simulator simmips for education and research of computer science. *IPSJ Journal*, 50(11):2665–2676, nov 2009.