IMPLEMENT AND EVALUATION OF A MULTI-CORE FPGA CALCULATION DEVICE ON A SMART HETERO-CLUSTER

Shinji Kawasaki and Kiyoshi Hayakawa
Osaka Prefectural University College of Technology Electronics and Information Course
Neyagawa, Osaka, 572-8572, JAPAN
email: hayakawa@osaka-pct.ac.jp

ABSTRACT
Recently, general-purpose processors have made the transition from high-frequency single cores to multiple cores as a consequence of multi-core processors requiring less power. To further reduce power consumption, it is necessary for field-programmable gate arrays (FPGAs) to embed multi-core processors. For smart cluster systems, it is important to reduce CO₂. Therefore, it is necessary for smart cluster systems to change dynamically the number of compute nodes including FPGAs as determined by the power consumption of the entire facility in order to reduce the total power consumption of the cluster system and other electronic devices. In this paper, we design a multi-core processor which is an integrated development environment for CPU set processor (ASIP core processor) using ASIP Meister. In the performance evaluations, we achieved a speed increase with eight cores of 7.8 times over that of a single core. The power consumption of eight cores is 2.58 times higher than that of a single core.

KEY WORDS
Heterogeneous Computing, FPGA, Multi-Core Processing, Low-Power Consumption, Application-Specific Instruction Set Processor

1 Introduction

Recently, general-purpose processors have been constructed as multi-core processors rather than high-frequency single cores. Currently, a standard processor consists of dual cores, while a high-end processor consists of eight cores. A multi-core processor is able to reduce power consumption, and as such, it is embedded in field-programmable gate arrays (FPGAs).

FPGAs tend to consume power of the order of tens of milli-watts, compared to multi-core processors and GPUs that tend to consume power of the order of tens of watts. One reason for the lower power consumption by FPGAs is that applications typically operate between 100–300 MHz on FPGAs compared to applications on high-performance processors, which typically run between 2–3 GHz. In these circumstances, CPUs execute operations faster than FPGAs, but the ability to parallelize applications on FPGAs results in increased performance-to-power efficiency of FPGAs[2]. Furthermore, since the fine-processing technology of semi-conductor manufacturing is improving rapidly, FPGAs may come to consume less power than expected.

For computer cluster systems, it is important to reduce power consumption and the space occupied by the system. PC clusters, which are made using general-purpose parts such as Intel or AMD CPUs, DDR memories, and SATA HDDs, are very cheap and easy to manufacture. Furthermore, since PC clusters have become considerably faster because of the high performance of the CPUs, only small companies and laboratories utilize them.

For PC clusters set in small companies and laboratories, it is necessary to fully consider power consumption and the space occupied by the cluster. PC clusters require larger and more fans to cool down the CPUs. Therefore, PC clusters that use low-power processors have been developed[3]. In order to reduce the power consumption, we must mount low-power CPUs on the PCs in the cluster. Recently, as a result of CPU vendors putting numerous types of chip on the market, it is easy to do so. Furthermore, it is possible to control the electric power consumption by changing the frequency and it's line voltage on the low low-power CPUs[4].

Meanwhile, in terms of reducing CO₂ and power consumption, we aim to operate a PC cluster in such a way as to balance out daytime and nighttime demand for power in the facilities such as small companies and laboratories. In order to reduce demands on power and space, we are developing Smart PC Hetero-Cluster (SPHC). SPHC consists of two types of nodes: CPU and FPGA (i.e., heterogeneous).

In terms of the FPGA calculation device, we implemented an execution environment on an FPGA for SPHC[1]. We designed an application-specific instruction-set processor (ASIP core processor) using ASIP Meister, which is an integrated development environment for CPU design. In the execution environment, we calculated the value of θ and the finite-difference time-domain (FDTD), and measured the execution time, circuit size, and electric power consumption.

However, we did not implement a multi-core processor. In this paper, we implement a multi-core processor on an FPGA calculation device. We design two types of multi-core processor architectures generated using ASIP Meister. To evaluate the performance, we measure execution time, circuit size, electric power consumption, and total power consumption.
2 Related Works

Cluster systems that are embedded on FPGAs have been studied and developed. The Cray XD1 computer achieves 58 GFlops with 12 Opteron CPUs and 6 Xilinx Virtex-II FPGA devices on a single motherboard[5]. A cluster with 64 Virtex-4 FPGA devices was built in the Maxwell project[6].

The Quadro Plex (QP) Cluster[7] was built at the University of Illinois at Urbana-Champaign. For each of the 16 nodes in the QP prototype, there are two AMD Opteron CPUs, four nVidia G80GL GPUs and one Xilinx Virtex-4 LX100 FPGA. This system can achieve 23 TFlops theoretically. The Axel cluster[8] from Imperial College London demonstrated collaboration between heterogeneous accelerators. With Xilinx Virtex-5 FPGAs and nVidia C1060 GPUs working together, this 16-node cluster running an N-body simulation achieved a speed up of greater than 22 times over a 16-node CPU-only cluster.

Financial applications in banks usually require substantial computing power for either processing huge amounts of raw data or simulating pricing models repeatedly. FPGA platforms have been used to accelerate Monte-Carlo simulations for financial applications[9].

FPGA Multi-core systems have been also studied. Clark[11] has studied issues related to multi-core in an FPGA. He has developed an FPGA multi-core consisting of PowerPC core and four MicroBlaze cores, and implemented DNA sequencing program. Hofmann[12] has proposed SDVM, the scalable dataflow-driven virtual machine. The SDVM is well suited to serve as a managing firmware for multicore FPGAs. An application has been tested on the one, two and four core systems including combinations of PowerPC and MicroBlaze cores.

The above studies do not discuss either the electric control method or load-balancing methods. Furthermore, they do not consider ASIP. We have proposed electric-power control methods[13] and load-balancing methods[14][15]. In this paper, we discuss ASIP on SPHC.

3 A Working Environment for SPHC

In this section, we explain a working environment for SPHC that aims to balancing out the power consumption.

3.1 An Overview of the System: Balancing Out the Power and Reducing \( \text{CO}_2 \)

Figure 1 gives an overview of the system. SPHC changes the number of work nodes and the load of the work nodes in accordance with the total power consumption of the facility. That is, SPHC works as a small cluster during the day because the air-conditioner and office PCs use more power, and as a large (full) cluster at night. SPHC controls its power consumption by watching watt-monitors that measure the facility’s power consumption. SPHC switches off the compute nodes when power consumption changes by large degree (i.e., coarse-grain control). On the other hand, SPHC reduces the CPU load when the power consumption changes only slightly (i.e., fine-grain control).

4 SPHC

4.1 System Organization

Figure 2 depicts the organization of an SPHC system that is constructed as a test bed of a working environment[13]. It consists of watt-monitors, an embedded middle-density cluster (EMDC) system[14][15], low-power chassis nodes, and a host computer.

The EMDC was built using PC boards for the node processor, an intra/inter chassis network for the network, and a host computer for the control-node processor. The node processors consist of Pentium III (600 MHz or 700 MHz) and Pentium M (2 GHz) processors. The Pentium III nodes have 256 MB of memory and the Pentium M node have 1 GB memory. Three nodes (called the center node, right node, and left node) are packed into one chassis each. The host computer consists of a Pentium 4 (3.0 GHz, Hyper Threading) processor with 1 GB memory (dual channel). The EMDC employs compact flash memory (1GB) instead of an HDD to reduce power consumption.

4.2 Organization of the Network

The network of the EMDC consists of two networks, an intra-chassis network and inter-chassis network. The former is characterized by a low-latency network. In this paper, the intra-chassis network is Ethernet, rather than low latency. The latter, on the other hand, is characterized by a relatively high network throughput.

The intra/inter-chassis network fit together for a tree-network topology. Since each node in a chassis is able to communicate frequently via intra-chassis network, we assume that the chassis has three links on inter-chassis net-
work. The chassis, therefore, is able to construct a tree-topology network.

The low-power chassis nodes each consist of an Atom motherboard ([A-node] in Figure 2) and an FPGA board ([F-node] in Figure 2). A Xilinx SPARTAN-6 is embedded in the FPGA board. 3 A-nodes and F-nodes (i.e., 6 nodes in total) are packed into one chassis. An A-node connects to an F-node via USB to send (or receive) data. The network of the A-node also provides two networks, the intra- and inter-chassis networks. The A-node switches on/off using Wake-on-LAN. A node controller in the chassis sends a Wake-on-LAN signal. The host computer controls the power consumption using the node controller indirectly.

The watt-monitors measure the power consumption of the Pentium M CPU chassis, the Pentium III chassis, the Atom+FPGA chassis, and the office's electronic equipment. They then send the data to the host computer via Ethernet. The host computer can accordingly control and balance out the power consumption.

5 FPGA Calculation Device

5.1 Design Strategy of CPU Cores

There are two steps needed to design FPGA with ASIP. We begin with the design of a single-core CPU, consisting of instruction memory, data memory, and peripherals. ASIP uses Harvard architecture. We design an instruction-set architecture for an application. For example, in cases where an application uses substantial multiplication and sequential addition operations, we would incorporate multiply-accumulate (MAC) operations. In cases where the application requires floating-point operations to be carried out, we would embed a floating-point unit into the CPU core.

After designing the single core, system performance is enhanced with multi-core co-processors. We employ a wishbone as a shared bus because it is easy to implement. However, ASIP does not employ a wishbone bus as a peripheral bus. Thus, interface logic is inserted between the ASIP’s bus and the wishbone bus.

ASIP cores communicate with each other using shared memory. In order to execute barrier and exclusive control, ASIP uses test-and-set (TAS) operation, which uses a synchronization manager.

5.2 Organization of FPGA Calculation Device

Figure 4 shows the organization of the FPGA calculation device (called the F-node). It consists of an Ethernet controller, a control core, and a multi-core processor. The control processor handles the multi-core processor. The host computer sends the control operations to the control processor via the Ethernet controller. The layers below the network layer are implemented in the hardware, while the upper transport layers are implemented in the control processor. Therefore, we design a control processor that has special operations of network and multi-core control by ASIP. We intend to embed a real-time OS into the control processor, and use it to manage multi-core tasks.

5.3 Organization of Multi-Core Processor on FPGA Calculation Device

Figure 5 shows the organization of two or four cores, and Figure 6 shows that of eight cores. Each ASIP core bus is connected to the peripheral bus (wishbone bus) by the interface. The interface selects an ASIP core bus according to a grant signal by the arbiter, and then connects to that ASIP core bus and peripheral bus. The arbiter uses a round-robin method to select the core bus. We employ two arbiters on eight cores to prevent long waiting times. On two or four...
cores, an ASIP core is granted access to peripherals by a single arbiter in a short wait time, but on eight cores, a single arbiter would mean that ASIP cores may have to wait for a long time.

There are two buses on the multi-core processor. In order to communicate from one ASIP core to another ASIP core that belongs to another bus, the FPGA calculation device employs a dual-port memory (D-mem and 2-port D-mem in Figures 5 and 6, respectively) and a synchronization manager (Sync Mgr. in Figures 5 and 6). The dual-port memory allows two ASIP cores (separated at the bus) to read data at the same time and the same address, but it does not allow them to write data at the same time and the same address. That is, memory access conflicts may occur. Thus, the ASIP cores control access of the 2-port D-mem to prevent these conflicts by the synchronization manager.

An ASIP core reads data to the synchronization manager before writing data to dual-port memory to prevent the ASIP core from memory access conflicts.

The synchronization manager executes a TAS operation when the ASIP core reads data to the synchronization manager. That is, reading to the synchronization manager corresponds to an automatic TAS operation. If a flag is read by the ASIP core that is 0, the synchronization control mechanism sets the flag to 1 and sends 0 to the ASIP core automatically. 0 indicates that the ASIP core has set the flag and has entered a critical section. On the other hand, if the flag is 1, it sends 1 to the ASIP core only. 1 indicates that the ASIP core failed to set the flag and to enter the critical section. If the ASIP core exits the critical section, it writes data to the synchronization manager. When the synchronization manager receives the write data, the synchronization control mechanism sets the flag to 0.

If two read accesses occur at the same time through the dual port (port0 and port1), the synchronization control mechanism gives priority to port0 access.

5.4 Synchronization Manager

The synchronization manager consists of eight synchronization flags and a synchronization control mechanism.

5.5 Application Specific Instruction Set Processor

We employed Brownie STD 32 as the ASIP core. Brownie STD 32 is a small 32-bit processor suitable for embedded systems. It is not used as a standalone processor but is rather customized by ASIP Meister. It is a RISC processor that has 32 general-purpose registers (GPR0–GPR31). Four of the registers are reserved for the processor’s specific use. We use the others as general-purpose registers in our software. Brownie STD 32 has 45 instructions that are suitable for the standard use of the embedded systems. Brownie also has a four-stage pipeline. Data forwarding on the pipeline allows the following operations to use the data that are obtained by forwarding operations.

Figure 7 shows the design flow of ASIP Meister. ASIP Meister generates information about the processor and the software. That is, ASIP Meister generates VHDL codes, such as source code and test-bench code.

We set up the design of the ASIP core processor to include data such as the number of pipeline stages, circuit
size, and power consumption. In addition, we defined resources inside the ASIP processor core such as PC (program counter), IR (instruction register), and ALU (arithmetic and logic unit). We also set up information regarding registers and memory. Furthermore, we set up data bus width, instruction-set bus width, and so on.

ASIP Meister generates the compiler, assembler, and linker as software tools. We were able to program an application in C using the software tools. The assembler and linker generate object code. ASIP Meister also generates VHDL code and test bench code from the information of the ASIP core processor. ASIP Meister then generates reports of the processor from the VHDL test bench and object code. The reports refer to area, delay, electric power consumption, and so on.

![Diagram](image_url)

Figure 7. Design flow of ASIP Meister

6 Evaluation

For evaluation purposes, we measured execution time, circuit size, and power consumption of the multi-core processor. The target FPGA calculation device was the Xilinx SP605 board.

We measured the execution time of a program that searches for prime numbers, a simple program due to the manner in which its calculations scale, the ease with which it can be parallelized, and its low memory requirements. We used the timer of a PC connected with an FPGA calculation device. The PC sends a start signal when the program is started, and the timer of the PC is also started. The FPGA calculation device sends an end signal to the PC when it finishes the program and the PC stops the timer. The program searched for prime numbers between 1000 and 20000.

Circuit size and power consumption were determined by using the Xilinx ISE Design Suite. We analyzed various aspects of the power consumption, such as dynamic power and quiescent power, with Exiling XPower Analyzer.

6.1 Evaluation Results

Figure 8 shows the execution time of search numbers from 1000 to 20000. On 20000 search numbers, we achieved good speed-up ratios: 1.88 times for two cores, 3.7 times for four cores, and 7.79 times for eight cores.

![Graph](image_url)

Figure 8. Execution time of the prime number search program

Figure 9 shows the circuit size of a multi-core processor (two, four, and eight cores). The circuit sizes of two, four, and eight cores are 1.88, 3.43, and 6.74 times larger than a single core, respectively. The ratio of the increase in circuit size is smaller than the speed-up ratio. The circuit size of the arbiter and the synchronization manager has little impact on the ratio of increase in circuit size. The circuit size of dual-port memory, dual bus, dual arbiter, and dual synchronization manager also has little impact on the ratio of increase in circuit size.

Figure 10 shows the power consumption of a multi-core processor. The more cores the FPGA device embeds, the more dynamic power it uses. Quiescent power consumption, however, is constant despite the increasing number of cores. The ratio of increase of power consumption is lower than the speed-up ratio. The ratio of increase in power consumption \( R_p \) expressed numerically gives the following approximation:

\[
R_p \approx 1 + \alpha(2^n+1 - 1),
\]

where the number of cores is \( 2^n(n=0,1,2,3) \) and \( \alpha \approx 0.1 \). On the other hand, the speed-up ratio \( R_s \) is

\[
R_s \approx \beta 2^n,
\]

where \( \beta \approx 1 \). Therefore \( \alpha \) is 10 times smaller than \( \beta \).
Figure 11 shows the total power consumption of multi-cores and AMD’s FX-4100. The FX-4100 is a four-core general-purpose processor. Since the prime number search program is a single-thread (core) program, the total power consumption of the FX-4100 is estimated as 1/4 the thermal design power (TDP). This estimated value about FX-4100 is only advisory, and it may decrease. The more cores the FPGA calculation device embeds, the lower total power it uses. Total power consumption of eight cores is 1/3 lower than that of single cores and 1/3.54 lower than that of the FX-4100 (single core).

7 Conclusion and Future Works

In this paper, we implemented two, four, and eight cores on an FPGA-based calculation device and evaluated their performance. We designed processor cores using ASIP Meister. The two- and four-core processors employed one-bus architecture, while the eight-core processor employed two-bus architecture. To evaluate the performance, we measured the execution speed, circuit size, electric power consumption, and total power consumption. We achieved a speed-up over a single core by 7.79 times with eight cores. The circuit size of eight cores was 6.74 times larger than that of a single core. The electric power consumption of eight cores was 2.58 times higher than that of a single core. The ratio of increase of electric power consumption was 1/10 lower than the speed-up ratio. Future works are as follows:

Future works are as follows:

- We will develop a power-consumption control system that includes a low-power CPU and FPGA calculation device.
- We will implement a scientific calculation program such as FDTD on an FPGA multi-core calculation device, and evaluate it.

References


Figure 11. Total power consumption of multi-core and FX4100 (estimation) processor


