Asymmetric Multi-Processing Mobile Application Processor MP211

By Sunao TORII,* Junji SAKAI,* INOUE, Hiroaki,* Tatsuya TOKUE† and Yoshiyuki ITO†

ABSTRACT We propose several techniques of Asymmetric Multi-processing (AMP) for mobile application processors. In our AMP architecture, multiple general purpose processors are integrated on a chip to reduce hardware development period and cost. Our techniques are well considered the role of hardware and software to maximize the benefits of parallel processing. The hardware is carefully designed to enlarge bus bandwidth, shorten memory latency, and reduce power consumption. Our software techniques realize application-level compatibility with single processor and excellent real-time performance with solving multi-processing issues. Moreover, our novel security techniques enable to execute download non-secure native application. We implemented these techniques on the mobile application processor MP211. Evaluation results show our AMP techniques on MP211 are very useful for future mobile terminals.

KEYWORDS **Multiprocessor, Application processor, Low power, Parallel execution, Security, Mobile**

1. INTRODUCTION

The applications of cellular phones have recently been evolving very rapidly. It is expected that high resolution real-time multimedia applications and comfortable multi-tasking environments will be realized very soon. Although these advances will require more and more processing performance the secure execution of downloaded native applications and media contents delivery services are confidently anticipated. On the other hand, a low power requirement is still a critical issue. Therefore, the next generation mobile application platform has to realize high performance and low power multi-tasking as well as a more secure environment.

Parallel processing using multi-core technology is one of the best ways to meet these requirements because it can achieve higher performance easily, with reasonable hardware resource and a relatively low clock frequency. Task distributing to multiple processors contributes to an improvement in real-time performance and whole system security. Therefore we are investigating how to apply parallel processing for mobile terminals and we conclude that asymmetric multi-processing (AMP) is one of the most suitable approach for mobile terminals in terms of current techniques and requirements.

In the AMP environment, each OS runs on each processor independently and each task is allocated on a specified processor. A task is executed serially and parallel execution is performed among multiple tasks. This means that task scheduling flexibility is limited. However, there are several advantages in AMP as follows,

- Easy hardware implementation: It can use an existing processor core even without a cache coherent mechanism. This contributes to a shortening of the hardware development term and a reduction in costs.
- Reducing Power Consumption: Distributing each task to multiple processor cores appropriately decreases clock frequency while maintaining total system performance. Lower clock frequency with lower supply voltage enables us to reduce power consumption.
- Scalable: Performance and cost adjustments can be realized only by increasing/decreasing the number of processor cores.
- Boosting real-time performance: Each application can allocate a different processor to each other. This feature reduces interference among multiple applications.
- Improving system security: System software and downloaded non-secure native applications can be

*System Devices Research Laboratories
†NEC Electronics Corporation

separated by executing them on each different processor.

This parallel execution model is called the 'Multitasking parallel method.' Our method considers how to improve the system performance throughput by executing multiple heavy applications simultaneously. **Figure 1** illustrates the task allocation of our AMP multitasking parallel method.

In this paper, we describe several AMP hardware and software techniques. We also describe the real implementation and evaluation for the MP211 application processor that has been developed using our AMP architecture. In the following three sections, we will describe the hardware and software design concepts and the security method. After that we will explain about the real implementation of MP211 and evaluate its performance. Finally, we summarize this work and show the way to the next research stages.

2. HARDWARE DESIGN

2.1 Enlarging Bus/Memory Performance

To maximize AMP benefits under the condition of running multiple tasks at the same time, we have to avoid performance degradation due to resource conflict. Usually, in many applications, the peak loads are repeated periodically. In the same power range, the combined performance of parallel processors can exceed that of a single high-speed one. So the total memory load fluctuation range will be increased. In the same application mix, although the amount of memory demand is almost the same as for a single processor, the peak is possibly too heavy when all processors run with enough load.

The conventional approach to enlarge bus and memory performance is to improve the clock frequency and widening the bus bit width. However, this method causes several serious problems such as rising power consumption, increasing relative memory response cycle and decreasing connectivity to legacy IPs. Another technique is split transaction enlargement of the access scheduling scope to improve bus and SDRAM transfer efficiency. However, this results in the bus control logic becomes very complicated. So we propose 3 new techniques to improve bus and memory ability as follows and as shown in **Fig. 2**[1]:

- Categorized partially connected multiple system bus architecture
- Read and write data buffering for SDRAM accesses packing,
- SDRAM access scheduling among multiple bus interfaces and buffers.

In the first technique, multiple system buses and interfaces are provided for the main memory and each bus connects with fewer master IPs. Each bus operates independently and is fully compatible with standard bus protocol. Moreover, each master IP is categorized into two groups, response sensitive and throughput. In general, processors require short memory response to realize comfortable user interfaces and for handling rapid interruptions. However, they do not require a large memory bandwidth. On the other hand, Graphics and DMA require a wide memory transfer bandwidth. Accordingly, we gather response sensitive master IPs into response sensitive group buses, while the throughput master IPs are connected to the other buses. Therefore, the response sensitive accesses are separated from large traffic media transfers.

This method has several advantages, such as reducing the number of connections in each bus, retaining compatibility and ease of connection to most IPs, and the possibility of short memory latency for response sensitive IPs to coexist with heavy multimedia applications.

The second technique is data buffering for SDRAM

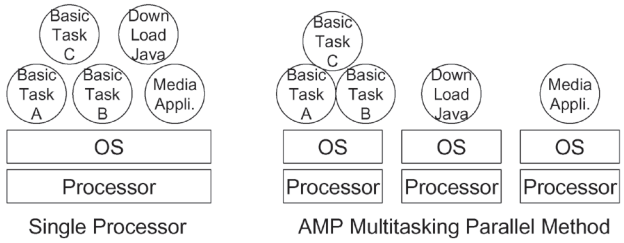


Fig. 1 AMP task allocation image.

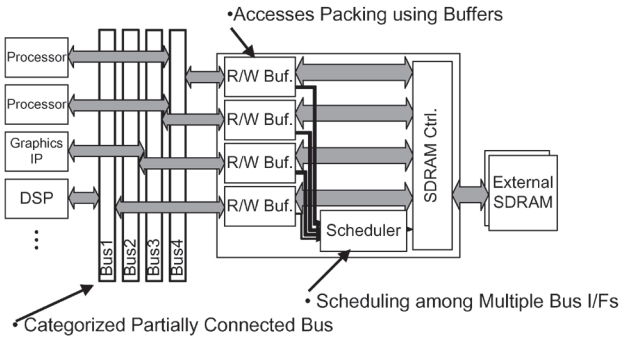


Fig. 2 Bus and memory architecture.

accesses packing. To maximize bandwidth and to meet the cache filling operation, SDRAM is used for burst data transfer. However, most peripheral IPs and un-cached processor access use single and short burst transfer. They degrade SDRAM access efficiency. So we prepare to read and write buffers in each bus slave interface. A read buffer retains the entire burst SDRAM response. If the succeeding access hits the data, it is provided from the buffer instead of SDRAM. Write data is held temporarily in the write buffer in order to immediately release bus occupation, and the write buffer merges a succeeding write data if possible. After completing the whole of data filling in the write buffer, these data will be written back to the SDRAM. This technique contributes to shorten response time and reduce SDRAM accesses.

The last technique is access scheduling. Generally, optimized SDRAM access scheduling by using multiple banks such as avoiding bank conflict tends to increase the SDRAM throughput. This sensible access scheduling is realized by a lot of data access candidates and by choosing the most suitable requests from these candidates. Multiple buses and write data buffering enlarges these candidates because many memory access requests are issued from each IP and the write buffer simultaneously.

2.2 Reducing Power Consumption

For mobile terminal use, lower active power extends application running time and reduced static current lengthens the stand-by time. Since in many media applications, peak load appears periodically, the power reduction of the short range idling time is very important. Therefore we focus both short range idling time power reduction and stand-by leakage power reduction as well as active power reduction with lower clock frequency and supply voltage.

To reduce short range idling power consumption, clock tree root level gating, dynamic voltage and clock frequency control by fully hardware logic should be applied. The system management unit keeps track of the whole system status and if it detects a preset idle state, it handshakes with several IPs to change the clock frequency and enables voltage safely. Since this handshake method requires only 10 to 100 clock cycles, so this clock frequency control works very effectively to reduce short range power consumption.

For the standby leakage reduction, our AMP architecture provides a simple programmable sequencer based power management unit (PMU) shown in **Fig. 3**. Because most IPs including processors are shut down during standby mode to minimize static current, a processor cannot change the system power mode. But fully hard wired power control logic cannot keep flexibility for any systems. The PMU consists of a simple sequencer and small-step SRAM for program instructions, so the static current is much smaller than for the general purpose processor. It provides several functions by only operating peripheral modules and power control. Consequently, a dramatic power reduction is realized during the stand-by mode.

3. SOFTWARE PLATFORM

Our basic idea is to put an independent OS instance for each processor and to run multiple applications on these multiple OS instances. This architecture brings advantages such as a good real-time performance and improved system security, as described in the introduction. However, the following issues remain to be resolved:

1) We cannot communicate with tasks on the different processors, since each OS instance only supports inter-task communications within a single processor.
2) Hardware accesses may bring unexpected results, because each OS instance does not prevent simultaneous hardware resource accesses from multiple processors.

On the other hand, we cannot overlook the customers' requirements of software compatibility and easy maintenance ability. Introduction of a novel OS kernel or distinct new libraries urges customers'

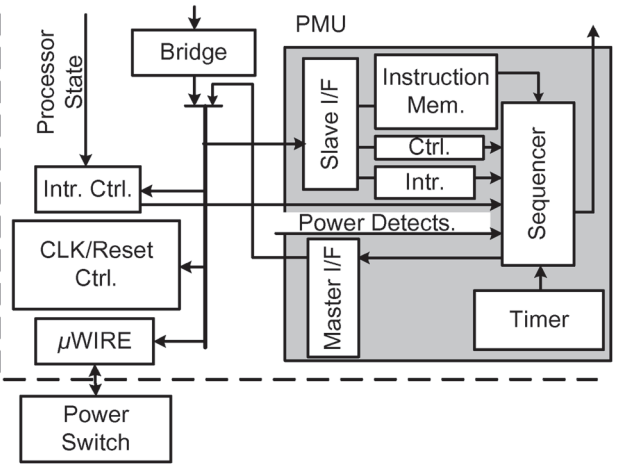


Fig. 3 PMU sample implementation.

applications to be modified, which is not welcomed.

Our approach is as follows: For the first issue, we have designed "OS Wrapper" shown in **Fig. 4**. It is a software layer between the OS kernel and user applications. It hooks the function calls of inter-task communications such as message passing and semaphore operations, and transmits the information to the destination processor through our inter-processor communication mechanism. Since these communications are done outside the OS kernel, we place proxy tasks to handle wakeup events. For example, when a message arrives to a blocked task, the proxy task sends a signal to the blocked task so that the blocked task is activated and resumes its execution.

As for the second issue, we have introduced the Client-Server model. For each hardware resource, we put an "access server" on one OS and only the access server is permitted to access the specified hardware resources(**Fig. 5**). Other tasks on each processor request the hardware resource access to this access server. We also prepare client libraries which act as a bridge between client side communication modules and the application body.

The OS Wrapper and the hardware access servers use our inter-processor communication hardware. It is such fast on-chip logic that the communication overhead is quite small.

We also note that the OS Wrapper and the hardware access server's mechanisms are designed within UserLand programs and device drivers. It is separated from the OS kernel, so it is easy to catch up with subsequent kernel version changes.

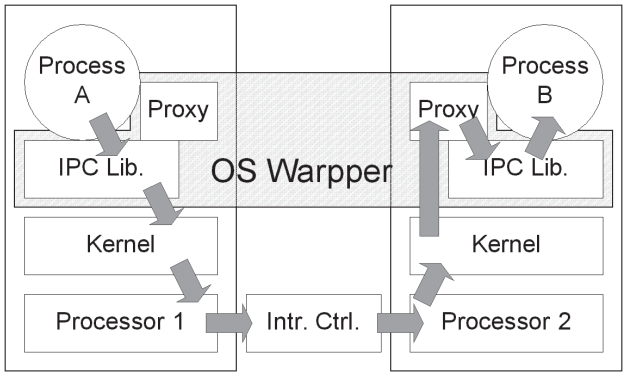
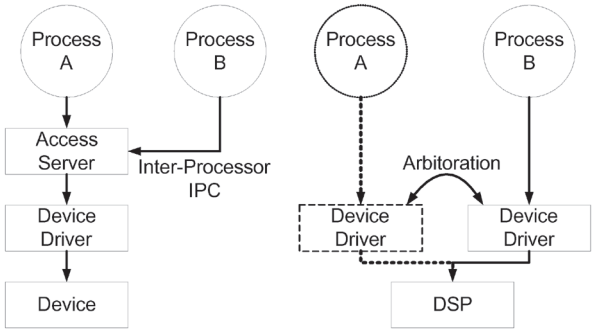


Fig. 4 OS wrapper and inter-processor communication.



a) Access Server **b) Direct Access**

Fig. 5 Shared resource accesses.

4. SECURITY FEATURES

Another advantage of AMP is the easy to improve whole system security because the system software and downloaded non-secure native applications can be separated by executing them on each different processor. We propose a domain separation method according to the security level[3](**Fig. 6**). This method composes three domain levels, "Base Domain," "Trusted Domain" and "Untrusted Domain." The base domain contains the basic functions of a mobile terminal, such as a mailer and a browser. Downloaded applications validated by communication carriers as being trustworthy are executed on the trusted domain. All other downloaded applications are executed on the untrusted domain.

Each processor belongs to one of these domain levels and only an application that fits into the security domain level can be executed on each processor. In

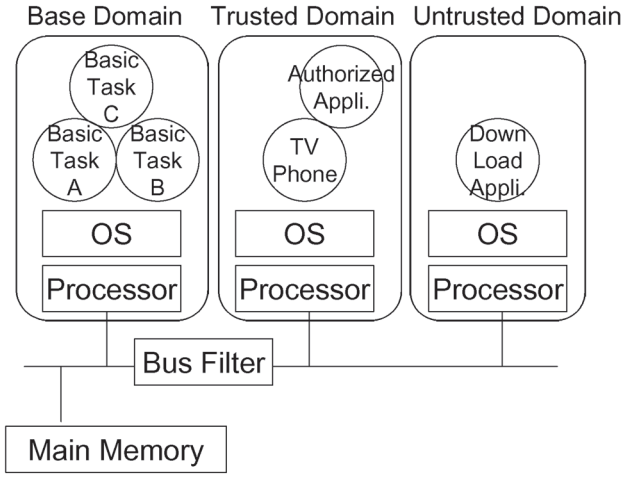


Fig. 6 Policy domain separation.

our multi-tasking parallel method, since each OS is running independently on each processor, the impact of an untrusted application behavior can be limited only to an untrusted domain processor. This means the basic functions of the mobile terminal are guaranteed regardless of executing untrusted applications.

5. MP211 IMPLEMENTATION

We have implemented these AMP techniques to the mobile application processor MP211. **Figure 7** shows a block diagram of this chip, which integrates three ARM926 processor cores (Processing Element PE0-2), DSP, graphic accelerators, 512KB SRAM, a DDR SDRAM interface and other intellectual properties (IPs) (**Table I**, **Fig. 8**). Each IP is connected by 32bit Multi-layer AHB. MP211 runs up to 192MHz (ARM926, DSP) and 96MHz (AHB, Most Graphic IPs).

MP211 allocates four AHBs for one SDRAM I/F and each IP is connected to one AHB. Two AHBs are for response sensitive IPs and the others are through-put IPs. In order to reduce the power consumption, the MP211 applies clock tree optimization, clock root-level gating, automatic idling time clock frequency reduction to 12MHz, PMU based power domain control and multiple Tox/Vt CMOS design.

We implemented our AMP software schemes to Linux OS on MP211. Since only the intra-processor process communication method is provided except for the INET domain socket on conventional Linux, we developed a "UNIX domain socket wrapper" and a "System V IPC wrapper" to realize both inter-processor UNIX domain socket and system V IPC.

6. PERFORMANCE EVALUATION

First of all, we measured the H.264 video decoder (QVGA 15fps) and the MPEG2 AAC decoder (48K Stereo 128kbps) in order to evaluate the MP211 power characteristics. These applications propose that a terrestrial broadcasting digital television (DTV) viewer runs on a DSP, co-operating with a video-sound de-multiplexing task on an ARM926. **Figure 9** shows the transition chart of the power consumption. L0 area means the power of the always on area and L1 area indicates that other areas are executed with the highest IP power consumption.

This result shows that the MP211 consumes 87mW (exclude I/O, SDRAM), 124mW (includes I/O, SDRAM) average power on this DTV program. It also

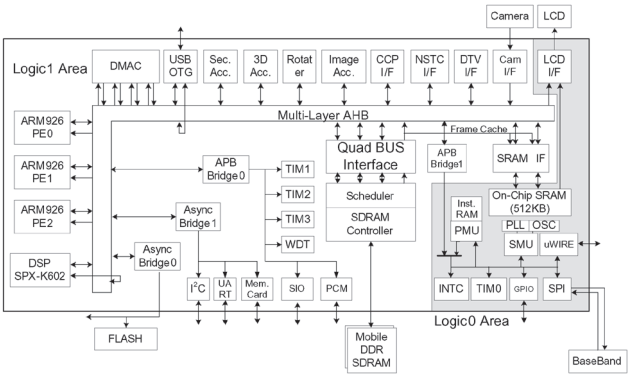


Fig. 7 MP211 block diagram.

Table I MP211 Chip Specifications.

Item	Specifications
Die Size	8.80 × 8.65mm 0.13μm 6-Metal Layer
Frequency	192MHz (ARM, DSP), 96MHz (AHB, OtherIPs)
Supply Voltage	1.2V (Internal), 1.8V (I/O)
Memory	512KB (On chip SPAM), 128KB (DSP Mem.), 16KB (DSP Cache), 16K+16K (ARM) × 3 512Mb Mobile DDR SDRAM × 2 integrated in SIP

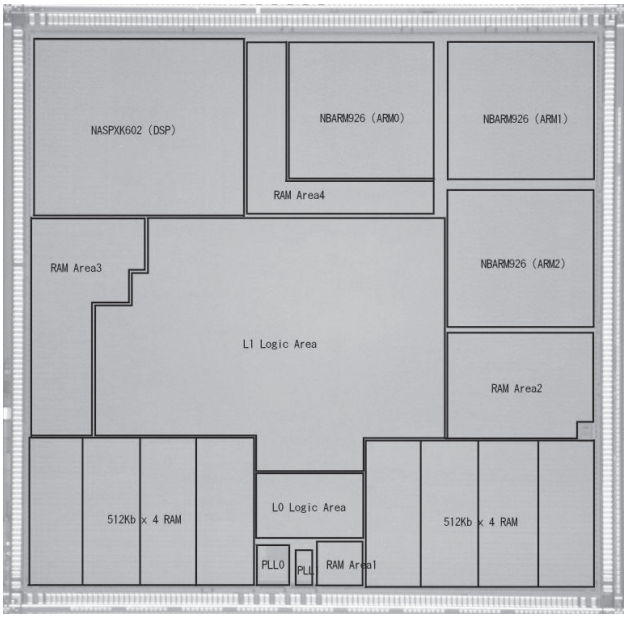


Fig. 8 MP211 die photograph.

shows that the idling time L1 area power is less than 10mA. This means that the automatic clock frequency control works very well.

Then we assembled a real time multimedia application mix for a near-future high-end cellular phone. This mix was composed of a DTV viewer, an RSS (Rich Site Summary) news reader and an HTML browser on an X-window manager with 3D-icons. All tasks are assigned to three-PEs and DSP as shown in **Fig. 10**. In order to compare with the conventional single processor performance, we also measured a 1PE environment model indicating that all PE tasks are the same as formerly allocated to one PE.

Figure 11 shows jitters of audio task intervals for the 3PE and 1PE execution. We observed occasional breaks of sound play and lost frame in 1PE execution although the average demand of total processor tasks does not exceed 1PE performance. This figure shows there are some long intervals that exceed maintaining real-time operation time on 1-PE execution. In contrast, in the case of 3-PE execution there is no long interval, so the execution is achieved very smoothly. Furthermore, we also evaluate 3-PE execution with a half processor clock to eliminate the effects of the processor performance and cache memory size, this also helps to realize smooth execution. This is because the real-time performance is too strict by 1-PE execution. Although some scheduling techniques reduce the number of sound breaks it is difficult for 1-PE execution to realize perfect execution on Linux.

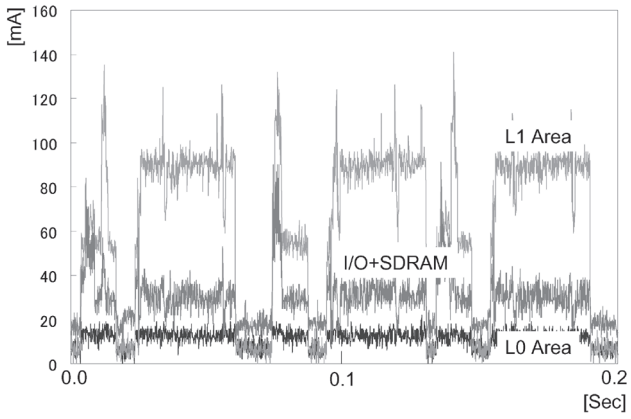


Fig. 9 Power consumption of H.264+AAC.

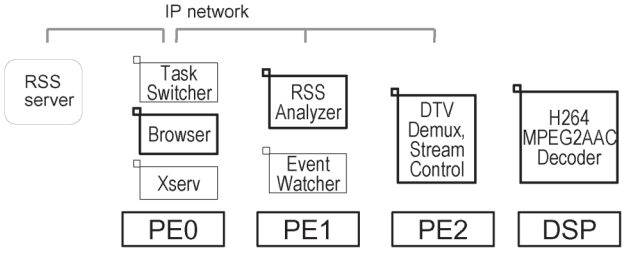


Fig. 10 Task allocation on MP211.

7. SUMMARY AND FUTURE WORK

In this paper, we have shown that our AMP techniques and the real implementation of MP211. MP211 achieves only less than 90mW power consumption with carrying out DTV viewer on Linux and perfect real-time application mix execution. We also provide a domain separation security technique that improves the resistance level of system level failures dramatically. These developments mean that our AMP architecture is one of the best ways to realize a high-performance mobile application processor.

However, boosting single task performance is a difficult feat for our AMP architecture because each processor executes each operating system independently and there are no cache coherency maintaining mechanisms among the processors. In fact, our research consists of three stages, 1) Asymmetric multiprocessing (AMP), 2) Symmetric multi-processing (SMP) and 3) Speculative control-flow level multiprocessing (CFP). The SMP supports hardware cache coherency management. Moreover the CFP supports architecture and code-translation level automatic single task parallelization. These techniques enable us to improve single task performance without boosting the clock frequency. Ultimately we will combine three parallel execution architectures and switch the current execution mode dynamically according to the current application mix.

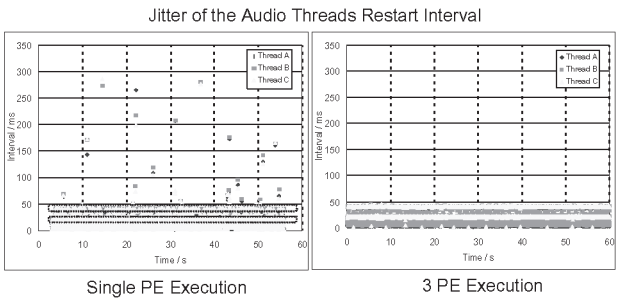


Fig. 11 Jitter of the audio threads result interval.

REFERENCES

[1] S. Torii, et al., "A 600MIPS 120mW 70μA Leakage Triple-CPU Mobile Application Processor Chip," *2005 IEEE Int. Solid-State Circuits Conf., Digest of Technical Papers*, pp.136-137, Feb.2005.

[2] J. Sakai, et al., "Multi-Tasking Parallel Method on MP211 Multicore Application Processor," *Proc. of the IEEE Symp. on Low-Power and High-Speed Chips (COOLChips VIII)*, pp.198-211, Apr. 2005.

[3] H. Inoue, et al., "FIDES: An Advanced Chip Multiprocessor Platform for Secure Next Generation Mobile Terminals," *Proc. of the IEEE/ACM/IFIP Intl. Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Sept. 2005.

Received July 29, 2005

* * * * * * * * * * * * * * * * *

Sunao TORII is a Principal Researcher at the System Devices Research Laboratories, NEC Corporation. He received ME degrees in computer sciences from Keio University 1992. He was a visiting researcher at the Univ. of Wisconsin from 2001 to 2002. He received Best Paper Award of the Journal of the IPSJ in 1999.

Mr. Torii is a member of both IEEE and IPSJ.

Junji SAKAI is an Assistant Manager of the System Devices Research Laboratories, NEC Corporation. In 1994, he joined a research project on chip multiprocessors at the NEC Central Research Laboratories. Since 2000, he has been involved in the development of parallel software for the MP211 multicore processor. He received his B.S and M.S degrees in information science from Kyoto University.

Mr. Sakai is a member of the IEEE Computer Society and IPSJ.

INOUE, Hiroaki received his B.S degree in physics and his M.E degree in computer science from Keio University in 1997 and 1999, respectively. He joined NEC Corporation in 1999 and is now an Assistant Manager of the System Devices Research Laboratories. He is engaged in the research and development of multiprocessor technologies.

Tatsuya TOKUE received his B.E degree in mechanical engineering from Yokohama National University in 1995. He joined NEC Corporation in 1995, and is now an Assistant Manager of the Mobile LSI Division of NEC Electronics Corporation.

Yoshiyuki ITO received his M.S degrees in information science from Tokyo Institute of Technology in 1991. He had been involved in the development of the MP211 processor at NEC Corporation, and is now a Manager of the 1st System Software Division of the NEC Electronics Corporation.

Mr. Ito is a member of IPSJ.

* * * * * * * * * * * * * * * * *