Prototype Microprocessor LSI with Scheduling Support Hardware for Operating System on Multiprocessor System

Naoki Nishimura Takahiro Sasaki Tetsuo Hironaka
Graduate School of Information Sciences,Hiroshima City University, Japan
3-4-1, Ozukahigashi, Asaminami-Ku, Hiroshima city, 731-3194, Japan
Tel:+81-82-830-1566, Fax:+81-82-830-1792
e-mail: {naoki,sasaki,hironaka}@csys.ce.hiroshima-cu.ac.jp

Abstract— In this paper we describe the details of SSH (Scheduling Support Hardware) which makes it possible to use fine grain parallelism effectively in multiprocessor systems. The SSH supports fast and concurrent thread scheduling that is very important in fine grain parallel processing.

I. INTRODUCTION

In recent years, multiprocessor systems are used in everyday usage. Most of these systems are based on the usage of coarse grain parallelism, with heavy requirement on program tuning for performance. Another method is fine grain parallelism, that the programs are decomposed into large numbers of very fine threads and executed on the multiprocessor system by dispatching threads to the idle processors dynamically. But the overhead seen on scheduling threads becomes large compared to the execution time of the threads. In order to reduce this problem, we introduce SSH(Scheduling Support Hardware) architecture which accelerates the performance of the OS with hardware, by scheduling threads/tasks concurrently with the thread execution on CPUs.

As a similar approach to this paper, reference[1] is proposed. Different from our proposition, this approach assumes to be implemented by FPGA, and scheduling policy is changed by reconfiguration the FPGA. Furthermore scheduling policy can be changed only once at the beginning. In our proposition, each thread can select and change its scheduling policy dynamically at runtime. Reference[2] is another similar approach, but this approach aims for real-time systems. Our approach aims for general multiprocessor systems and operating systems.

II. MULTIPROCESSOR SYSTEM WITH SSH

Figure 1 shows the architecture of the multiprocessor system with SSH. In our architecture, we assume shared memory system to make it easy to dispatch the thread to any arbitrary PE(Processing Element). This system is constructed by the following units, a number of PE's constructed by a CPU and `SSH-s`(SSH-slave), a `Shared Memory`, `Memory Bus Arbiter`, and `SSH-m`(SSH-master). CPU and `SSH-s` are assumed to be implemented on a single LSI, especially in our implementation CPU and `SSH-s` is tightly coupled by a 32-bit memory bus with a memory mapped I/O interface.

To meet the goals, SSH takes over the thread scheduling, usually done by operating system or by user program, using hardware scheduler. The main part of the thread scheduling takes place in the `SSH-m`. Data of the new created thread are transmitted to `SSH-m`

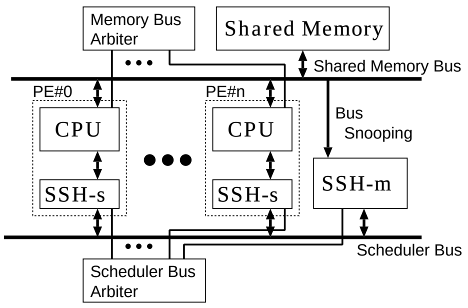

Fig. 1. Multiprocessor architecture with SSH.

and registered here for scheduling based on scheduling policy of each thread. When there is a request for dispatching thread from `SSH-s`, `SSH-m` returns the thread with the highest priority to the concerned `SSH-s`. As seen, thread scheduling by SSH is done completely concurrently with the execution of user program on CPU.

`SSH-s` and `SSH-m` are connected by the special bus called `Scheduler Bus`, which is used to keep all the memory bus bandwidth just for computing. In our implementation, the bus width of `Scheduler Bus` is set to 16-bit, based on the bandwidth needed by SSH. `Scheduler Bus Arbiter` is an arbiter to guarantee that the dispatch request from each `SSH-s` is handled fair. `Scheduler Bus Arbiter` gives the request from `SSH-m` always the highest priority, and the request priority from `SSH-s`'s is given by using the Round Robin algorithm, for arbitration. Details of `SSH-m` and `SSH-s` will be discussed in the next section.

III. STRUCTURE OF SSH

SSH is constructed by `SSH-m`, where the scheduling is done, and `SSH-s` which provides the interface to CPU and `SSH-m`. This section will discuss the detailed architecture of `SSH-m` and `SSH-s`.

A. SSH-m

`SSH-m` schedules threads, and sends the thread data with the highest priority to `SSH-s`. `SSH-m` is constructed by three units; `Bus Snooping Unit`, `Hardware Scheduler`, and `Global Queue`.

Functions of each unit are:

Bus Snooping Unit: This unit snoops `Shared Memory Bus` to check if there is any access to the semaphore variable. If the access to the semaphore variable is detected, `Bus Snooping Unit` reports it to `Hardware Scheduler`.

Hardware Scheduler: This unit schedules the thread for dispatching. The scheduling policy implemented here is FIFO, Round Robin, and Other as defined in pthread.

Global Queue: `Global Queue` contains the `Ready Queue` and the `Wait Queue` for each scheduling policy.

B. SSH-s

`SSH-s` works as an interface between CPU and `SSH-m`, and constructed by the following three units; `CPU-SSH Interface Unit`, `SSH-SSH Interface Unit`, and `Register Unit`.

Functions of each unit are:

CPU-SSH Interface Unit: This unit provides an interface for interaction with CPU. CPU and `SSH-s` communicate with each other via memory mapped I/O. This unit maps the internal register in the `SSH-s` to the memory address space.

SSH-SSH Interface Unit: This unit watches the `Read Queue` inside the `Register Unit`, when the `Read Queue` gets empty, request for the next thread to dispatch is sent out to `SSH-m` on demand. And also if a thread is newly created dynamically by CPU, it will be buffered in the `Write Queue` inside `Register Unit` for transmitting to `SSH-m`. The thread transmitted to `SSH-m` is registered in the suitable queue inside `Global Queue`.

Register Unit: This unit functions as a buffer for the thread downloaded from `SSH-m` and for the new thread to upload, which are registered in `Global Queue` inside `SSH-m`.

C. Performance of SSH

As SSH schedules the next thread to execute in parallel with the execution of the current thread by CPU, the minimum time needed for scheduling is the data transmission time between CPU and the `SSH-s`. The LSI we have implemented requires 20 cycles to look up the next thread to execute.

IV. LSI DESIGN

Detailed design of the multiprocessor system using SSH proposed in this paper was done using Verilog-HDL. And we had implemented the PE which was constructed by a CPU and `SSH-s`.

The LSI was designed as a PE of the multiprocessor system. The specifications of the PE are the following:

- Five stages 32-bit RISC CPU with the subset of `MIPS R3000` instruction set, without the multiply/ divide unit, because of the limitation of die size.
- For chip I/O, the LSI has one 32-bit `Memory Bus` that is time shared with address and data transfer, two 32-bit buses for instruction address and instruction fetching, and one 16-bit `Scheduler Bus` for SSH.
- LSI was designed to work at 32 MHz. 32MHz was chosen by the available print board design technology we could use.

TABLE I
SPEC OF THE PE CHIP.

Chip size	6.0 × 6.0mm
Cells of CPU	9,739
Gates^{*1} of CPU	18,706
Cells of SSH-s	954
Gates^{*1} of SSH-s	2,264
Technology	Hitachi 0.35μm
Max Clock Freq^{*2}.	51MHz

$*1$: in terms of NAND, $*2$: measured

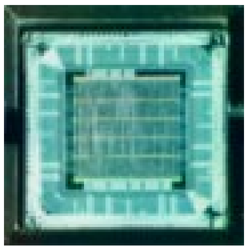
Fig. 2. The micrograph of the LSI designed.

The detailed specification of the LSI designed is listed in Table I. Max Clock Frequency is a measured number using the LSI tester, but since the limitation of Clock Frequency available from the LSI tester was 50MHz, the real Max Clock Frequency maybe more higher. Figure 2 shows the chip micrograph.

V. CONCLUSION

In this paper we described the details of PE LSI with SSH which make it possible to use fine grain parallelism effectively in multiprocessor systems. The SSH supports fast and concurrent thread scheduling that is very important in fine grain parallel processing. In the status of the project, we finished all of the logic designs of the multiprocessor system, and now we are implementing an environment to evaluate the LSI which we have designed.

ACKNOWLEDGEMENTS

The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration by Hitachi Ltd. and Dai Nippon Printing Corporation. We will like to thank Seiko Instruments Inc. for the support with VisualHDL.

REFERENCES

[1] M. Iida, M. Kuga, T. Sueyoshi: "On Chip Multiprocessor Using Multithread Control Library Implemented as Hardware," *Joint Symposium on Parallel Processing'97*, pp.337-344, 1997(Japanese).

[2] T. Nakano, Y. Komatsudaira, A. Shiomi, M. Imai: "Evaluation of a Real-Time OS Chip and Extension for a Distribution OS" *Technical report of IEICE. CPSY98-51*, pp.23-30, 1998(Japanese).