Evaluation of Single Chip Multiprocessor Core Architecture with Near Fine Grain Parallel Processing

Keiji Kimura and Hironori Kasahara

Dept. of Electrical, Electronics and Computer Engineering, Waseda University.
Okubo, Shinjuku-ku, Tokyo, Japan, 169-8555, TEL: +81-3-5286-3371
URL: http://www.kasahara.elec.waseda.ac.jp/

1 Introduction

Superscalar processors that have been widely used are facing their performance limitation with the increase of design time and cost. To cope with these problems, researches on single chip multiprocessor architectures (SCM) have been started [1, 2, 3, 4, 5, 6]. Among them, a SCM architecture supporting multigrain parallel processing, which exploits coarse grain task level, loop level and (near) fine grain level parallelism hierarchically, will be one of the most promising approaches to attain scalability and cost effectiveness. To examine the most suitable SCM processor core architecture for multigrain parallel processing, this paper evaluates several SCM processor core architectures having different instruction issue widths for near fine grain parallel processing which is one of the key issues in multigrain parallel processing.

2 Multigrain Parallel Processing

The multigrain parallel processing[7] realized by OSCAR Fortran compiler[8, 9] uses coarse grain task parallelism among loops, subroutines and basic blocks [8], loop parallelism among loop iterations and near fine grain parallelism[10] among statement inside a basic block.

2.1 Coarse-grain Task Parallel Processing [7, 8, 9]

In the coarse grain task parallel processing[7], a source sequential program is decomposed into three kinds of coarse grain tasks, or macrotasks (MTs), such as Block of Pseudo Assignment statements (BPA), Repetition Block (RB) and Subroutine Block (SB). After generation of macrotasks, the compiler analyzes control flows and data dependencies among MTs. The result of analysis is represented by a directed acyclic graph called Macro-Flow-Graph (MFG). Next, in order to find the maximum parallelism among MTs considering control dependences and data dependences, the compiler analyzes an earliest-executable-condition for each MT. These earliest-executable-conditions of MTs are represented by acyclic graph called MacroTask-Graph (MTG). After generation of MTG, MTs are assigned onto processor-groups (PGs).

2.2 Loop Iteration Level Parallel Processing

If a MT assigned to a PG is a Doall loop, the MT is processed in the iteration level grain by processing elements (PEs) inside a PG.

2.3 Near-fine Grain Parallel Processing [10]

If a MT assigned to a PG is a BPA or some kinds of sequential loops, it is decomposed into near fine grain tasks, each of which consists of a statement, and processed into parallel by PEs inside a PG. The compiler analyzes data dependences among statements and generates tasks. Next, OSCAR compiler assigned these tasks onto PEs statically to minimize data transfer and synchronization overhead. At this time, the compiler uses four heuristic scheduling algorithms, CP/DT/MISF, CP/ETF/MISF, ETF/CP and DT/CP, and chooses the best schedule automatically. After scheduling, the compiler generates the machine codes for each PE by putting together instructions for tasks assigned to the PE and by inserting instructions for data transfer and synchronization into the required places using statically scheduled result.

3 Evaluated Single Chip Multiprocessor Architecture

This section describes single chip multiprocessors (SCMs) and their processor cores evaluated in this paper. For this evaluation, clock level detailed simulator was developed.

3.1 Memory Architecture

Network and memory architecture for single chip multiprocessor (SCM) for near fine grain parallel processing evaluated in this paper is based on multiprocessor system OSCAR[9] as shown in Figure 1. In this SCM architecture, each processing element (PE) has CPU, local program memory (LPM), local data memory (LDM), distributed shared memory (DSM) having two ports and data transfer unit (DTU). DTU is

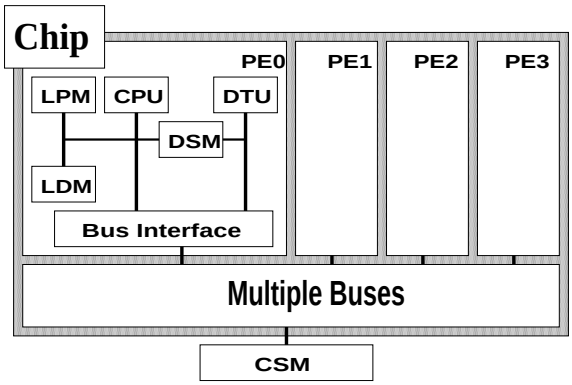

Figure 1: OSCAR type architecture.

used for overlapping of data transfer and computation by compiler control though this function is not used in this paper. Three buses connect these PEs in this evaluation through other interconnects like crossbar can be also used.

LPM stores program code exclusively generated to the PE by the compiler. This LPM supplies four instructions in one clock cycle. DSM provides low-latency data transfer and low-overhead synchronization in near fine grain parallel processing. DSM can be used for data transfer without preventing remote PE execution and minimize data synchronization overhead since "busy wait" for synchronization is performed inside a PE without consuming network and centralized shared memory band width. LDM stores PE local data and can have twice larger memory than DSM since local memory only has a single port. The total capacity of LDM inside a chip is 1M bytes. In case of 4PEs inside a chip, the capacity of LDM is 256K bytes per PE respectively. In addition, in case of 2PEs inside a chip, each PE has 512K bytes LDM. The access latency of LDM is one clock cycle. The capacity of DSM is 16K bytes per PE respectively. Local DSM access latency is one clock cycle and that of remote DSM is four clock cycles. In this SCM memory architecture, storing data and setting synchronization flag take four clocks respectively, loading data takes one clock cycle and checking synchronization flag takes five clock cycles.

Furthermore, this SCM processor has centralized shared memory (CSM) outside the chip. Access latency to the CSM is assumed as 20 clock cycles.

3.2 A Base Processor Core Architecture

A base processor core inside the chip has in-order-issue superscalar processor core that is based on UltraSPARC II[11] microprocessor. This processor core has two integer execution units (IEU), one load-store unit (LSU) and two floating-point units (FPU). In this evaluation, the number of function units and instruction issue width is changed as shown in Table 1 for single-issue, two-issues and for-issues processor core.

4 Performance Evaluation

In this section, SCM architecture having single- or multiple-instruction-issue-width processor core are evaluated. For the evaluation, the following workloads are used.

Random sparse matrix solution (Program S)
: This program is Fortran loop-free code that consists of 94 arithmetic assignment statements, or near fine grain tasks.

NS3D
: This program is a part of CFD program "NS3D" developed by National Aerospace Laboratory in Japan. It is a loop body of the largest loop inside a subroutine SUB4. This loop body has 429 near fine grain tasks.

FPPPP
: This program is a subroutine "FPPPP" of program "FPPPP" from SPECfp95 benchmark programs. This subroutine consumes about 35% of total execution time and has 333 near fine grain tasks.

These programs are processed in near fine grain parallel processing on target SCM architectures.

4.1 Effect of Issue Width and Number of PEs

Considering with total instruction issue width, three types of architectures are evaluated such as 1PE × 4Issue, 2PE × 2Issue and 4PE × 1issue. Figure 2 shows the speed-up ratio based on sequential execution time on single-issue processor core.

Figure 2 shows that 1Issue × 4PE gives us 2.84 times speed-up for random spars matrix solution, while 4Issue × 1PE gives us 1.15 times and 2Issue × 2PE gives us 2.12 times respectively. Similarly, 1Issue × 4PE gives us 2.18 times speed-up for NS3D, while 4Issue × 1PE gives us 1.15. For FPPPP, 1Issue × 4PE gives us 2.98 times speed-up, while 4Issue × 1PE gives us 1.24 times. For all cases, 1Issue × 4PE gives us the best performance of all three types of architectures. These results show that increasing the number of PEs contributes to scalable performance improvement more than increasing the number of instruction issue width.

Table 1: The number of function units

Issue-width	IEU	LSU	FPU
1	1	1	1
2	1	1	1
4	2	1	2

IEU: Integer Execution Unit
LSU: Load-Store Unit
FPU: Floating-Point Unit

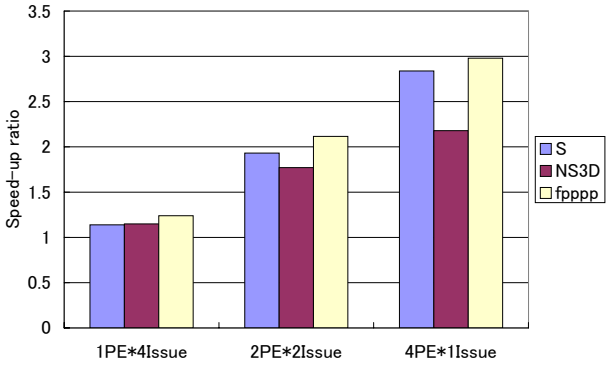

Figure 2: Multiple issues processor cores vs. single issue processor core SCM

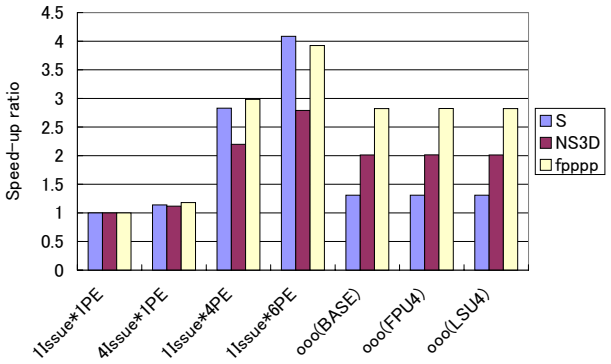

Figure 3: Simple processor core SCM vs. out-of-order processors

4.2 Simple processor core SCM vs. Out-of-order Processor

Next, performances of 1Issue × 4PE SCM and out-of-order superscalar processor are evaluated. The function unit organization of evaluated out-of-order processor is similar to Compaq Alpha21264[12]. This processor has four integer execution units (IEU), two load-store units (LSU) and two floating-point units (FPU), and can issue up to six instructions in one cycle. In addition to this out-of-order processor (ooo(BASE)), two types of extended out-of-order processors like four FPU (ooo(FPU4)) and four LSU (ooo(LSU4)) are also evaluated.

Figure 3 shows the speed-up ratio against sequential execution time by single-issue processor core. In this figure, 1Issue × 4PE gives us a little better performance than ooo(BASE), ooo(FPU4) and ooo(LSU4). In addition, SCM architecture gives us about 1.4 times performance improvement by adding two PEs (namely 1Issue × 6PE) requiring comparable number of transistors as ooo(FPU4) and ooo(LSU4). This shows that SCM architecture achieves scalable performance improvement by adding PEs.

5 Conclusions

This paper evaluated single chip multiprocessor (SCM) processor core architectures for near fine grain parallel processing as the preliminary research for SCM architecture that supports multigrain parallel processing. The evaluation shows that the SCM architecture having four single-issue processor cores gives us 2.49 times better performance than a four-issue in-order superscalar processor. Furthermore, the simple processor core SCM was compared with complex six issue out-of-order processor. As a result, the simple SCM architecture having four single issue PEs gives us a little better performance than the complex out-of-order processor. In addition, SCM architecture gives us about 1.4 times performance improvement by adding two PEs though the 1 Issue × 6PE only consumes less transistors than complex six issue out-of-order processor. These results show that SCM architecture having simple processor core gives us scalable performance improvement by near fine grain parallel processing.

ACKNOWLEDGEMENTS

A part of this research has been supported by STARC. The authors thank to Mr. Ozawa (STARC), Mr. Hirata (STARC), Mr. Asano (Toshiba), Mr. Kurata (Sony), Mr. Takahashi (Fujitsu) and Mr. Takayama (Matsushita).

References

[1] Sun Microsystems, Inc. *MAJC Home Page*, 2000. http://www.sun.com/ microelectronics/ MAJC/.

[2] K. Diefendorff. Power4 focuses on memory bandwidth. *Microprocessor Report*, 13(13), 1999.

[3] L. A. Barroso, K. Gharachorloo, et al. Piranha: A scalable architecture based on single-chip multiprocessing. In *Proc. ISCA 00*, June 2000.

[4] L. Hammond, B. Hubbert, M. Siu, M. Chen, and K. Olukotun. The Stanford HYDRA CMP. *IEEE MICRO Magazine*, 20(2):71–84, 2000.

[5] NEC Corporation. *MP98 Project*, 2000. http://www.labs.nec.co.jp/ MP98/.

[6] R. Barua, W. Lee, S. Amarasinghe, and A. Agarwal. Maps:A Compiler-Managed Memory System for Raw Machines. In *Proc. of ISCA-26*, June 1999.

[7] H. Kasahara, H. Honda, and S. Narita. A multigrain parallelizing compilation scheme for oscar. In *Proc.4th Workshop on Lang. and Compilers for Parallel Computing*, Aug 1991.

[8] H. Kasahara, M. Obata, and K. Ishizaka. Automatic coarse grain task parallel processing on smp using openmp. In *Proc. of 13th International Workshop on Language and Compilers for Parallel computing (LCPC'00)*, August 2000.

[9] H. Kasahara, M. Okamoto, et al. OSCAR Multi-grain Architecture and Its Evaluation. In *Proc. IWIA'97*. IEEE computer Press, Oct 1997.

[10] H. Kasahara, H. Honda, and S. Narita. Parallel processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR (Optimally Scheduled Advanced Multiprocessor). In *Proc. of Supercomputing '90*, Nov 1990.

[11] Sun Microelectronics. *UltraSPARC^{TM} User's Manual*, Jul. 1997.

[12] R. E. Kessler. The alpha 21264 microprocessor. *IEEE MICRO Magazine*, 19(2):24–36, 1999.