A CUSTOM PROCESSOR FOR THE MULTIPROCESSOR SYSTEM ASCA

Takashi Fujiwara* Katsuto Sakamoto** Takahiro Kawaguchi* Keisuke Iwai*
Hideharu Amano*
* Department of Computer Science, Keio University
3-14-1, Hiyoshi Yokohama 223-8522 Japan.
** Hitachi, Ltd. Software Development Center
5030, Totsukacho, Totsukaku, Yokohama, 223 Japan.
{fujiwara,sakamoto,kawaguti,iwai,hunga}@aa.cs.keio.ac.jp

Abstract

In this paper, a custom microprocessor MAPLE (Multiprocessor system ASCA Processing eLEment) which is the constituent element for the multiprocessor system ASCA is proposed and designed.

MAPLE core and the barrier unit have been designed with Verilog HDL. From the synthesis result of the module using $0.4\mu m$ CMOS, maximum frequency of MAPLE core works over 50MHz and the amount of gates is approximately 185,000.

KEYWORDS
"multiprocessor", "microprocessor","multi-grain paralleizing scheme"

1 INTRODUCTION

Although it is easy to enhance the peak performance of the multiprocessor, it is difficult to exploit effective performance for users without the support of the automatic parallelizing compiler and scheduler. However, parallelizing software including compilers and schedulers have been tailored for existing multiprocessors, and it prevents an epoch-making performance improvement.

The multiprocessor system ASCA (Advanced Scheduling oriented Computer Architecture) is proposed based on the idea that not the parallelizing software is tailored for machines, but the multiprocessor system should be designed to make the best use of parallelizing software.

ASCA adopts a multi-grain parallelizing compilation scheme[1] which can exploit parallelism of the user program at various levels: coarse-grain parallelism, medium-grain parallelism and near-fine-grain parallelism. MAPLE is designed mainly for exploiting near-fine-grain parallelism. Since near-fine-grain parallelizing compilation requires a precise static scheduling among operations, uncertain behavior of MAPLE must be completely excluded.

First, MAPLE has a simple structure for supporting compiler. MAPLE implements an extended instruction set architecture of DLX for multiprocessor system ASCA. Five-stage pipeline structure of MAPLE is designed to execute every operation in a fixed number of clocks. Complicated enhancement techniques which introduce uncertain factors such as dynamic hardware branch prediction and out-of-order issue are excluded.

Second, a software controlled cache scheme is used to eliminate the uncertainty factor of hit/miss-hit which occurs on a general hardware controlled cache scheme. In this scheme, load and replacement of lines are mainly controlled by the software, and the cache always hits except under special cases. Using this scheme and an intelligent automatic compiler, MAPLE requires few synchronizations in a macro task at the near-fine-grain parallelizing scheme.

Finally, special operations of ASCA related to data transfer and barrier are designed. MAPLE can transfer data between registers across the network and synchronize at high speed among processing elements in the processing cluster.

In this paper, a dedicated custom processor MAPLE (Multiprocessor system ASCA Processing eLEment) is proposed as an element processor of ASCA. In the section 2, multi-grain parallelizing compilation scheme and ASCA are introduced. In the section 3, the architecture of MAPLE is proposed. In the section 4, the initial evaluation results are presented.

2 MULTIPROCESSOR SYSTEM ASCA

A multi-grain parallelizing compilation scheme[1] is hopeful scheme to exploit various level of parallelism in a program: coarse-grain parallelism (macro-data flow computation), medium-grain parallelism (loop-level parallelism) which is used in most of the current compilers, and near-fine-grain parallelism (statement level parallelism).

Although this scheme is useful for various multiprocessors, the maximum performance will be obtained on a dedicated multiprocessor architecture whose processor, memory system and interconnection network are tailored for the scheme. For example, if the behavior of a multiprocessor is completely predictable, the compiler can perfectly schedule at compile-time. It can also remove explicit synchronization for near-fine-grain parallel processing in the basic block. For this purpose, we propose a multiprocessor system ASCA(Advanced Scheduling oriented Computer Architecture) for efficient execution of multi-grain parallelizing scheme.

Figure 1 shows the conceptual structure of ASCA.

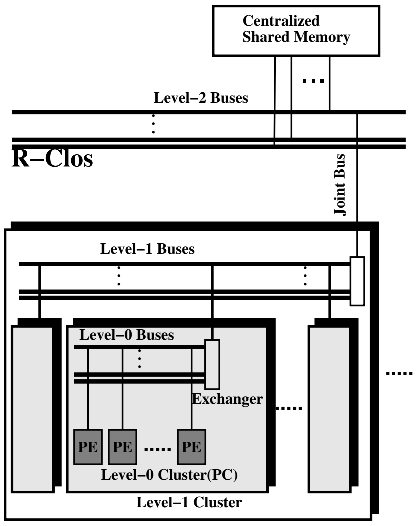

Figure 1: MULTIPROCESSOR SYSTEM ASCA

ASCA consists of processing clusters (PCs) and a centralized shared memory (CSM) to store shared variables. PC includes four or eight processing elements (PEs), and connected with a novel interconnection network called Recursive Clos (R-Clos). R-Clos is a hierarchically extended Clos network with extra intermediate stages, and can emulate hierarchical multi-bus model easily. The compiler can predict packet conflicts at compile-time and optimize communication by static scheduling using R-Clos network.

PE consists of a custom microprocessor, distributed shared memory (DSM), local memory(LM) and controllers for them(Figure 2).

3 CUSTOM MICROPROCESSOR MAPLE

For near-fine-grain parallel processing in the multi-grain parallelizing compilation scheme, the behavior of a processor must be perfectly predicted, and scheduled at compile-time. However, it is difficult in the current processors run-time uncertain factors, such as latencies depending on the operation and data size, dynamic hardware branch prediction and speculative execution. Moreover, in most processors, pipeline interlock caused by the cache miss hit prevents the static scheduling. Since the near-fine-grain parallelizing compilation requires a precise static scheduling between statements, uncertain behaviors mentioned above must

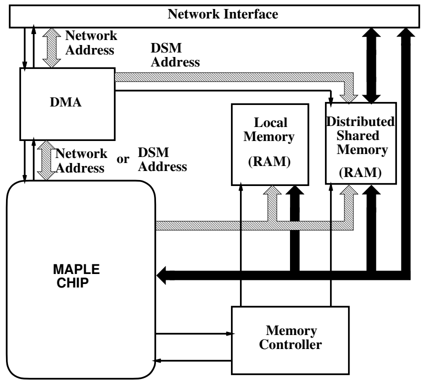

Figure 2: PE BLOCK DIAGRAM

be completely excluded. To address this problem, we propose and design a custom microprocessor MAPLE (Multiprocessor system ASCA Processing eLEment).

3.1 MAPLE

As shown in Figure 3, MAPLE consists of three units: core unit, barrier unit and cache unit.

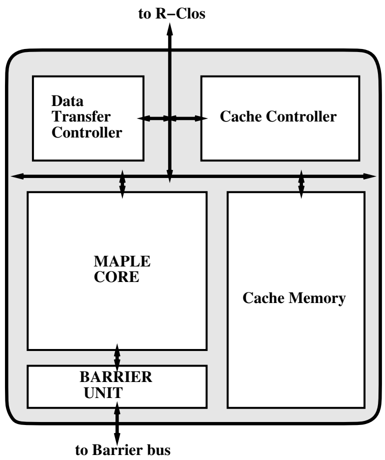

Figure 3: MAPLE BLOCK DIAGRAM

3.1.1 MAPLE CORE

MAPLE core is a 32-bit RISC processor which provides a simple structure with highly predictable operations. Its instruction set is an extension of that of DLX[2]. Five-stage pipeline structure of MAPLE is designed to execute every operation in a fixed number of clocks. Pipeline provides data forwarding considering out-of-order execution and completion which be predictable. A floating-point execution unit includes 32-bit IEEE std 754-1985 floating-point unit. There are 32 integer registers and 32 floating point registers. Furthermore 16 (in 8 PEs/PC) 32-bit receive registers for data transfer between PEs are provided. The primary on-chip cache is 32KByte.

MAPLE provides special instructions for multiprocessor system ASCA. The quick data transfer operations and synchronization operations.

The fast data transfer operations are classified into two types: between DSMs operations and between registers operations. When MAPLE executes a transfer operation between DSMs, data transfer command is issued to the DMA controller. DMA controller transfers data on the DSM of source PE to the DSM of destination PE independently from the core operation. The destination PE checks the flag on the local DSM whether data transfer is completed. In this type, the DMA can exploit a high bandwidth of the network independently from the core processor, while it takes a certain time for setting up. Thus, it is suitable for a large data transfer.

On the other hand, operation between registers are suitable for one word transfer used in the near-fine-grain parallelizing. In these operations, a special register called the receive register is used. The register is directly connected to MAPLE pipeline ID stage, and when the source processor executes a transfer operation between registers, data is directly sent out from the MEM stage of the pipeline. The transferred data is also directly received by a receive register of destination processor. It is necessary to detect the flag on receive register, when the destination processor wants to use transferred data. If the flag is not set, the pipeline of receiving processor stalls. As soon as the flag is set, the pipeline stall is cancelled and receiving processor can use the transferred data in the receive register. Figure 4 illustrates the data transfer between registers.

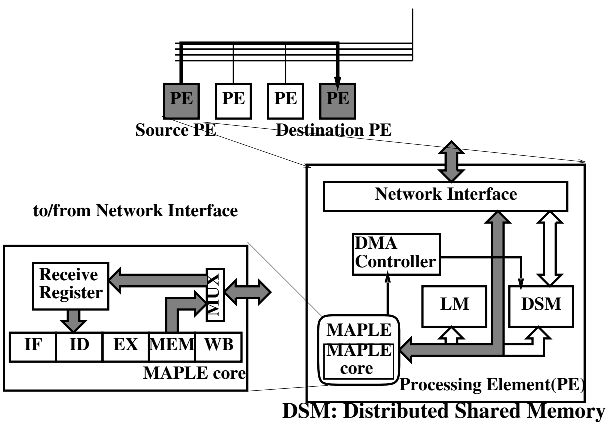

Figure 4: DATA TRANSFER BETWEEN REGISTERS ACROSS THE NETWORK

Since the receive register is embedded within the chip, the size is restricted. In the first prototype, the size is only one word. However, since most of data transfer size between PEs in near-fine-grain parallelizing is one word, the performance degradation caused by this restriction is not so large.

3.1.2 BARRIER UNIT

MAPLE provides barrier operations for quick synchronizations. For flexible synchronization, a group barrier between PEs in a PC is implemented. Figure 5 shows the structure of barrier unit.

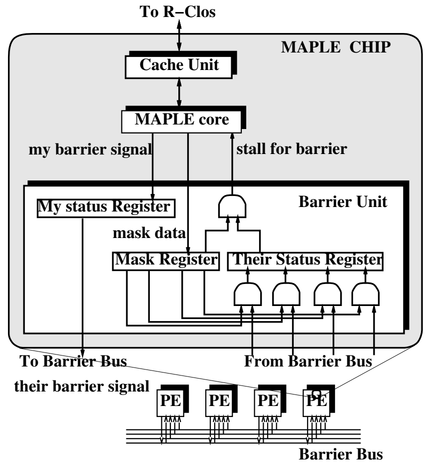

Figure 5: BARRIER UNIT

When MAPLE core executes a group barrier operation, first the core sets the mask register in barrier unit to indicate which PEs will be synchronized. Then the core signals 1-bit barrier information to barrier bus, and the pipeline stalls. The barrier unit latches barrier information of other PEs in the status register. The unit compares the mask register with the status register, and if they are matched, the MAPLE pipeline resumes operations. Since the barrier unit is implemented inside the chip, barrier operation completes with a cycle overhead.

3.1.3 CACHE UNIT

MAPLE cache system adopts a large software cache which excludes the uncertain factor on the hit/miss-hit of the hardware cache. In this cache, the loading and replacing of the line are mainly controlled by the software, and the cache always hits except special cases. As shown in Figure 6, a software cache is managed both by its dedicated controller and an intelligent controller called DTC(Data Transfer Controller). DTC provides 3-stage pipeline and executes its own instruction generated at the compile-time. The instruction has information of when and where the data must be transferred. DTC executes DTC instructions before the MAPLE core requests the data. If DTC instruction is data transfer operation, it sends a request to the software cache controller. Although the loading and replacing data are triggered by an DTC instruction, the operation itself is executed in the software cache controller. Once triggered, the software cache controller manages the data transfers between memory(DSM,CSM,LM) and cache independently of the DTC.

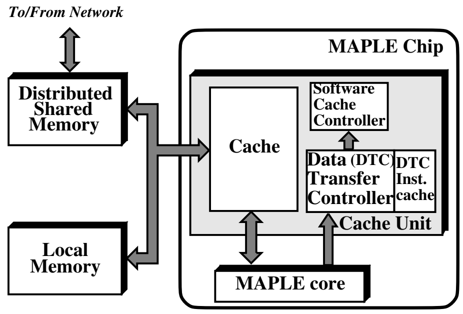

Figure 6: SOFTWARE CONTROL CACHE

Using this cache, synchronization operations are not required in most tasks. Finally the multiprocessor executes a near-fine-grain parallelizing scheme without synchronization code in a macro task [3].

However, all macro tasks can not be perfectly scheduled data transfer at compile-time. For this reason, MAPLE provides two cache control modes. One is software controlled cache mode mentioned above, and the other is hardware controlled cache mode. The hardware controlled cache mode uses the general hardware for cache control. If the data movements run off the predicted behavior, it is detected by the software cache controller and informed to the MAPLE core. In this case, MAPLE changes the mode of cache into the hardware controlled mode. The mode alternation triggers a write back operation for maintain consistence between data in cache and memory. After that, the hardware controlled cache causes cold miss hit like starting up. The mode can be changed explicitly by the software with executing a mode change instruction.

4 EVALUATION

MAPLE core and barrier unit have been designed with Verilog-HDL. The cache unit is currently under development.

From the synthesis result of the module using Toshiba ASIC TC203G library($0.4\mu m$ CMOS), maximum frequency of MAPLE core is more than 50MHz and the amount of gates is approximately 185,000. A total gates of MAPLE is estimated to be approximately 400,000 (32KByte on-chip cache).

With the assumption that MAPLE core runs at 50MHz, we simulated the core with FFT program on Mentor Graphics Corporation Quick HDL. We assume that load from on-chip cache (32KByte) requires extra 1 cycle. Figure 7 shows the execution time of FFT with 2^{10}. FFT program is compiled with no option using GCC. MAPLE core is of little difference in execution time from the early superscalar processor(HP-7100, SuperSPARC, HyperSPARC), because the design of MAPLE floating-point unit lays more stress on speed than the amount of gates.

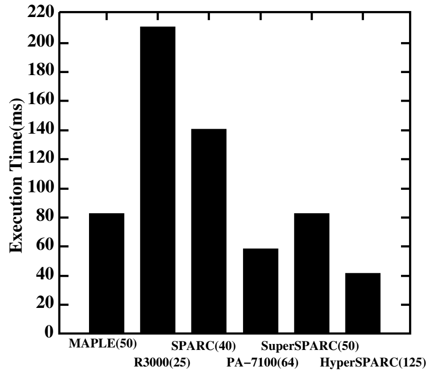

Figure 7: FFT EXECUTION TIME

5 CONCLUSION

In this paper, we proposed and designed a custom microprocessor MAPLE for Multiprocessor system ASCA. The MAPLE, except for the cache unit, has been designed, and the cache unit is now under development. MAPLE core and barrier unit approximately requires 185,000 gates. From the synthesis result of MAPLE core, it runs over 50MHz. The total amount of gates is approximately 400,000.

ASCA equipped with MAPLE make the best use of the multi-grain parallelizing compilation scheme.

6 ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Mentor Graphics Japan for supporting design tools.

References

[1] Kasahara H., Honda H., and Narita S.,A Multi-Grain Parallelizing Compilation Scheme for OSCAR, *Proc.4th Workshop on Languages and Compilers for Parallel Computing*, 1991

[2] John L. Hennessy, and David A. Patterson, *COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH SECOND EDITION*, Morgan Kaufmann Publishers, 1996

[3] Ogata W., Fujimoto K., Oota M. and Kasahara H., Compilation Scheme for Near Fine Grain Parallel Processing on a Multiprocessor System without Explicit Synchronization, *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, May, 1995, 327-332