A Floating Point Arithmetic Unit for a Static Scheduling and Compiler Oriented Multiprocessor System ASCA

Takahiro Kawaguchi Takayuki Suzuki Hideharu Amano

Keio University
Dept. of Computer Science
3-14-1, Hiyoshi Yokohama, 223-8522 Japan
Tel: +81-45-560-1063
Fax: +81-45-560-1064
e-mail: {kawaguti, takayuki, hunga}@am.ics.keio.ac.jp

Abstract— We describe a floating point arithmetic unit(FPU) which supports static scheduling by automatic parallelizing compiler. This FPU designed to work with 50MHz clock with the assistance of EDA synthesis and layout tools. Under the clock rate condition, it appears that this FPU requires about 120,000 gates and marks 8.2 MFLOPS with the clock level simulation.

I. Introduction

Although it is easy to enhance the **peak** performance of the multiprocessor only by adding many processing units, it is difficult to exploit **effective** performance for users without support of automatic parallelizing compilers. However, such compilers have been tailored for existing multiprocessors which are designed without care of them well. The multiprocessor system ASCA(Advanced Scheduling oriented Computer Architecture) has been proposed based on the idea that not the parallelizing software is tailored for machines, but a multiprocessor system should be designed to make the best use of parallelizing software. The goal of ASCA system is to exploit parallelism of the user program in various levels of granularity by scheduling at compile time with automatic multigrain parallelizing compiler and scheduler[1], and the processor MAPLE(Multiprocessor system ASCA Processing eLEment) is designed to support them over-all[2]. The pipeline structure and instruction set architecture of MAPLE are similar to those of DLX[3], a simple but powerful RISC model processor.

II. MAPLE FPU

A. Features

The key function of MAPLE is a high performance floating point arithmetic unit since application field of ASCA system mainly targets on scientific calculations. The FPU conforms to the IEEE 754 floating point binary arithmetic standard[4], and provides instructions of the four fundamental rules of arithmetic in double precision, floating point comparison, and format conversion between integer and floating point number.

The FP execution stage is further divided into seven pipelined functional units. Each functional unit is fully pipelined except for divide instructions(figure 1).

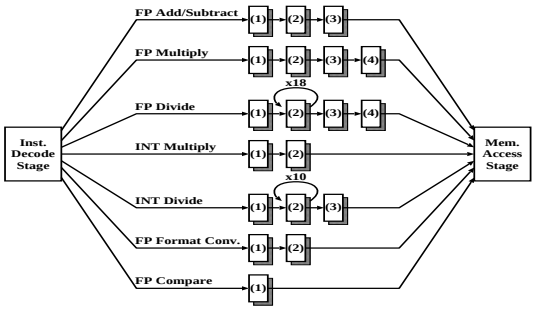

Fig. 1. Floating Point Execution Stage

Table I shows the required cycle time to complete each operation at the FP-EX stage. Latency and throughput for single precision data are the same as those for the double precision.

TABLE I
FP execution stage specification

Operation	Algorithm	†1	†2
FP add/sub.	-	3	1
FP mult.	Radix-4 Booth encode, Wallace tree compress	4	1
FP div.	Radix-2 SRT	21	18
FP compare	-	1	1
Format conv.	-	2	1
INT mult.	Radix-4 Booth encode, Wallace tree compress	2	1
INT div.	Radix-2 SRT	12	10

†1: Latency [clock cycle], †2: Throughput [clock cycle]

Generally, floating point arithmetic instructions take a long time to complete compared with integer instructions. A common FPU integrated in an off-the-shelf general-purpose processor returns the result as quick as possible according to input values. For example, when executing 0.0×0.0, such FPUs return the result in a cycle. However, this method is not desirable for ASCA system, since

a compiler cannot predict the behavior of the processor. That is, the MAPLE FPU should complete every instruction in a fixed cycle time.

However, such approach often degrades the performance. MAPLE FPU is designed not to degrading it as possible. In order to achieve high performance scientific calculations even with such a restriction, the RTL design of the FPU is highly timing aggressive. Figure 2 shows an example of block diagram of floating point add module. Two 53bit adders are parallelized for reducing the latency. Moreover, the disadvantage is also recovered by the ASCA parallelizing compiler using static scheduling method which can make the best use of operation level parallel processing.

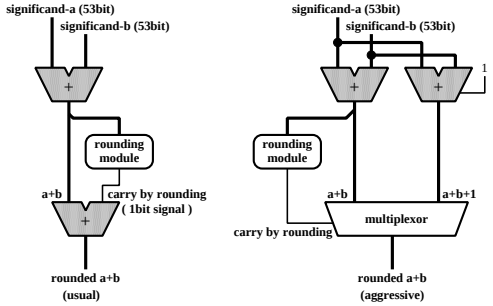

Fig. 2. Basic Pattern of Aggressive RTL Design

B. Performance

The performance of the MAPLE FPU with 50MHz clock is evaluated with Mentor Graphics' Quick HDL simulator. Figure 3 shows the reciprocal of FFT execution time and performance (MFLOPS) of 'FLOPS' benchmark program. The FFT processes double precision 2^{20} data and 'FLOPS' executes numerical integration and Maclaurin series expansion in double precision.

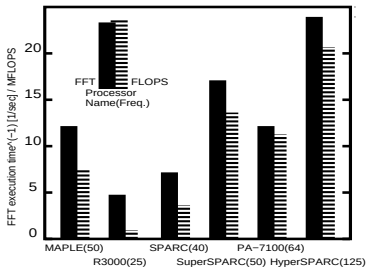

Fig. 3. Evaluation of FFT and FLOPS Program

III. Implementation

The MAPLE FPU chip in this study has been fabricated with the program of VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration by Rohm Corporation and Toppan Printing Corporation.

The synthesis result of the module using the Rohm technology(table II), shows that the hardware amount of FPU module requires about 120,000 gates(table III).

TABLE II
The Specification of MAPLE FPU

Technology	Die size	Package
CMOS 0.6μm Std.Cell Metal 3, Poly 2, Vdd=5V	79.39mm^2	QFP 208pin

TABLE III
Amount of gates of MAPLE FPU

Block	#of cell	NAND-gate eqv.
Fp add/sub.	9,222	17,438
Fp mult.	21,728	41,086
Fp div.	10,184	19,258
Int mult.	6,484	12,261
Int div.	5,087	9,620
Rest of FPU	12,717	24,048

FPU total: 65,422 cells (123,712 gates)

Figure 4 shows the FPU cell layout and photograph.

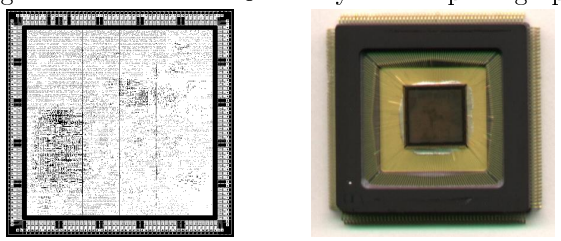

Fig. 4. The MAPLE FPU Cell Layout and Chip

IV. Conclusions

We describe that the MAPLE FPU supports automatic parallelizing compiler for effective performance. Although the performance is lower than those of recent superscalar processors, the gate number of MAPLE FPU is so small that the cost/performance of the MAPLE cluster has possibility to overcome that of recent high performance processors in scientific calculations. The FPU chip will be verified and evaluated with the logic tester system (Schlumberger ITS 9000 EXa) installed in VDEC.

Acknowledgements

We would like to express our sincere gratitude to Mentor Graphics Japan for providing design tools in the university program(HEP). This study is supported by STARC (Semiconductor Technology Academic Research Center, Japan) as the project "A processing element for static scheduling and compiler oriented multiprocessor."

References

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

[2] T.Fujiwara, K.Sakamoto, T.Kawaguchi, K.Iwai, H.Kasahara, "A Custom Processor for the Multiprocessor System ASCA",*Applied Informatics '98*, 1998, pp258-261.

[3] J.L.Hennessy and D.A. Patterson, *COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH SECOND EDITION*, Morgan Kaufmann Publishers, 1996.

[4] The Institute of Electrical and Electronics Engineers Inc., *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754-1985, 1985.