A Performance Improvement for Floating-Point Arithmetic Unit with Precision Degradation Detection

Soseki Aniya Toshiaki Kitamura

Graduate School of Information Sciences
Hiroshima City University
3-4-1 Ozuka Higashi, Asaminami-ku, Hiroshima, Japan, 731-3194
aniya@csl.info.hiroshima-cu.ac.jp kitamura@hiroshima-cu.ac.jp

Abstract— Some errors are very important in the scientific computation observed in floating-point calculations caused by rounding, overflow, underflow, loss of significant digits, or loss of trailing digits. In the prior work, we designed a vector co-processor that has floating-point arithmetic units with detection of loss of significant digits and precision degradation. We propose a partitioned vector co-processor design. The design can improve performance of the data transfer throughput between vector co-processor and SSRAM. Compared to the prior work, the number of execution cycles of the vector load instruction becomes twice faster in the RTL simulation.

I. Introduction

Recently, the computation scale is increasing by the cutting-edge high performance computing. As in large-scale scientific computation, some errors may be observed in floating-point calculations caused by rounding, overflow, underflow, loss of significant digits, or loss of trailing digits. IEEE754[1] handles the problems by signaling the exception of rounding, overflow and underflow. However, the calculated result is insufficient by precision degradation due to cannot detect loss of significant digits and loss of trailing digits.

The prior work[2] attacked above problems but performance was insufficient by two factors. First, the data transfer throughput between vector co-processor and SSRAM is limited. Second, the divide operation latency is too slow due to the slow Restoring Division algorithm.

For this study, we were improved the data transfer throughput by the partitioned vector co-processor. We used the FPGA board that connected to the host PC via PCI-Express (Fig.1). The FPGA board is composed of two SSRAMs and two FPGAs, and each FPGA has a partitioned vector co-processor. These partitioned vector co-processors can access to each other resource via External Local Bus.

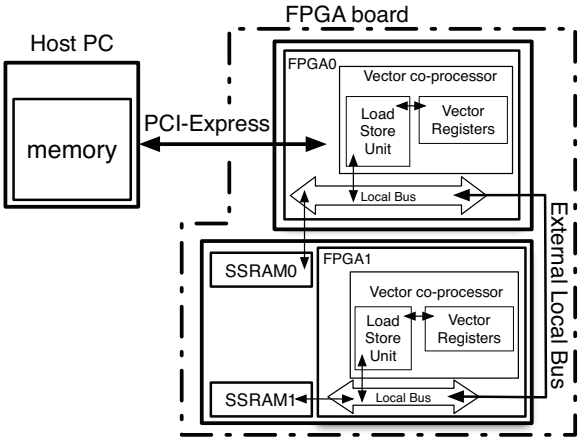

Fig. 1. Structure of the FPGA board

II. Related Work

In the prior work[2], we designed an architecture that has floating-point arithmetic unit with detection of loss of significant digits and precision degradation. This architecture has been implemented on the FPGA board that connected with the host PC via PCI-Express. In order to detect precision degradation for large-scale scientific computation, the architecture is implemented like vector processor structure (Fig.2) that has many floating-point arithmetic units. The vector co-processor core is composed of 16 vector registers and double precision floating-point arithmetic units; 8 adders, 4 multipliers, and 4 dividers. Each interleaved vector register with 8 banks has 512 elements of 64-bit width. Each 1-read 1-write bank connects to each floating-point arithmetic unit. In addition, the hardware design provides quadruple precision floating-point arithmetic consisting of dual double precision floating-point arithmetic units (Fig.3). The implementation has the problem that uses only one of two FPGAs, for simplicity.

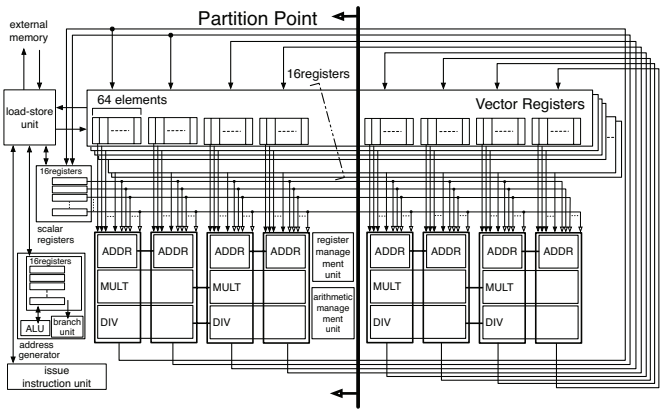

Fig. 2. Block diagram of the vector co-processor

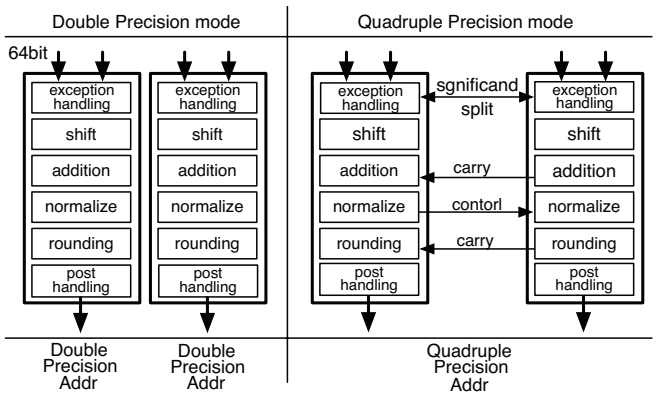

Fig. 3. Double-Quadruple Floating-Point Adder unit

III. Hardware Design

A partitioned vector co-processor on the left of the Partition Point (Fig.2) is composed of 16 vector registers, load-store unit, instruction issue unit, scalar register file, register management unit, arithmetic management unit and double precision floating-point arithmetic units; 4 adders, 2 multipliers, and 2 dividers. Each interleaved vector register with 4 banks has 256 elements of 64-bit width.

However, some instructions require the synchronization between two partitioned vector co-processors. For example, one operation, like a vector sum, is executed adding up partial sum on each partitioned vector co-processor independently but two partial sum results must be added beyond a partitioned vector co-processor. We solve this problem by using the high-speed serial communication of the Xilinx RocketIO.

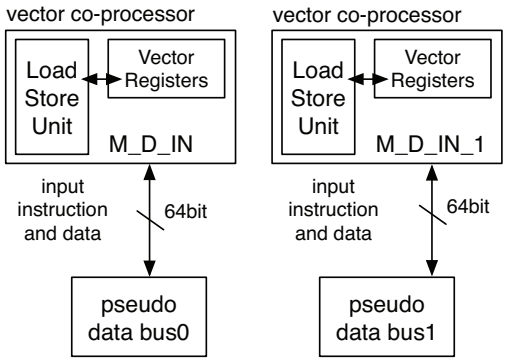

Fig. 4. Overview of the sub-system

IV. Evaluation Environment

We created the sub-system for the evaluation (Fig.4). The sub-system is composed of two partitioned vector co-processors and two pseudo data buses, and we input same data and instruction for two partitioned vector co-processors. We evaluate the number of execution cycles of the vector load instruction for 512 elements, by simulating RTL in Mentor ModelSim.

V. Summary and Conclusions

We evaluated the vector load instruction for 512 elements, on the sub-system. As the result, the number of execution cycles has been decreased to 256 cycles from 512 cycles. This result shows that the data transfer throughput is improved, and it contributes to an improvement on the performance of the vector co-processor. A synthesis result for a partitioned vector co-processor implemented on the Xilinx VIRTEX-5 (XC5VLX330T) occupies 36846 slice registers and 70541 slice LUTs, and clock frequency is 50MHz.

Acknowledgements

This work is partly supported by Grant-in-Aid for Scientific Research(C) No.22500051. This work is also partly supported by VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Mentor Graphics, Inc.

References

[1] Microprocessor Standards Committee of the IEEE Computer Society. "IEEE Standard for Floating-Point Arithmetic" IEEE Stndard 754, August 2008.

[2] Keita Kaneko and Toshiaki Kitamura. "Evaluation of floating-point arithmetic unit with precision degradation detection". *2011-ARC-193*, Vol. 16, pp. 1–6, January 2011.