Software-Controlled On-Chip Memory for High-Performance and Low-Power Computing

Masaaki KONDO Motonobu FUJITA Hiroshi NAKAMURA

Research Center for Advanced Science and Technology, The University of Tokyo

E-mail: `[kondo, mfujita, nakamura]@hal.rcast.u-tokyo.ac.jp`

1 Motivation

The performance gap between processor and main memory speed, called memory wall, is serious problem especially in High Performance Computing (HPC). This memory wall problem is addressed by two factors, large memory access latency and lack of memory throughput. There have been proposed many techniques for tolerating memory access latency, including cache prefetching. However, these techniques increase main memory traffic[1].

In order to overcome this problem, we have proposed a new processor architecture SCIMA (the abbreviation of Software Controlled Integrated Memory Architecture) which integrates software controllable memory (SCM) into a processor chip[2]. Our previous work revealed that SCIMA has the potential to tolerate memory latency without wasting memory bandwidth. Currently, we are working on the strategy how to control SCM by software. In this presentation, we present the optimization methodology and show its effectiveness.

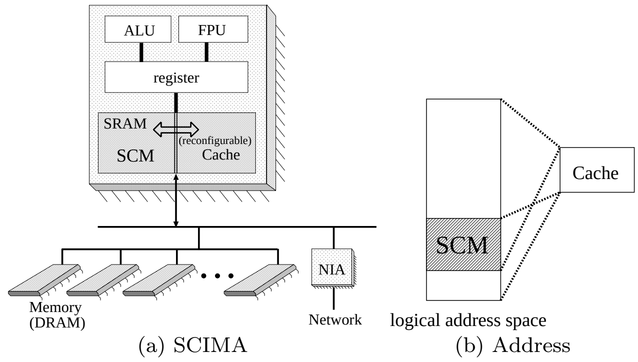


Figure 1. Overview of SCIMA

Table 1. Effects on Execution Time

On-Chip Memory Features	T_b	T_l	T_t
software controllability	-	-	↓
burst data transfer	↑	↓	-
stride data transfer	↑	↓	↓
Latency Tolerating Techniques of Cache	T_b	T_l	T_t
larger cache line	-	↓	↑
lock-up free cache	-	↓	↑
cache prefetching	↑	↓	↑

2 SCIMA

Figure 1-(a) shows the schematic view of the proposed SCIMA. Addressable SCM is integrated into a processor chip in addition to ordinary cache. SCM occupies one consecutive and uncacheable part of logical address space as shown in Figure 1-(b).

Data transfer among memory hierarchy Two kinds of data accesses are available in SCIMA.
1) register ↔ SCM ↔ off-chip memory
2) register ↔ cache ↔ off-chip memory

Data transfers between SCM and off-chip memory are invoked explicitly by new instructions. These instructions can identify large amount of data transfer, which reduces the number of off-chip memory accesses for consecutive data access.

These instructions also support block-stride data transfer which packs non-consecutive data of off-chip Memory and transfers into a consecutive area of SCM. This is helpful for reducing the required SCM area and off-chip bandwidth.

Reconfiguration of SCM and cache SCM and cache are reconfigured at the granularity of ways. This mechanism is helpful because the best ratio of cache and SCM sizes highly depends on applications. The size of SCM can be 0. Thus, SCIMA includes the ordinary cache architecture.

Benefit of SCM Table 1 presents how the characteristics of SCIMA and latency tolerating techniques of conventional cache affect the processor's execution time. In the Table, T_b, T_l, and T_t represent CPU-busy time, stall time caused by main memory latency, and stall time caused by lack of main memory throughput respectively. Note that SCIMA can reduce both T_l and T_t whereas cache techniques increases T_t.

3 Strategy of Software Control

Arrays are classified based on their access characteristics as shown in Table 2. Figure 2 describes the strategies for making good use of SCM depending on the classification Table 2 of each array.

Among the strategies, (1) and (2) are more effective than the others. Transferring such non-reusable arrays with large granularity is very helpful for reduc-

Table 2. Classification of Arrays

Category	Classification
reusability	reusable / not-reusable
consecutiveness	consecutive / stride / irregular

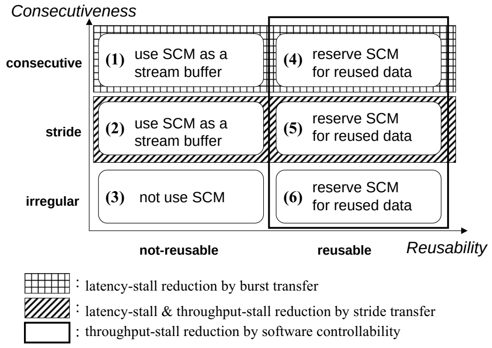


Figure 2. Strategy for Using SCM

ing latency-stall. Therefore, we give higher priority to (1) and (2). After they are applied, strategies of (4), (5), and (6) are applied to reusable arrays by using the rest area of SCM.

4 Preliminary Evaluation

We used two programs for evaluation. One is kernel FT from NAS Parallel Benchmarks and the other is QCD (Quantum ChromoDynamics) computation[3].

We compared the performance of SCIMA with cache-based architecture. High-level optimization such as loop tiling and loop exchange are applied to both SCIMA and Cache respectively by hand. Here, optimization for SCIMA follows the strategy of Section 3.

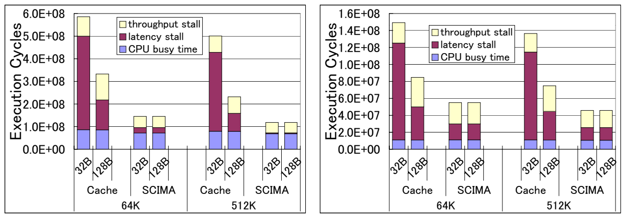
The following is the assumptions.

- number of execution units: integer=2, floating-point (multiply-add)=1
- total on-chip memory size: 64KB or 512KB
- cache line size: 32B or 128B
- throughput of cache/On-Chip Memory = 8B/cycle
- throughput of Off-Chip Memory: 1B/cycle
- Off-Chip Memory access latency: 160 cycle
- *page* size: 4KB
- Cache: prefetching cache (Cache model only)

Evaluation Result Figure 3 presents the required cycles for each program. The cycles are decomposed into three categories, CPU busy time, latency stall, and throughput stall which correspond to T_b, T_l, and T_t in section 2 respectively.

In the graph, the left side shows the case of 64KB on-chip memory whereas the right side represents that of 512KB. In each part, performances of Cache and SCIMA are presented for two kinds of cache lines (32B and 128B).

As seen from Figure 3, SCIMA achieves better performance than Cache for both the programs under all the configuration. This is because SCIMA can successfully and drastically reduce latency-stall time with reducing



(a) FT (b) QCD

Figure 3. Performance Result

throughput-stall time. Compared with Cache, for example, SCIMA reduces 32% of the off-chip traffic in FT and 34% in QCD under the assumption of 512KB on-chip memory and 128B cache line. This reduction of the off-chip traffic is beneficial for saving power consumption because off-chip access is the major factor of power consumption.

These results reveal that proposed optimization strategy is very effective for making good use of SCM and SCIMA is superior to cache based architecture.

5 Work in Progress

In this paper, optimization strategy for SCM usage is presented. Though it is quite primitive, the evaluation results show the effectiveness of the strategy.

We are designing an adaptive compiler for SCIMA toward an automatic optimization adopting the proposed optimization strategy. It is important to design more detailed algorithms, for example, determining the best configurations of cache and SCM, and detecting the best loop transformation. We are planning to evaluate SCIMA on a wider variety of applications optimized by the developed compiler.

Saving power consumption is also our objective. We expect power dissipation decreases because the SCIMA can reduce off-chip memory traffic and the number of data transfer. Evaluating power consumption is also our current work.

Acknowledgment This work is partially supported by the Research for the Future Program of JSPS in the "Computational Science and Engineering" Field (Project No. JSPS-RFTF 97P01102), and by CREST of JST (Japan Science and Technology Corporation).

References

[1] D. Burger, J. Goodman, and A. Kagi, "Memory Bandwidth Limitation of Future Microprocessor", Proc. ISCA-23, pp.78–89, 1996

[2] M. Kondo, H. Okawara, H.Nakamura, and T. Boku, "SCIMA: Software Controlled Integrated Memory Architecture for High Performance Computing", Proc. ICCD-2000, pp.105–111, Sep. 2000

[3] S. Aoki, R. Burkhalter, K. Kanaya, T. Yoshié, T. Boku, H. Nakamura, and Y. Yamashita, "Performance of lattice QCD programs on CP-PACS", Parallel Computing 25, pp.1243–1255, 1999.