Shared vs. Snoop: Evaluation of Cache Structure for Single-chip Multiprocessors

Toru Kisuki†,Masaki Wakabayashi†,Junji Yamamoto†,Keisuke Inoue†,
Hideharu Amano†

†Department of Computer Science, Keio University
3-14-1, Hiyoshi Yokohama 223 Japan
{kisuki,masaki,junji,keisuke,hunga}@aa.cs.keio.ac.jp

Abstract. The shared cache structures and snoop cache structures for single-chip multiprocessors are evaluated and compared using an instruction level simulator. Simulation results show that 1-port large shared cache achieves the best performance if there is no delay penalty for arbitration and accessing the bus. However, if 1-clock delay is assumed for accessing the shared cache, a snoop cache with internal wide bus and invalidate style NewKeio protocol overcomes shared caches.

1 Introduction

In order to make the best use of a large silicon area, single-chip microprocessor approach using simple microprocessors has been investigated[1][2]. The precise simulation results demonstrated that this approach is hopeful compared with complicated superscaler processors for parallel applications[1].

In these evaluations, processors are connected with a shared cache which is suitable for on-chip implementation. However, a private cache with snoop mechanism which is mainly used in board level implementation is also hopeful structure for on-chip implementation, since each cache can be connected with a high bandwidth on-chip bus. The snoop cache protocol can be optimized for on-chip implementation. In this paper, the structure of snoop cache for single-chip multiprocessor is discussed and the performance is compared with shared cache using an instruction level simulation.

2 Cache structures for Single-Chip Multiprocessors

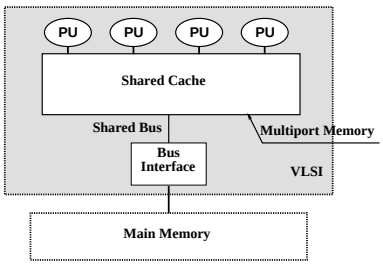

Fig. 1. Shared Cache Structure

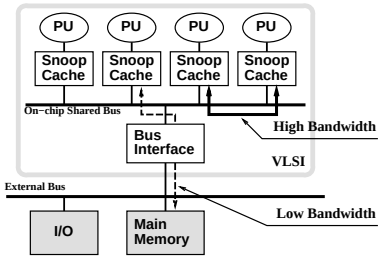

Fig. 2. Snoop Cache Structure

2.1 Shared cache structure In many possible connection architectures for single-chip multiprocessors, the most simple structure is connecting processors to shared cache directly. Fig.1 shows the most simple and quite realistic structure of the single-chip multiprocessor. In this structure, a large SRAM shared cache is also connected to main memory outside the chip. Due to the overhead caused by the conflict, multi-port memory which allows simultaneous access is sometimes introduced. But the access time is often stretched by a large fan-out required in cells of m-port memory.

2.2 Snoop cache structure Private cache with snoop mechanism is an alternative approach to efficient cache for single-chip multiprocessors. In this approach, each processor provides its private cache connected with each other by a shared bus. Each cache checks the address and data on the shared bus and maintains the consistency according to the cache consistency protocol. This structure is commonly used in the recent multiprocessor workstations. In the board level implementation, since the common 64 bits address/data multiplexed bus is used, the cost for maintaining cache consistency often dominates the performance.

However, in the single-chip multiprocessor, the cost of wire is much less than that in the board-level implementation, and a bus with a large bandwidth can be used. Here, we propose the snoop cache with a wide bus for single-chip multiprocessors. In this cache, the following bus is used: (1) the size of the data bus is the same as the cache line, and (2) 64-bit address bus is provided independent from the data bus. In such a snoop cache, a cache line can be transferred between caches only with a clock, and overhead for maintaining the consistency is drastically reduced.

2.3 NewKeio protocol In a snoop cache with a wide bus, the gap between inside and outside of the chip is an essential problem. In order to cope with this problem, we have proposed a coherent protocol which minimizes the data transfer between cache and main memory.

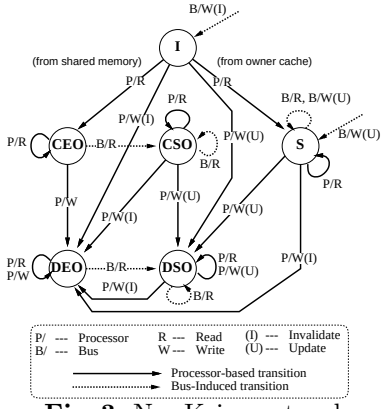

Fig. 3. NewKeio protocol

This protocol is called NewKeio Protocol[3]. As shown in Fig.3, the ownership is introduced even for a clean line, and the miss-hit cache line is transferred from cache as possible. Moreover, the attribute is attached to the page table to select whether the protocol uses invalidation or updating.

By using the invalidation protocol for instruction or local data, and the update protocol for shared data, the loss caused by consistency is minimized. In Fig.3, (U),(I) indicates Update and Invalidate type, respectively.

3 Simulation Environment

3.1 Target systems By the end of 1997, it is possible to make a processor chip which has $500mm^2$ die area in $0.25\mu m$ process technology. Since a sophisticated processors requires a large area, the possible number to be implement becomes small. Therefore, there are many combinations between the class and number of processors. Here, we select rather simple R3000 class RISC processor. In this case, from four to six processors and 256KB SRAM can be mounted. 500MHz system clock is assumed, and a latency accessing the outside main memory becomes 50 clocks.

Here, we evaluate the following cache structures:

- 4-Port Shared Cache
  In this cache, 4-port memory is used. Each processor has its own port and accesses cache memory without any conflicts. However, the total cache size is set to be a forth (64KB) of the 1-Port Shared Cache.
- 1-Port Shared Cache
  A single-port memory is used for this cache. Since each processor shares a port, a conflict occurs when multiple processors access the port simultaneously. However, its simple structure allows a large cache size (256KB).
- Snoop Cache(Illinois protocol)
  A common cache protocol (Illinois protocol[4]) is used as snoop cache protocol. Each processor has its own 64Kbyte cache, so total cache size is 256KB.
- Snoop Cache(NewKeio protocol)
  The same structure as the snoop cache with Illinois protocol, but NewKeio protocol is used.

3.2 Instruction-level simulator ISIS Since the performance of cache is depending on the detail behavior of hardware including pipeline structure, a precise simulation is required.

For this purpose, we developed an instruction level simulator called ISIS. ISIS simulates the behavior of the RISC processor pipeline stage for every instruction. Precise operation of each pipeline stage including delay slots can be simulated.

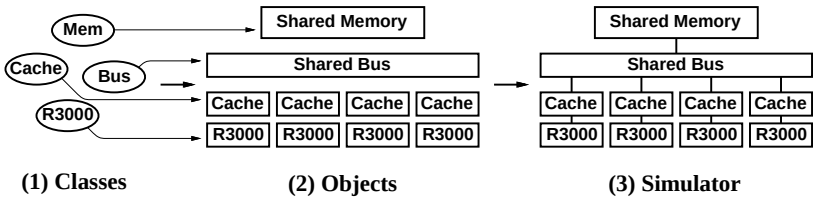

Fig. 4. Organization of ISIS

Fig.4 shows the organization of ISIS. All units such as processor, cache, bus and memory are implemented as classes. At first, each class is constructed, then each object is connected through an interface. In this way, various architectures can be simulated by replacing units.

Three parallel applications from SPLASH benchmark programs[5] are selected for evaluation.

- FFT(data points is set to be 4096)
- MP3D(100 steps with 8000 particles at test.geom)
- LU(128×128 matrix with a block size 16)

4 Simulation Results

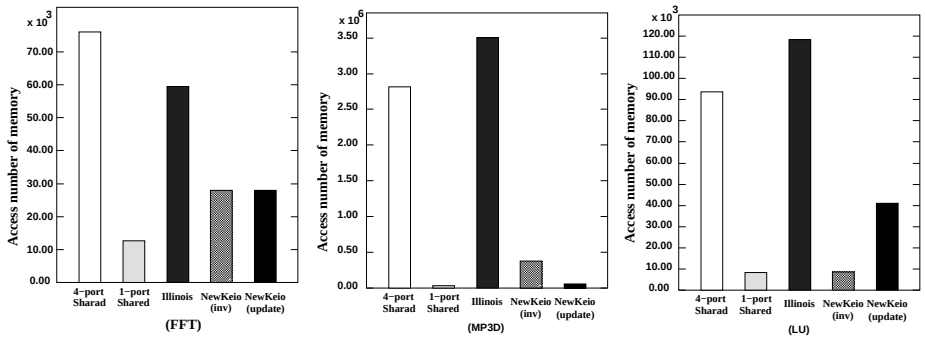

Fig. 5. Access Number of Memory

4.1 Access frequency of the outside memory Fig.5 shows the number of accessing outside memory. Since there is a large gap between the bandwidth of inside and outside chip, a large number of access to the main memory causes performance degradation.

In FFT, 4-port shared cache shows most frequent memory access by the capacity miss. Illinois protocol is better than 4-port shared cache, since cache to cache inside the chip is provided for access miss of a clean line. NewKeio protocol shows further better performance than Illinois protocol. Unlike Illinois protocol which requires write back when the dirty line is required from other processor, the write back occurs only at replacing in NewKeio protocol. This causes the difference of the access number, especially in MP3D which requires a large amount of interprocessor communication. Due to this reason, the update type protocol is advantageous in MP3D.

4.2 Execution Time Considering the multi-port and the arbitration delay, it takes a few more clocks to access the shared cache. We assume this delay by penalizing additional clocks to shared cache (1-3 clocks). In order to investigate the influence of the conflicts which occur in 1-port shared cache, the result of ideal execution time without conflicts is also shown in Fig.6.

Without delay, the performance of 1-port shared cache is the best in all applications. In 1-port shared cache, the influence of the conflict is not so large if no delay is assumed. As mentioned above, the performance of 4-port shared cache is degraded mainly by the communication with outside memory.

However, with the delay penalty, the performance of 1-port shared cache is severely degraded because of the conflict. Even with 1-clock delay, invalidate

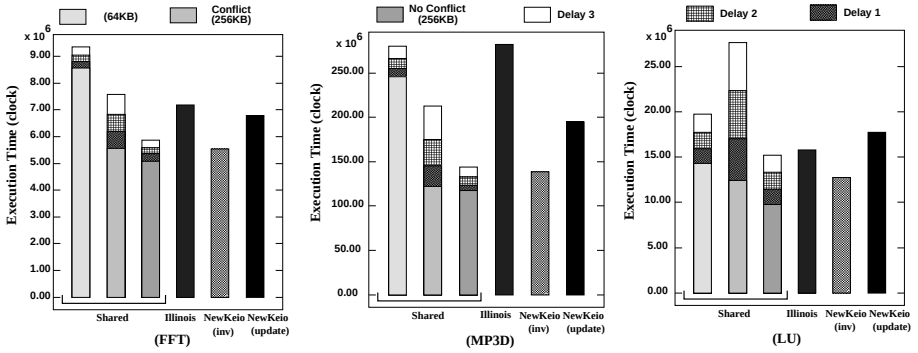

Fig. 6. Execution Time

type NewKeio protocol shows better performance than 1-port shared cache. Since there are a few shared line and interprocessor communication, each private cache can be used efficiently and reduce the overhead to maintain the consistency. From this reason, the performance of invalidate type NewKeio protocol is better than that of update type. Compared with Illinois protocol, NewKeio protocol improves the performance in 15%-20%.

5 Conclusion

The simulation results show that with the parameters used here, 1-port large shared cache achieves the best performance if there is no delay penalty for arbitration and accessing the bus. However, if 1-clock delay is assumed for accessing the shared cache, a snoop cache with internal wide bus and invalidate style NewKeio protocol shows better performance. Further simulations with various parameters and applications are required for investigating optimized cache structure for single-chip multiprocessors.

References

1. Basem A. Nayfeh, Lance Hammond, and Kunle Olukotun. Evaluation of Design Alternatives for a Multiprocessor Microprocessor. *ISCA*, 1995.
2. Marco Fillo, Stephen W. Keckler, William J. Dally, Nicholas P. Carter, Andrew Chang, Yevgeny Gurevish, and Whay S. Lee. The M-Machine Multicompuer. 1995.
3. T. Terasawa, S. Ogura, K. Inoue, and H. Amano. A Cache Coherence Protocol for Multiprocessor Chip. *In proc. of IEEE International Conference on Wafer Scale Integration*, pages 238–247, January 1995.
4. M.S.Papamarcos and J.H.Patel. A Low-overhead Coherence Solution for Multiprocessors with Private Cache Memoryes. *ISCSA84*, pages 348–354, 1992.
5. Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. *ISCA*, pages 24–36, June 1995.

This article was processed using the LaTeX macro package with LLNCS style