Utilization of Cache Area in On-Chip Multiprocessor *

Hitoshi Oi[1] and N. Ranganathan[2]

[1] University of South Florida, Dept. of Computer Science and Engineering, Tampa, FL 33620. oi@csee.usf.edu

[2] Dept. of Electrical and Computer Engineering, The University of Texas at El Paso, El Paso, Texas 79968. ranganat@ece.utep.edu

Abstract. On-chip multiprocessor can be an alternative to wide-issue superscalar processor to exploit increasing number of transistors on a silicon chip. Utilization of cache has more performance impact due to its higher penalty for remote (off-chip) communication than board-level implementation. We examine two options for better utilizing cache resource: (1) private data is only cached at L1 and L2 is used only for shared data, (2) dividing cache area into L2 and remote victim cache or just a large L2 cache. Results of execution-driven simulations showed that the first option improved the performance up to 10%. For the second option, four out of six benchmark programs showed that large L2 is more effective than the combination of L2 and remote victim cache.

1 Introduction

Superscalar processor has been the mainstream of exploiting increasing number of transistors on a silicon chip. However, the limitation of this approach has been reported for mainly two reasons [1]. First, the increase of die area on the chip is not justified by the performance gain since most applications have only limited instruction level parallelism (ILP). Second, the circuit complexity of wider issue superscalar lowers clock speed.

An alternative approach is to implement a multiprocessor on a single chip [1]. In this approach, each processor is moderately superscalar and hence it operates at a very high clock rate. Also, processors can exploit fast on-chip communication for parallel applications. Yet another approach to exploit a large number of transistors is to integrate main memory and processor on a chip [2]. With this approach, memory latency is significantly reduced due to the wider on-chip datapath and the lower signal propagation delay.

When several on-chip multiprocessors are connected by an interconnection network, the ratio of local-remote access latency is expected to be higher than that of traditional board-level implementation. In this paper, we examine two design options to reduce remote misses using limited cache resource of on-chip multiprocessor. Due to on-chip integration, local memory access speed becomes

* Supported in part by a National Science Foundation Grant No. MIPS 9522265

close to that of L2. The first option is to use L2 cache only for shared data, and single level caching (using only L1) for private data. Higher locality of private data access and faster on-chip memory make this option feasible. Another option is whether or not to use remote victim cache (cache that only accommodates remote blocks replaced from L2 cache), when the total cache area is limited.

This paper is organized as follows. In the next section, the architecture of on-chip multiprocessor is described. In Section 3, our methodology for performance evaluation is presented. Performance evaluation of the on-chip multiprocessor and the effectiveness of the design options are given in Section 4. Related work is introduced in Section 5. Some conclusions are provided in Section 6.

2 On-Chip Multiprocessor Architecture

We use Olukotun and others architecture [1] as a starting point. The semiconductor technology they assumed was that of 1997. When manufactured in the near future from now, it is expected that more transistors are available on the silicon chip than Olukotun assumed. We assume that this extra die area to be used for accommodating main memory on the same chip. The access time of on-chip memory is significantly faster because it can avoid I/O delays and its datapath width is not restricted by the pin count of the chip.

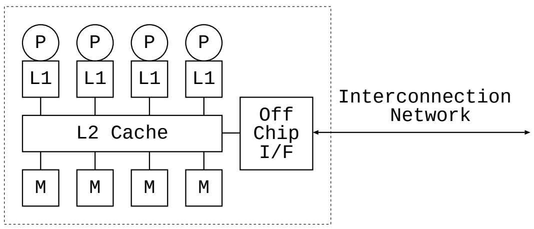

Fig. 1. On-Chip Multiprocessor Architecture

Fig. 1 shows the architecture of on-chip multiprocessor. There are small number of processors that are moderately superscalar (For example, there were four 2-issue processors in the estimation of [1]). Each processor has own instruction and data L1 cache. The size of L1 cache is small and it has low degree of set-associativity for fast access speed. The processors on the same chip share an L2 cache that has a larger size, higher degree of set-associativity and hence slower access speed. On-chip main memory is divided into several units to avoid contention, and each memory unit is associated with directory entries representing copies of memory blocks. There is an off-chip communication interface to an interconnection network to other chips and/or to an external memory unit.

3 Simulation Environment

For performance evaluation, we have developed an execution-driven simulator using Augmint multiprocessor toolkit [3]. Each chip has four processors as in [1]. We consider a bus-connected medium size configuration of 32 processors (8 chips). Each on-chip multiprocessor has the configuration in Fig. 1 with the following assumptions: All the instruction fetches are L1 hit. The size of on-chip memory is large enough to store all the instructions and data.

L1 cache is direct-mapped while L2 cache is 4-way set-associative. We use a split transaction bus to connect multiple chips that are separated for request (read request, invalidate request and acknowledge) and data. An invalidation request is broadcast to all the chips via request bus, and each chip having the copy of the block will send acknowledgment back individually. The on-chip memory unit is four-way interleaved. Contention at shared resources (L2 cache, memory, off-chip interface, request/data bus) are modeled.

The system parameters that will be used in performance evaluation in Section 4 are shown in Table 1. We use relatively small size of L2 cache for two reasons. First, we use the default problem sizes of SPLASH2 suites which were decided to run simulations in reasonable time [4], but practical applications are likely to have larger problem sizes, and hence will be spilt out from the caches [5]. Second, as stated above, we assume all the instruction fetches are L1 hit. In reality, however, instruction words occupy L2 cache, and they conflict with data blocks. The clock speed of bus is five times slower than the processor. Since the width of data bus is 16Bytes, it takes four bus cycles (20 processor clock cycles) to transfer a L2 cache block (64Bytes).

L1 Cache	Direct Mapped Size: 8KB/Proc	Latency: 1 Block Size: 32B
L2 Cache	4-way Set Assoc Size: 32KB/Chip	Latency/Cycle Time: 8/4 Block Size: 64B
Memory	4-way Interleaved Latency/Cycle Time:	 8/4 (Directory) 30/40 (Data)
Off-Chip I/F	Latency/Cycle Time: 10/5	
Bus	Cycle: 5	Data Bus Width: 16B

Table 1. System Parameters. Timing Parameters Are Represented By Processor Clock Cycles

We use a set of parallel applications from SPLASH 2 [4] with their default problem sizes (Table 2). The allocation of shared data to memory unit is as follows. When a memory location is accessed by a processor for the first time, the location is assumed to be allocated in one of the memory units in the same chip as the processor. Within a chip, one of memory units is chosen in a round-robin manner.

Application	Description	Problem Size
FFT	1-D fast Fourier transformation	64K complex
Radix	Radix sort	256K keys
Barnes	Interaction of N-bodies in 3-D space	16K bodies
Ocean	Multigrid solver of ocean movement	258×258 grid
Volrend	Rendering of 3-D volume	Head
Water-N^2	Forces and potentials in water molecules	512 molecules

Table 2. Benchmark Programs and their Problem Sizes

4 Performance Evaluation

In this section, we evaluate performance of the on-chip multiprocessor together with design options for performance improvement.

4.1 Properties of Base Configuration

The speed up of the on-chip multiprocessor is shown in Fig. 2. Cache hit ratio profile of simulation results for 8 chips (32 processors) is shown in Table 3. As we have chosen small L2 size, its hit ratio is relatively low. The L2 hit ratio for shared data of radix increased for larger configurations. This is because the L2 cache size was too small for radix's working set. As the number of chips increased, working set was divided and it began to fit into L2 cache. However, it still incurred a lot of inter-chip communication, especially by writebacks of dirty remote blocks. In radix more than 20% of cache block replacements caused remote writeback, while in barnes and water it was around 1.5% and in fft, ocean and volrend, it was much less than 1%. This large amount of inter-chip communication caused data bus saturations, and resulted in higher growth rate of access latency. Fft and water showed similar speed up by 4-chip configuration. However, fewer bus traffic of water allowed it a speed up of 92% for 4 to 8-chip while that of fft was 77%. In ocean, only 4% of L2 misses were remote. In water, it had high L2 hit ratio for shared data and it increased for larger configuration. These factors resulted in good speed up of ocean and water. Relatively low speed up of volrend was mainly due to synchronization.

In the following subsections, we consider design options to improve performance of the on-chip multiprocessor.

4.2 Single-Level Caching of Private Data Access

It is noticed from Table 3 that hit ratio of private data access is quite high ($>$ 98%). This higher hit rate of private access than shared access leads to an idea

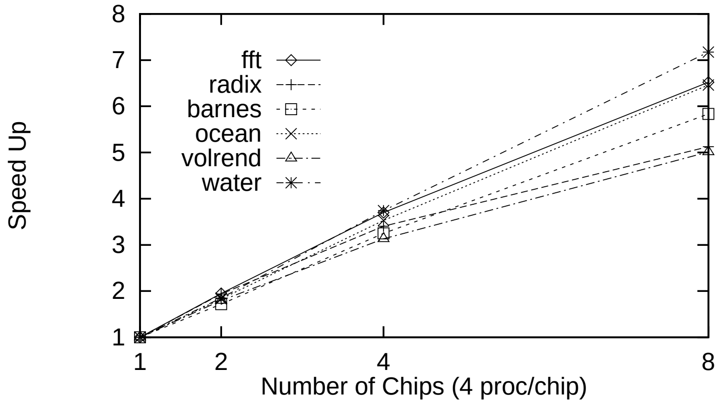

Fig. 2. Speed Up of On-Chip Multiprocessor

Benchmark Program		fft	radix	barnes	ocean	volrend	water
L1	Overall	94.5	94.2	95.1	96.2	96.8	96.9
Hit	(Private)	98.0	99.4	98.9	99.1	98.4	98.2
Ratio	(Shared)	90.6	71.1	69.4	81.5	75.9	93.6
L2	Overall	57.0	55.7	55.05	49.4	42.3	62.4
Hit	(Private)	48.7	45.3	50.9	53.1	38.1	50.4
Ratio	(Shared)	59.0	56.7	56.1	48.4	45.9	71.7
L2	Private	22.7	10.1	21.3	18.4	49.6	57.8
Miss	Local	65.4	68.3	22.1	77.5	11.8	18.4
Location	Remote	11.9	21.6	56.6	4.1	38.6	23.8

Table 3. Cache Hit Profile of Benchmark Programs (8 chips/32 processors)

of using L2 cache exclusively for shared accesses. The advantage of this scheme is higher L2 hit rate for shared access, while the disadvantage is higher L1 miss penalty for private access. Low access speed ratio between L2 cache and on-chip memory supports this scheme to work.

We examin four cache/memory configurations. *Base* configuration (both private and shared data occupy L2) is used as a base of comparison. Other configurations are, single-level caching of private data (*SLCPD*), separation of private/shared memory units (*Sep Mem*), and combination of SLCPD and Sep Mem. Separation of memory units for private and shared data is to see whether or not SLCPD increases memory busy rate too much.

Fig. 3 shows the simulation result of 32-processor configuration. Among six benchmark programs, ocean, volrend, and water-N^2 showed relatively large benefit of SLCPD. Their performances were improved by 6 to 10%. Since with SLCPD, private data do not occupy L2 cache, large fraction of replacements at L2 cache could be avoided. For example, in the case of water-N^2, the number of replacements was reduced to 26% of base configuration. Accordingly, the L2 hit ratio of shared data has increased from 71.7% to 78.7% and average access latency was reduced from 1.98 to 1.63 clock cycles. Except ocean, memory separation itself did not improve the performance, and even when combined with SLCPD, we could not see difference too. The performance of fft was improved slightly with SLCPD. Although SLCPD itself did not increase the busy rate of memory unit (actually it was lowered by SLCPD), it was more effective when combined with memory separation. Combination of memory separation and SLCPD improved fft's performance by 5%, while SLCPD alone did by 2%. In radix and volrend, despite L2's being used only for shared data, the improvement of L2 hit ratio of shared data was small (from 56.1% to 57.7% for barnes, and from 56.7% to 58.3% for radix). Thus, the improvement of overall performance was also small.

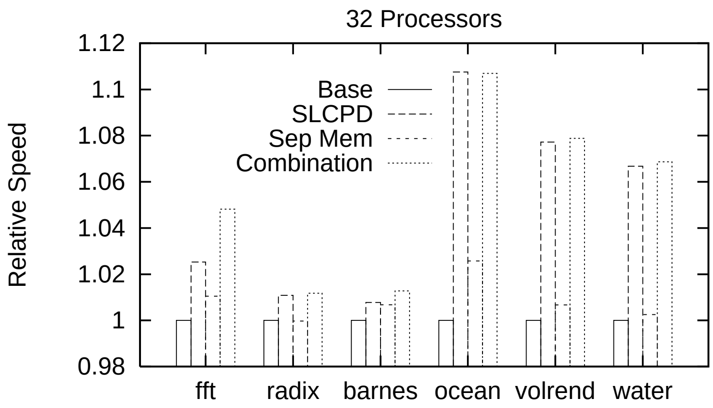

Fig. 3. Single-Level Caching of Private Data (8 Chips)

4.3 Effectiveness of Remote Victim Cache

In CC-NUMA multiprocessors, the latency of remote data access is significantly higher than that of local data. One way to alleviate the latency of remote access is to add a remote cache to the memory hierarchy [5]. Moga and Dubois proposed an augmentation to remote cache, remote victim cache: a cache memory that only stores remote blocks replaced out from L2 cache [6].

In these past studies, board-level implementation of nodes/clusters of CC-NUMA multiprocessor was assumed. Thus, addition of remote cache was mainly a matter of cost. However, in the case of on-chip implementation, total amount of cache (L2 and remote victim caches) is restricted by the die area. A question that arises in this situation is whether we should divide die area into L2 and remote victim cache or just use it for a large L2 cache.

We examine five cache confugrations: 32:0, 32:8, 32:16, 32:32 and 64:0, where $X : Y$ stands for L2 cache of size XKB/chip and remote victim cache of size YKB/chip, respectively. The total cache area used should be in the order of $32 : 0 < 32 : 8 < 32 : 16 < 64 : 0 < 32 : 32$. This is because 32:32 consumes more area for cache tag and datapath than 64:0. If 32:16 (or 32:8) outperformed 64:0, we could say remote victim cache can better utilize a limited cache area, vice versa. We assume remote victim cache is 4-way set associative and its access hit takes four clock cycles in addition to L2 latency.

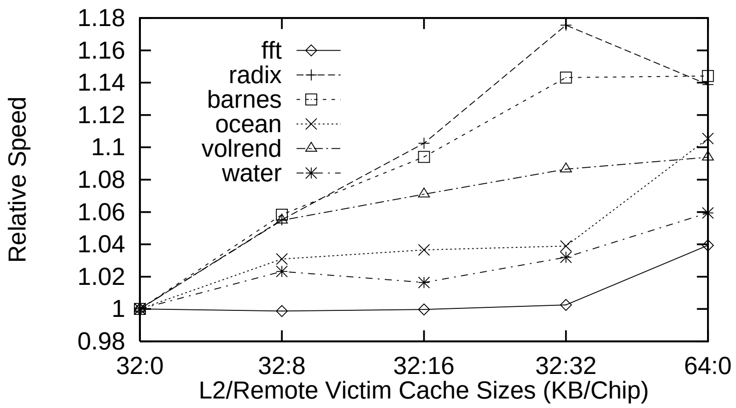

Fig. 4. Effect of L2 Cache and Remote Victim Cache Sizes

Remote victim cache did not improve the performance of fft. Only 0.1% of L2 misses hit at remote victim cache. It even increased remote accesses: remote victim cache prevents write back of modified blocks to home location, and results in remote access (between home and owner nodes) when another processor accesses it. In radix, remote victim cache improved the performance most significantly. In radix, most of remote victim cache hits ($> 94\%$) were in write mode. It saved a large fraction of remote memory accesses in write mode. On the other hand, high hit ratio of 64:0 mainly saved local read misses whose penalty is much lower than remote write misses. Hence, 32:32 outperformed 64:0. In barnes, 32:32 is advantageous for read access while 64:0 is advantageous for write (especially private data) access. Thus, the performance of 32:32 and 64:0 were similar. As shown in Table 3, the fraction of remote miss is quite small in ocean. Also, shared L2 structure benefits for the nearest-neighbor communication pattern of ocean. The nature of remote victim cache that it only accommodates remote blocks was not beneficial for ocean. Thus, 64:0 outperformed 32:32. In volrend, remote victim cache was effective. However, as shown in Table 3, large fraction of L2 misses were private data, which were not saved by remote victim cache. Hence, 64:0 slightly outperformed 32:32. Remote victim cache was not very effective for water since it has high L2 hit ratio and large fraction of L2 misses are private access. 32:32 was only better by 3% , and it was outperformed by 64:0 which was better than base case by 6%.

5 Related Work

Olukoton and others compared a six-issue superscalar processor and a four-way on-chip multiprocessor, for both parallel applications and multiprogramming workload [1]. They assumed the technology of 1997, and that only L1 and L2 can be implemented on a chip. We assumed technology of near future, and hence main memory was also included on the same chip. Remote cache and its augmentation, remote victim cache for CC-NUMA multiprocessor were studied in [5, 6]

6 Conclusions

In this paper, we examined design options to exploit cache resource of on-chip multiprocessor. We evaluated the effectiveness of these options through execution-driven simulations. Using L2 cache exclusively for shared data improved three out of six benchmark programs from SPLASH2 suite up to 10%. We also examined the effectiveness of remote victim cache. When the total amount of cache (L2 and remote victim cache) is limited, our results showed that in five out of six benchmark programs a large L2 cache gave similar or better performance than the combination of L2 and remote victim cache.

Topics of further study include efficient data placement among on/off-chip memory units and suitable cache coherence protocol and latency hiding for a larger configuration (more than ten's of chip).

References

1. K. Olukotun et al., "The Case for a Single-Chip Multiprocessor" in *Proceedings of 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM Press, New York, 2–11, October 1996.
2. Y. Nunomura, T. Shimizu and O. Tomisawa, "M32R/D-Integrating DRAM and Microprocessor", IEEE Micro, Vol. 17, No. 6, 40–48, November/December 1997.
3. A-T. Nguyen, M. Michael, A. Sharma and J. Torrellas, "The Augmint Multiprocessor Simulation Toolkit for Intel x86 Architectures", in *Proceedings of 1996 International Conference on Computer Design*, 486–490, October 1996.
4. S. C. Woo et.al., "The SPLASH-2 Programs: Characterization and Methodological Considerations", in *Proceedings of the 22nd International Symposium on Computer Architecture*, 24–36, June 1995.
5. Z. Zhang and J. Torrellas, "Reducing Remote Conflict Misses: NUMA with Remote Cache COMA", in *Proceedings of International Symposium on High Performance Computer Architecture*, 272–281, February 1997.
6. A. Moga and M. Dubois, "The Effectiveness of SRAM Network Caches in Clustered DSMs", in *Proceedings of The Fourth International Symposium on High Performance Computer Architecture*, 103–112, February 1998.