Chip Multiprocessors – A Cost-effective Alternative to Simultaneous Multithreading

BORUT ROBIČ
Faculty of Computer and Information Sc.
University of Ljubljana
Tržaška 25, 1000 Ljubljana
SLOVENIA

JURIJ ŠILC
Computer Systems Department
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana
SLOVENIA

THEO UNGERER
Dept. of Computer Design and Fault Tolerance
University of Karlsruhe
76128 Karlsruhe
GERMANY

Abstract: - In this paper we describe the principles of the chip multiprocessor architecture, overview design alternatives and present some example processors of this type. We discuss the results of several simulations where chip multiprocessor was compared to other advanced processor architectures including superscalars and simultaneous multithreading processors. Although simultaneous multithreading seems to be most efficient when compared architectures have equal total issue bandwidth, chip multiprocessor may outperform simultaneous multithreading when implemented with equal number of transistors.

Key-Words: - chip multiprocessor, instruction-level parallelism, simultaneous multithreading, thread-level parallelism.

1 Introduction

Memory access *latency* is the interval between the processor's sending of the request for memory access until the return of the result. There are additional latencies that can arise in a processor's pipeline and are due to long operations and branch interlocking. Clearly, the latency becomes a problem if the processor spends a large fraction of its time sitting idle and waiting for memory accesses to complete [13].

A way to look at latencies that arise in a pipelined execution is the *opportunity cost* in terms of the instructions that might be processed while the pipeline is interlocked. The opportunity cost of single-issue processors is the number of cycles lost by latencies. Multiple-issue processors (e.g., superscalar [1], VLIW [9], etc.) potentially execute more than one instruction per cycle, so the opportunity cost also depends on the issue bandwidth. Unfortunately, due to missing instruction-level parallelism rarely all the issue slots are fully filled.

The opportunity cost can be reduced by the additional utilization of more *coarse-grained parallelism*. The main coarse-grained parallelism approaches are represented by the *chip multiprocessor* (CMP) [2, 6, 12, 15] and the *simultaneous multithreading* (SMT) [3, 15, 16].

In this paper we survey the chip multiprocessor approach and compare it with the simultaneous multithreading approach.

2 Coarse-Grained Parallelism

The first main approach to coarse-grained parallelism is SMT. This approach combines the multithreading technique with a wide-issue superscalar processor. The second approach is CMP. This approach integrates two or more complete processors on a single chip.

In the following we will denote by **p** x (**t**, **i**) approach with **p** CPUs per chip, each CPU equipped with **t** threads and **i** issue slots.

2.1 SMT

Figure 1 demonstrates a four-threaded eight-issue SMT processor. The processor exploits instruction-level

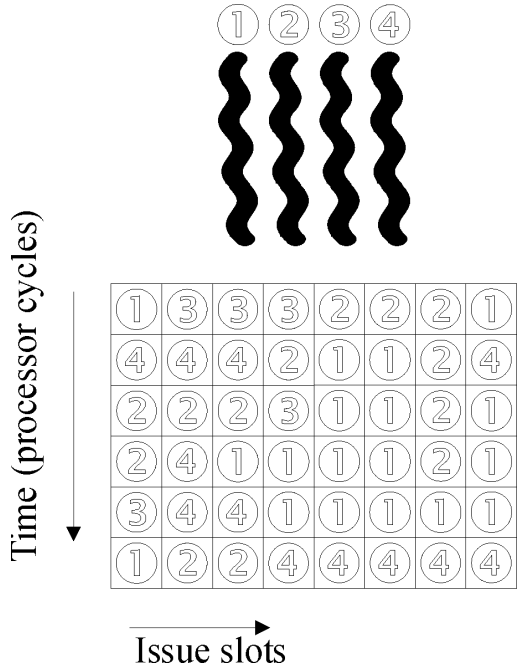

Figure 1: SMT 1 x (4, 8) (one CPU, four-threaded, eight-issue).

parallelism by selecting instructions from any thread that can potentially issue. If one thread has high instruction-level parallelism, it may fill all horizontal slots depending on the issue strategy of the SMT processor. If multiple threads each have low instruction-level parallelism, instructions of several threads can be issued and executed simultaneously.

2.1 CMP

Figure 2 shows a CMP with four two-issue CPUs on a single chip. Each CPU is assigned a thread from which it can issue up to two instructions each cycle. Thus, each CPU has the same opportunity cost as in a two-issue superscalar model. The CMP is not able to hide latencies by issuing instructions of other threads. However, because horizontal losses will be smaller for two-issue than for high-bandwidth superscalars, a CMP of four two-issue processors will reach a higher utilization than an eight-issue superscalar processor.

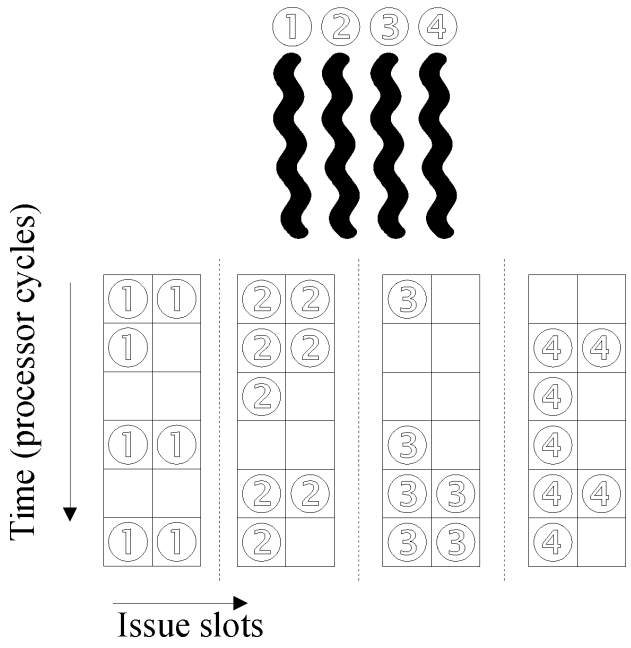

Figure 2: CMP 4 x (1, 2) (four CPUs, four-threaded, two-issue).

3 CMP Design Alternatives

Today the most common organizational principles for multiprocessors are the symmetric multiprocessor (SMP), the distributed shared memory multiprocessor (DSM), and the message-passing shared-nothing multiprocessor.

The SMP and the DSM multiprocessors feature a common address space, which is implemented in the SMP as a single global memory where each memory word can be accessed in uniform access time by all processors (uniform memory access). In the DSM multiprocessor a common address space is maintained despite physically distributed memory modules. A processor in a DSM may access data in its local memory faster than in the remote memory (the memory module local to another processor). DSM multiprocessors are therefore nonuniform memory access systems. Shared-nothing multiprocessors feature physically distributed memory modules and no common address space. Therefore, communication can only be performed by passing messages between processors. Shared-nothing multiprocessors are highly scalable but harder to program than shared-memory multiprocessors. They are beyond the scope of today's reasoning about CMPs, which, by their tight physical coupling on a single chip, may also feature a very tight coupling of instruction streams, usually expressed by a common memory organization.

The principal organizational forms of multiprocessors do not regard cache organization. Commodity microprocessors, which are usually used today as building blocks for multiprocessors, contain on-chip caches, often coupled with off-chip secondary cache memories. Shared-memory multiprocessors maintain cache coherence by a cache coherence protocol. SMPs consist of a moderate number of commodity microprocessors with cache memories coupled by a fast memory bus with the global memory. In the latest SMPs the memory bus is replaced by an address bus and a data crossbar switch for faster transfer of cache lines. SMPs are the starting point for CMPs.

From the applications perspective, whether a CMP works best depends on the amount and the characteristics of the parallelism in the applications. These fall into three broad classes depending on the degree of interprocessor communication, which can be low, moderate, or high. From the architectural perspective, the performance of a CMP will depend on the level of the memory hierarchy at which the CPUs of the CMP are interconnected.

In order to develop insight about the most appropriate memory hierarchy level for connecting the CPUs in a CMP, three alternatives were compared in [8]: a shared-main memory multiprocessor (i.e., the typical SMP today), a shared-secondary cache multiprocessor, and a shared-primary cache multiprocessor. They found that, when applications have a high or moderate degree of interprocessor communication, both shared-primary cache and shared-secondary cache architectures perform similarly and outperform the shared-main memory architecture substantially. There are two reasons for this. First, the shared cache was assumed large enough to accommodate most of the working sets of independent threads running on different CPUs, so that the cache miss rate is low. Second, when there is interprocessor communication, it is handled very efficiently in the

shared (primary or secondary) cache. Even for applications with little or no interprocessor communication, the performance of the shared-primary cache architecture is still slightly better than shared-main memory architecture.

To maintain the performance growth of microprocessors, the details of implementing a CMP were discussed in [10].

4 CMP Examples

In the following we itemize some realized CMP examples:

- The Texas Instruments TMS320C8x (or 'C8x) family of processors are CMPs suitable for system-level and embedded implementations [15]. Such a CMP is a multimedia video processor which replaces several system components by integrating multiple processors, memory control logic, instruction cache and internal memory, an advanced DMA controller, and video timing generation logic ('C80 only) onto a single chip. They provided an order of magnitude increase in computational power over existing digital signal processors (DSPs) and general-purpose processors in 1994. Two types of processors are combined in the chip: a single RISC master processor (MP) and a number of VLIW DSP-like parallel processors (PP). Moreover, the chip contains a programmable DMA transfer controller (to handle all off-chip data transfer operations required by the MP and PPs), a video controller, and a boundary-scan test access port. All processors are interconnected by a crossbar with instruction caches, and data RAM and parameter RAM areas. The 'C8x family consists of two members, the 'C80 [15], which features four PPs, and the 'C82 [4] with only two on-chip PPs.
- The Hydra proposal [6] is composed of four 2-issue superscalar CPUs on a single chip. Each of the CPUs is similar to a small MIPS R10000 processor and is attached to its own on-chip primary instruction and data caches. In addition, a single, unified secondary cache is included on the chip.
- Sun's microprocessor architecture for Java computing (MAJC) based on a VLIW approach and embraces Java technology [12]. The MAJC-5200 is the first implementation of the MAJC architecture. Exploiting a thread-level parallelism, the MAJC-5200 processor has two CPUs on the same chip. Each CPU is 4-threaded block interleaving VLIW processor.
- IBM Power 4 has two 5-issue superscalar CPUs on the same chip [2].

5 CMP versus SMT

In this section, we compare the CMP and SMT approach.

In [11, 16] the results of simulations comparing SMT with CMP are given (see Table 1). The two simulations produced slightly different results. The difference follows from the high number of execution units in [16] (for example, up to eight load/store units are used in this simulation ignoring hardware cost and design problems, whereas the performance of SMT model in [11] is restricted by the assumption of a single load/store unit). In [16] the SMT performs better than the CMP, whereas

Table 1. Simulation results in IPC.

Approach	**p** x (**t**, **i**)	[16]	[11]
SMT	1 x (4, 8)	4.15	3.37
	1 x (8, 8)	6.64	4.19
CMP scalar	8 x (1, 1)	5.13	6.07
CMP superscalar	2 x (1, 4)	1.94	2.56
	4 x (1, 2)	3.44	4.32
CMP + SMT	2 x (4, 4)	6.80	6.80

Table 2. Architectures simulated in [3]: SS (1 x (1, 8) superscalar), TSS (1 x (8, 8) cycle-by-cycle multi-threaded superscalar), CMP2 (2 x (1, 4) chip multiprocessor), CMP4 (4 x (1, 2) hip multiprocessor), and SMT (1 x (8,8) simultaneous multithreading processor).

Features	SS	TSS	CMP2	CMP4	SMT
# of CPUs	1	1	2	4	1
CPU issue bandwidth	8	8	4	2	8
# of threads	1	8	1/CPU	1/CPU	8
# of arch. registers	32	32/thread	32/CPU	32/CPU	32/thread

in [11] the CMP reaches a higher throughput than the SMT, when using the same issue bandwidth and number of threads (see 1 x (8,8) and 8 x (1,1) in Table 1). However, if chip costs were taken into consideration, a 4-threaded 4-issue superscalar processor showed the best performance/cost relation.

Further simulations were described in [3]. They compared two CMPs (one with 2 CPUs and one with 4 CPUs) with a superscalar, cycle-by-cycle interleaving multithreaded superscalar, and SMT (Table 2). The simulation results (see Table 3) were obtained on a workload which consisted of a group of coarse-grained (parallel threads) and medium-grained (parallel loop iterations) parallel programs.

Table 3. Simualtion results in IPC: SS (1 x (1, 8) superscalar), TSS (1 x (8, 8) cycle-by-cycle multithreaded superscalar), CMP2 (2 x (1, 4) chip multiprocessor), CMP4 (4 x (1, 2) chip multiprocessor), and SMT (1 x (8,8) simultaneous multithreading processor).

Threads	SS	CPM2	CMP4	TSS	SMT
1	3.3	2.4	1.5	3.3	3.3
2		4.3	2.6	4.1	4.7
3			4.2	4.2	5.6
4				3.5	6.1

The average instruction throughput of an 8-issue superscalar was 3.3 IPC, which is already high compared to other measured superscalar IPCs, but rather low compared to the eight instructions possibly issued per cycle. The superscalar's inability to exploit more ILP or any thread-level parallelism contributed to its lower performance. By exploiting thread-level parallelism, a cycle-by-cycle interleaving multithreaded superscalar technique provided an average instruction throughput of 4.2 IPC. This IPC occurred with only four threads while performance fell with additional threads. One of the reasons is that a cycle-by-cycle interleaving multithreaded superscalar can issue instructions from only one thread each cycle and therefore cannot hide conflicts from interthread competition for shared resources. SMT obtained better speedups than CMP2 and CMP4, the latter being CMPs with respectively, two four-issue, and four two-issue CPUs. Speedups on the CMPs were hindered by the fixed partitioning of their hardware resources across the CPUs. Bridging of latencies is only possible in the multithreaded processor approaches, and not in CMP. CPUs in CMPs were idle when thread-level parallelism was insufficient. Exploiting large amounts of ILP in the unrolled loops of individual threads was not possible due to the CPU's smaller issue bandwidth in CMP. On the other hand, an SMT processor dynamically partitions its resources among threads, and therefore can respond well to variations in both types of parallelism, exploiting them interchangeably.

In contrast to the previous simulation (which compared architectures having constant total issue bandwidth), the simulation in [5] first fixed a standard chip area as well as integration density, and then determined the parameters for three architectures: superscalar, CMP, and SMT (Table 4). They argued that design complexity for a 16-issue CMP was similar to that of a 12-issue superscalar or a 12-issue SMT processor.

In this case, 8 x (1, 2) CMP outperforms a 12-issue superscalar and a 1 x (8, 12) SMT on four SPEC95 benchmark programs. Table 5 shows the performance of the three processors relative to a single 2-issue superscalar.

The CMP achieved higher performance than the SMT due to a total of 16 issue slots instead of 12 issue slots for the SMT.

Table 4. Architectures simulated in [5]: SS (1 x (1, 12) superscalar), CMP (8 x (1, 2) chip multiprocessor), and SMT (1 x (8,12) simultaneous multithreading processor).

Features	SS	CMP	SMT
# of CPUs	1	8	1
CPU issue bandwidth	12	2	12
# of threads	1	1/CPU	8
# of arch. registers	32	32/CPU	32/thread

Table 5. Performance of SS (1 x (1, 12) superscalar), CMP (8 x (1, 2) chip multiprocessor), and SMT (1 x (8,12) simultaneous multithreading processor) relative to 2-issue superscalar.

Benchmark program	SS	CMP	SMT
compress	1.5	1.25	1.5
mpeg	3.25	7.75	6.75
tomcatv	1.5	7.5	5.75
multiprogram	1.5	8	7

6 Conclusions

The performance race between SMT and CMP has yet to be decided. CMP and SMT each have advantages and disadvantages, and it will be interesting to see which approach offers better performance. CMP will be easier to implement, but only SMT has the ability to hide latencies. A functional partitioning as required by the on-chip wire-delay of future microprocessors is not easily reached within a SMT processor due to the centralized instruction issue. A separation of the thread queues is a possible solution, although it does not remove the central instruction issue.

A combination of SMT with the CMP was proposed in [7, 11] and shows the path towards a CMP consisting of moderately equipped SMTs.

Usually, a CMP will feature separate primary instruction and data caches per on-chip CPU and an optional unified secondary cache. If the CPUs always execute threads of the same process, the secondary cache organization will be simplified, because different processes do not have to be distinguished.

Moreover, the multiprocessor which is formed by the CMPs will be a symmetric multiprocessor (SMP) or even a distributed shared-memory multiprocessor (DSM).

Similarly, if all (hardware-supported) threads of a SMT processor always execute threads of the same process, preferably in SPMD fashion, a unified (primary) instruction cache may prove useful since the code can be shared between the threads. Primary data cache may be unified or separated between the threads depending on the access mechanism used.

If CMP or SMT are the design choice of the future, the impact on multiprocessor development will favor shared-memory multiprocessors (either SMPs or DSMs) over message-passing machines. Since multithreading and message passing do not mix well even in state-of-the-art multiprocessor programs, there is an indication that message-passing programs using PVM or MPI will soon be outdated and produce a legacy problem.

In future, we will observe merging of SMT and CMP with today's multiple-issue processor techniques.

References:

[1] Tilak Agarwala, John Cocke, High performance reduced instruction set processor, *Technical Report #55845*, IBM Thomas J. Watson Research Center, 1987.

[2] Keith Diefendorff, Power4 focuses on memory bandwidth, *Microprocessor Report*, Vol. 13, No. 13, Oct. 1999.

[3] Susan J. Eggers, Joel S. Emer, Henry M. Levy, Jack L. Lo, Rebecca M. Stamm, Dean M. Tullsen, Simultaneous multithreading: a platform for next-generation processors, *IEEE Micro*, Vol. 17, Sep./Oct. 1997, pp.12-19.

[4] Jeremiah Golston, Single-chip H.324 video-conferencing, *IEEE Micro*, Vol. 16, No. 4, 1996, pp.21-33.

[5] Lance Hammond, Basem A. Nayfeh, Kunle Olukotun, A single-chip multiprocessor, *Computer*, Vol. 30, No. 9, Sep. 1997, pp.79-85.

[6] Lance Hammond, Kunle Olukotun, Considerations in the design of Hydra: a multiprocessor-on-chip microarchitecture, *Technical Report CSL-TR-98-749*, Computer Systems Laboratory, Stanford University, 1998.

[7] Venkata Krishnan, Joseph Torellas, A clustered approach to multithreaded processors, *Proc. 1998 IPPS/SPDP Conf.*, Orlando, FL, Mar./Apr. 1998, pp.627-634.

[8] Basem A. Nayfeh, Lance Hammond, Kunle Olukotun, Evaluation of design alternatives for a multiprocessor microprocessor, *Proc. 23rd Ann. Int. Symp. Comp. Arch.*, Philadelphia, PA, May 1996, pp.67-77.

[9] Alexandru Nicolau, Joseph A. Fisher, Measuring the parallelism available for very long instruction word architecture, *IEEE Transaction on Computers*, Vol. C-33, No. 11, 1984, pp.968-976.

[10] Kunle Olukotun, Basem A. Nayfeh, Lance Hammond, Ken Wilson, Kunyung Chang, The case for a single-chip multiprocessor, Proc. 7th Int. Conf. ASPLOS, Cambridge, MA, Oct. 1996, pp.2-11.

[11] Ulrich Sigmund, Theo Ungerer, Evaluating a multithreaded superscalar microprocessor versus a multiprocessor chip, *Proc. 4th PASA Workshop Parall. Sys. and Algorithms*, Jülich, Germany, Apr. 1996, pp.147-159.

[12] Subramania Sudharsanan, MAJC-5200: a high performance microprocessor for multimedia computing, *Proc. Workshop on Parallel and Distributed Computing in Image Processing, Video Processing, and Multi-media*, Cancun, Mexico, May 2000.

[13] Jurij Šilc, Borut Robič, Theo Ungerer, *Processor Architecture: From Dataflow to Superscalar and Beyond*, Springer-Verlag, Berlin, New York, 1999.

[14] Jurij Šilc, Borut Robič, Theo Ungerer, Simultaneous multithreading – blending thread-level and instruction-level parallelism in advanced microprocessors, *Proc. 5th Word Multiconf. on Circuits, Systems, Comm. & Computers*, Rethymnon, Greece, July 2001. (submitted)

[15] Texas Instruments, *TMS320C80 Technical Brief*, Texas Instruments, Houston, TX, 1994.

[16] Dean M. Tullsen, Susan J. Eggers, Henry M. Levy, Simultaneous multithreading: maximizing on-chip parallelism, *Proc. 22nd Ann. Int. Symp. Comp. Arch.*, Santa Margherita Ligure, Italy, June 1995, pp.392-403.