Regular Paper

A Case Study: Energy Efficient High Throughput Chip Multi-Processor Using Reduced-complexity Cores for Transaction Processing Workload

Hisashige Ando,† Akira Asato,†† Motoyuki Kawaba,††
Hideki Okawara†† and William Walker†††

The pursuit of instruction-level parallelism using more transistors produces diminishing returns and also increases power dissipation of general purpose processors. This paper studies a chip multi-processor (CMP) with smaller processor cores as a means to achieve high aggregate throughput and improved energy efficiency. The benefit of this design approach increases as the number of cores on a chip increases, as enabled by semiconductor process scaling. The feasibility of a processor core 40% of the size of a baseline high performance processor that delivers about 70% of its performance is shown. The CMP populated by smaller cores to fill the same silicon area delivers 2.3 times higher performance in transaction processing represented by TPC-C benchmarks than the baseline processor scaled into the same technology. The CMP also achieves 38% higher energy efficiency.

1. Introduction

The number of transistors available on a chip is continuously increasing in accordance with Moore's law. In the past, additional transistors have been used for processor performance improvement by implementing mechanisms to exploit more instruction level parallelism. However, the effort to find more parallelism is becoming increasingly difficult and the return is diminishing. Also, the increase in power dissipation of a chip becomes an obstacle to achieve higher performance by using more transistors. Finally, as the disparity between the processor and memory speed widens, memory access latency occupies a significant portion of the instruction execution time.

Chip multiprocessors have been proposed as solutions to the saturation of usable instruction parallelism and memory latency problems [1)~5)]. A commercial server using a dual core processor [6)] is on the market. Although thread level parallelism is only useful for multi-threaded applications, on-line business applications which handle a large number of simultaneous users such as web servers and transaction processing are inherently highly parallel in nature and therefore can take advantage of thread level parallelism. The data in Ref. 7) shows that OLTP, SPECjbb, Apache and Slashcode programs exhibit good multi-processor scalability. An aggregate throughput from all the threads running on a chip/system is a more important performance measure for these applications than the traditional single thread execution speed. In this paper, we used the SPARC64 V commercial high end microprocessor [8),9)] as a baseline to investigate a design approach that delivers higher throughput from the same silicon area. In our design approach, a smaller simplified core is derived from the baseline processor, then it is multiply-instantiated to create a CMP. We show that aggregate performance of the small core CMP is significantly higher than the baseline. We limited the exploration space to the baseline modification for reliable results both in chip area and performance estimate. A new design from scratch, or a significantly modified baseline design, such as the change to an in-order design may yield better CMP design, but the accuracy of the estimates will be sacrificed.

Figure 1 is a normalized plot showing the advance of microprocessor performance measured in SPEC benchmark. **Figure 2** is the clock frequency, number of transistors ($\#Tr$) and the number of processor core transistors ($\#CoreTr$) of the microprocessors presented at the International Solid State Circuits Conference (ISSCC) over time. The performance improves by 40%/year and the clock frequency increases by 25%/year. Hence, the performance per cycle (proportional to instruction per cycle (IPC)) improves by 16%/year. Assuming this IPC improvement is effected by increasing the number of transistors, it is proportional to

† Fujitsu Limited, Japan
†† Fujitsu Laboratories Ltd., Japan
††† Fujitsu Laboratories of America, USA

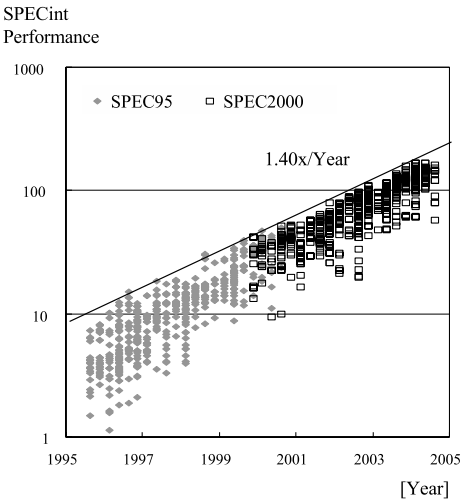


Fig. 1 Microprocessor SPECint performance trend (SPECint2000 performance is divided by 10 to make them continuous with the SPECint95 trend).

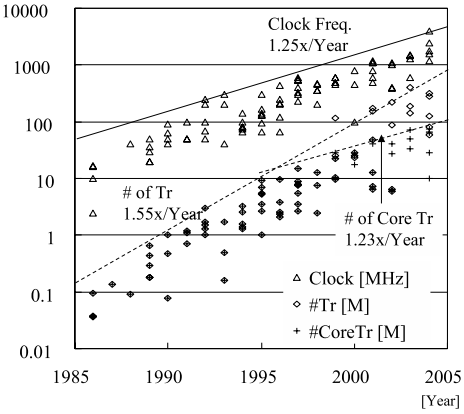


Fig. 2 Microprocessor number of transistors and clock frequency trend (Microprocessors presendted in ISSCC for each year).

$(\#Tr)^{0.250}$. This is much worse than Pollack's rule [10], which states that performance is proportional to $(\#Tr)^{0.33\sim0.5}$. Since the recent increase in transistor count is mostly in the cache, by counting the processor core transistors only, the IPC is proportional to $(\#CoreTr)^{0.504}$. This result agrees with Pollack's rule. Although Pollack's rule is not a technically precise relationship, since IPC improvements are possible without increasing in the number of transistors, such as by compiler improvements. Conversely, additional transistors do not necessarily improve IPC, such as by using deeper pipelining or a carry select adder in place of a simpler but slower adder. Nevertheless, the historical transistor vs. performance trend can be used as a guideline. Based on this trend, we work backwards to design a processor core with one-third to one-half the number of transistors and about 60∼70% of the performance of the baseline processor core.

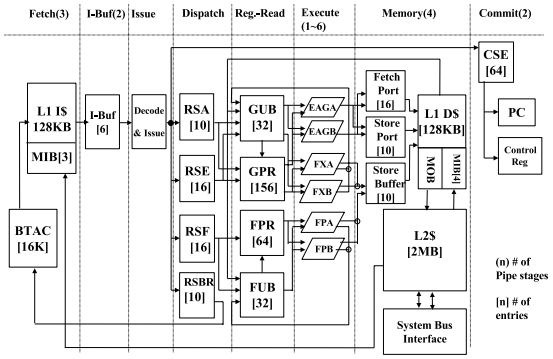


Fig. 3 SPARC64 V, baseline processor block diagram.

Codrescu, et al., [11] investigated the relationship between the core size and aggregate performance of a fixed size chip, but, their work is based on the simplified relationship of the chip size and clock frequency vs. the number of execution units, etc., whereas we studied the performance and power relationship of various CMP design points at a finer level of granularity, starting from a high-end microprocessor SPARC64 V.

2. Baseline Microprocessor Architecture

Figure 3 shows a block diagram of the baseline microprocessor. The pipeline stages and the size of buffers are marked in the figure. This processor delivers 767 SPECint2000_base and 1047 SPECfp2000_base as used in Fujitsu unix server systems running at 1.35 GHz. The 128 CPU server achieved 455.8 K tpmC, which was the highest non-cluster TPC-C benchmark score when it was registered.

The baseline processor is a 4-issue superscalar design with in-order issue, out of order execution and in-order commit. It has 128 KB each level-1 instruction cache (L1I$) and level-1 data cache (L1D$). The L1I$ has a 3-entry move-in-buffer (MIB, also called as miss buffer) for non-blocking operation up to 3 misses. The L1D$ has a 4-entry MIB.

The decode & issue block decodes up to 4 instructions in one cycle and issues them into the reservation stations (RSA, RSE, RSF or RSBR). Dispatched instructions read either fixed point reorder buffer (GUB), fixed

point registers (GPR), floating point reorder buffer (FUB) or floating point registers (FPR) for operands. When the instruction becomes ready to execute with the operands, it is dispatched from the reservation station into the corresponding execution pipeline. There are two integer execution pipelines (FXA/B), two floating point execution pipelines (FPA/B), two load/store pipelines (EAGA/B) and one branch pipeline. For the load and store instructions, the address is calculated and stored into either the fetch port or store port. The store data is buffered in the store buffer. Then, the L1D$ is accessed with this information. The L1I$ and L1D$ are backed by a 2 MB 4way set associative level-2 cache (L2$) integrated on the chip. The latency of the L1D$ is 4 cycles from the address generation and the latency of the L2$ is 6 cycles form the L1$ miss.

There are two fully associative 32 entry micro TLBs (Translation Lookaside Buffer), one each for L1I$ and L1D$. These micro TLBs function as caches for the 1024 entry main TLB. (The micro and main TLBs are not shown in Fig. 3.)

A branch target address cache (BTAC) is a 128 KB memory organized as a 4-way set associative 16 K entry array. Each BTAC entry holds a branch target address, address tag and branch history information of a taken branch instruction.

It also contains a commit stack entry (CSE), a program counter (PC) and other control registers.

3. Performance Evaluation Method

The SPEC CPU2000 benchmark was selected as the performance evaluation workload for the micro-architecture level resource reduction of the processor since this benchmark is widely accepted as a representative collection of programs which exercise various aspects of processor core microarchitecture. The TPC-C benchmark is used for the performance evaluation in multi-processor environments since this is a typical transaction workload. It uses a large memory area as a data base buffer and stresses the cache and memory subsystem. The run time traces of SPEC benchmark program were collected using the Shade tool [12] running on the Solaris operating system. Using a 16CPU SPARC/Solaris server, TPC-C traces were obtained using an in-house trace tool that records all the instruction executions including the ones in the kernel state. This is important since more than 20% of the TPC-C instructions are executed in the kernel state, as shown in Refs. 13) and 14).

A cycle-accurate trace driven simulator developed for the SPARC64 V processor [15] was used for performance evaluation, with modifications for CMP support. Since the simulator can not handle asynchronous multiple clock domains, the CPU clock is chosen to be an integer multiple of the DRAM clock, for example, 1.33 GHz with a 133 MHz DRAM clock. The multiprocessor synchronization instruction sequences are simulated as they were captured in the trace. Although the trace driven simulation can not handle multiprocessor interaction adaptively, the error involved is believed to be small as Ranganathan, et al., [16] reported that lock contention overhead is less than 5%. In addition, the synchronization communication in a CMP is faster than in a multiprocessor system on a board and the performance impact is less.

4. Processor Core Area Reduction and Its Performance Impact

The reduction of the processor core area was performed in four steps, with the area performance trade-off verified in the first three steps by the performance simulator. The reason we divided the process into four steps is described in the subsections below. The results of the reductions are summarized in **Table 1**, **Table 2** and Fig. 6 at the end of this section. The fourth step involves a custom layout resizing of large circuit macros, and does not involve the performance simulator. In this section, we also ignore clock cycle time improvement that might be possible due to the area reduction.

The baseline processor core area is 104.6 mm^2 using a 130 nm semiconductor process. However, it is estimated that a reduction to 93.2 mm^2 would be possible with layout improvement, and this reduced area is used in the performance study to avoid skewing the results. This layout improvement and the use of custom macros described in Section 4.4 could have been done for the baseline processor if design resource and schedule limitations did not exist.

4.1 Step 1: Issue and Execution Resources Reduction

Since our goal was to improve performance per unit area by reducing the core size by at least by 50%, the instruction issue width was reduced from 4 to 2 and the integer, floating point and load/store execution pipelines were

Table 1 Resource reduction item list of each step.

Step		Baseline core	Small core
1	Instruction Issue	4	2
	Execute Units	2 each	1 each
2	L1I$/D$	128 K(2way)	16 KB(2way)
3	CSE	64	24
	Fetch/Store port	16/10	8/4
	GUB/FUB	32/32	16/16
	(FX/FP Reorder buffers)		
	RSA/RSBR	10/10	4/4
	RSE/RSF	8×2/4×2	8/8
	L1I$ Move In Buffer	3	1
	L1D$ Move In Buffer	4	2
	L1$ → L2$ data bus	16 B	8 B
	L2$ → L1$ data bus	32 B	8 B
	BTAC	4 K× 4way	1 K × 2way
	TLB	1024	512
4	Register File	Standard cell	Custom
	TLB CAM	Standard cell	Custom

Table 2 Processor core area reduction (130 nm).

Chip (in mm^2) Area			Area	Layout Improve	Reduction step1	step2	step3	step4
IU	Array	BTAC	9.33	8.50	8.50	8.50	1.06	1.06
		IBUF	6.22	4.20	2.94	2.94	2.94	2.94
	Logic		15.55	12.62	10.56	10.56	6.54	6.54
	Subtotal		*31.1*	*25.32*	*22.00*	*22.00*	*10.54*	*10.54*
EU	RegFile		10.66	10.66	9.14	9.14	8.14	1.09
	Logic	ALU etc.	11.88	8.54	6.44	6.44	6.44	6.44
	Subtotal		*22.54*	*19.20*	*15.58*	*15.58*	*14.58*	*7.53*
SU	Array	L1I$ Tag	7.18	7.18	7.18	1.15	1.15	1.15
		L1D$ Tag	9.6	9.6	9.6	1.65	1.65	1.65
		TLB	3.1	3.1	3.1	3.1	2.1	2.1
		uTLB	7.22	7.22	7.22	7.22	7.22	0.32
	Logic		23.89	22.45	21.38	16.52	14.04	14.04
	Subtotal		*50.99*	*49.55*	*48.48*	*29.64*	*26.16*	*19.26*
Total			**104.63**	**93.21**	**86.06**	**67.22**	**51.28**	**37.33**

halved. These reductions were chosen as the first step since they lower the usage of level 1 caches and other buffer resources and make it easier to identify the excess resources in the following reduction steps.

The performance loss for this reduction is 10.8% in SPECint2000, 12.5% in SPECfp2000 and 1.9% in TPC-C. Since these reductions throttle the issue-execution, performance loss in SPECmark is relatively large compared to the area reduced.

4.2 Step 2: L1 Cache Size Reduction

The sizes of the L1 caches were reduced as the second step since they occupy a large chip area. The SPEC2000 and TPC-C performance sensitivity to the L1$ size (both instruction and data cache sizes were made equal) was obtained as shown in **Fig. 4**. With this data, the sizes

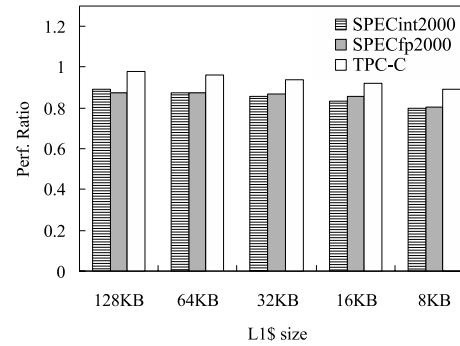


Fig. 4 Relative performance with L1$ size reduction (Baseline 4 issue core = 1.0).

of the L1$s were reduced to 16 KB. Although the performance degradation with 8 KB cache is 3∼6% compared to 16 KB cache, a 0.8 mm^2 area reduction is too small a gain compared to

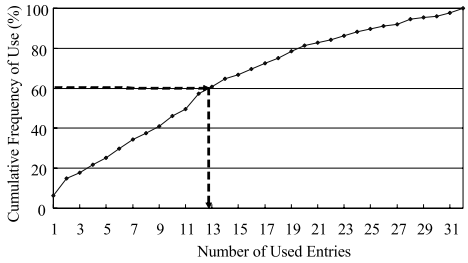


Fig. 5 Cumulative GUB entry usage. For the 60% of the execution time, 13 or less GUB entries are used among 32 entries in the baseline processor.

the performance degradation.

4.3 Step 3: Buffer Resources Reduction

For the third step reduction, histograms of the usage of each buffer entry were obtained by simulation with the processor core after the second step reduction. Then the sizes of the various buffers on the chip were reduced to cover more than 60% usage of each histogram as a guideline. An example of GUB usage is shown in **Fig. 5**. The final sizes of the buffers were mostly rounded up to the nearest integer power of 2. Also the number of BTAC entries was reduced from 16 K (4 K × 4way) to 2 K (1 K × 2way) and the TLB size was reduced from 1024 to 512 entries.

The buffers reduced are listed in Table 1 and this step reduces the core size by 15.94 mm^2. With these buffer reductions, the performance losses are 11.5% in SPECint2000, 16% in SPECfp2000 and 23% in TPC-C compared to the performance before the third step reduction.

4.4 Step4: Use of Full Custom Macros

The opportunity to further reduce area with the use of full custom design macros was investigated. The following 3 types of macros are designed: a 156×72 bit 5R2W Fixed Point Register File (FXRF), a small register file (RF) for various registers and buffers, and a special function CAM for the TLB. For each macro, we designed the leaf cells and estimated the area. Subsequently, we built critical path models using the leaf cells and evaluated the access time using a SPICE circuit simulator.

These full custom macros reduced the core area by 14.0 mm^2 in the EU (Execution Unit) register file and SU (Store Unit) uTLB unit. The resulting core size became 37.3 mm^2, as shown in Table 2.

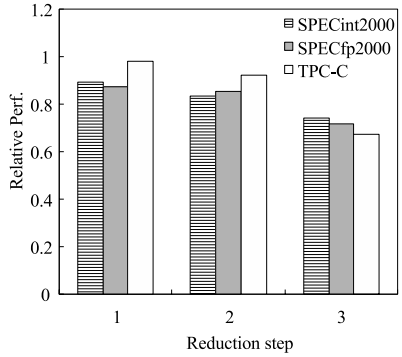


Fig. 6 Relative performance with resource reduction (Baseline 4 issue core = 1.0).

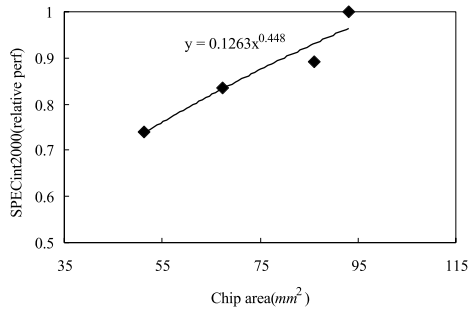


Fig. 7 SPECint2000 performance vs. chip area up to step3.

4.5 Summary of the Resource Reduction

The processor resource sizes after the reduction are summarized in Table 1. Table 2 shows the core area with the breakdown in major processor structures. **Figure 6** shows the relative performance for the selected workloads at each reduction step.

Figure 7 shows the SPECint2000 performance against the processor core area, which is roughly proportional to the number of transistors. The performance is proportional to $(Area)^{0.448}$. Although the square root relationship is empirical, this result demonstrates that, starting from the SPARC64 V processor, smaller core processors having a performance roughly proportional to the square root of the area can be built.

The size of the processor core is reduced to 37.3 mm^2 which is a 40% of the baseline processor core area. The reduced processor core delivers 74% of the SPECint2000, 72% of the SPECfp2000 and 67% of the TPC-C performance compare to the baseline processor.

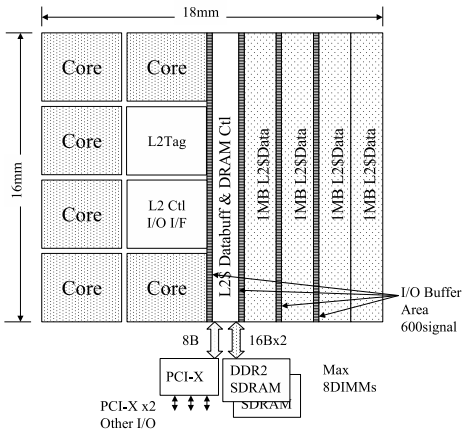


Fig. 8 Chip image with 90 nm semiconductor process, integrating 6 processor cores and 4 MB L2$.

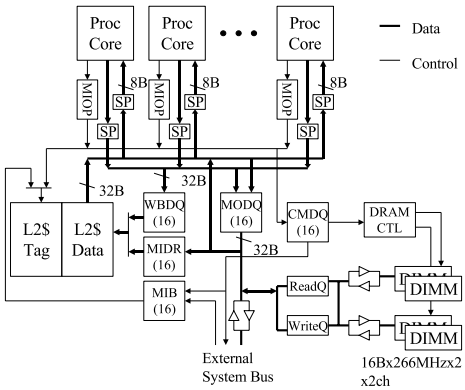


Fig. 9 The CMP block diagram with a memory controller and 2 sets of DIMMs.

5. Single Chip SMP Scalability

In this section, we propose and study a CMP architecture. Using the above described small processor core, multiple processor cores can be integrated on a chip. An L2$ is shared among all cores on the chip, since this configuration reduces cache misses by taking advantage of constructive hit as described in Section 5.1.

As a next generation product, the use of a 90 nm semiconductor process is assumed. The 90 nm process has twice the transistor density of the 130 nm process used for the baseline processor, and allows integrating 6 processor cores and a 4 MB on-chip level-2 cache on a chip slightly larger than the size of the 130 nm baseline processor chip. **Figure 8** shows the chip image.

Figure 9 shows a block diagram of the CMP. A read/write request from the processor core goes into a move-in and move-out port (MIOP) and is then arbitrated with the requests from other processors for accessing L2$ tag for L2$ data array read/write. When the access misses an L2$ tag, the request goes into a command queue (CMDQ) and then the DRAM is accessed through a DRAM controller (DRAM CTL). Write data is supplied from a move-out data queue (MODQ) and a read data is buffered into a move-in data register (MIDR) for the L2$ write. An MIOP can queue up to 4 requests each for read and write. An SP is a 64 B buffer with an 8 B/32 B (or 32 B/8 B) serial to parallel conversion functionality.

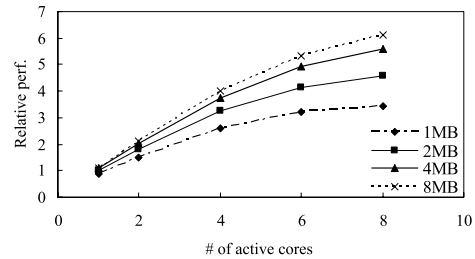


Fig. 10 TPC-C performance scalability vs. the number of active cores (2 MB single core=1.0) with 1∼8 MB L2$ sizes.

The DRAM controller in the baseline processor supports only one set of dual inline memory modules (DIMMs), but it is expanded to support two DIMM sets for the CMP as described below. The address of DIMM banks are interleaved with the 64 byte cache line size when two DIMM sets are used.

Performance scalability of this CMP was investigated. The focus of the investigation is the behavior of the L2$ and the memory subsystem, especially access congestion and resulting increase in latency, since the L2$ and the memory subsystem are shared among all processor cores on a chip.

5.1 The Shared L2 Cache and Its Effect

Figure 10 shows the TPC-C performance scalability against the number of active processor cores. A 133 MHz clock is used for the memory subsystem, and the processor cores are clocked at 1.33 GHz. Although the chip image in Fig. 8 shows 6 cores with 4 MB L2$, these parameters are varied up to 8 cores and 8 MB L2$ for the scalability evaluation.

As all the processor cores are sharing a single L2$, the number of requests increase as the number of active cores increase. The L2$ can process one access request every other cycle. The path between L2$ and each core is 8 B wide and takes 8 cycles to transfer a 64 byte cache

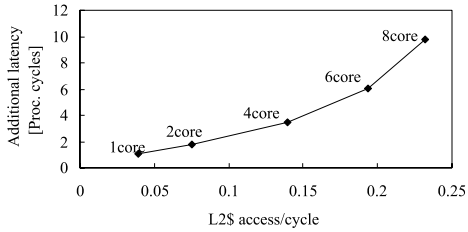


Fig. 11 L2$ bus busy rate and additional access latency due to CMP access congestion. L2$ size is 4 MB.

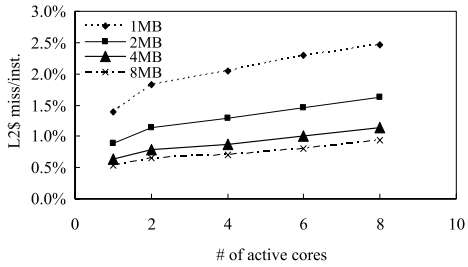


Fig. 12 L2 Cache Miss rate against number of active cores.

line. Hence, the L2$ becomes busier and the access latency increases as the number of active cores increases as shown in **Fig. 11**. Even when only one core is active, the average L2$ access latency is about 1 cycle longer than the minimum latency of 6 cycles. The L2$ is busy for 38.6% of the time when 6 cores are active, as they generate 0.193 access per cycle (shown in Fig. 11), and the L2$ only can handle one access every 2 cycles. An additional 0.488 cycles per instruction (CPI) is incurred at this L2$ busy rate due to the additional L2$ latency of 6.1 cycles (Fig. 11) and the 4.85% L1 I$ and 3.15% L1 D$ miss rates (**Fig. 13**.) This is acceptable compared to the total CPI of 4.4.

Figure 12 shows the L2$ miss rate per instruction. The rate of increase of the L2$ miss is relatively small, 0.6% for the single core and 1.1% for 8 cores with 4 MB L2$.

Figure 13 shows the breakdown of the L2$ access for a self hit, constructive hit and miss. A constructive hit is defined as a hit to a cache line that was last touched by another processor. Figure 13 (a) shows the breakdown for the instruction access. The instruction miss rate is 0.06% for a single active core. This means that almost the entire footprint of the instruction fits in the 4 MB L2$. Since all the cores are processing the same type of transactions and executing the same code, increasing the number of active cores only results in an increase

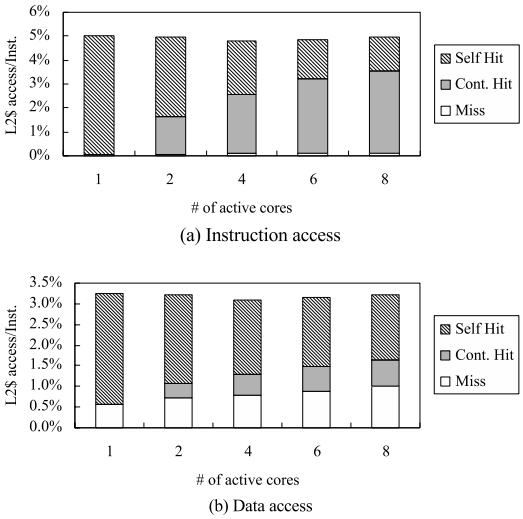


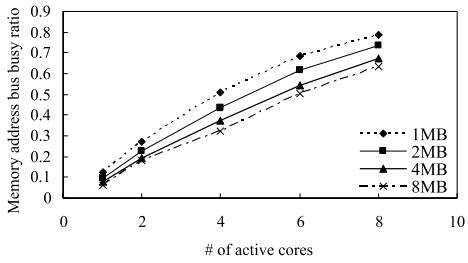
Fig. 13 L2$ access breakdown (4 MB L2$).

in the constructive hit rate for instructions; the miss rate remains at 0.13% with 8 active cores. Figure 13 (b) shows the breakdown for the data access. Although the amount of constructive hits is not as many as the instruction access, the amount of constructive hits for the data is non-negligible. Since they are more likely to miss the L2$ if it is not a CMP, these constructive hits contribute to slow down the increase of the L2$ miss rate as the number of active cores increases.

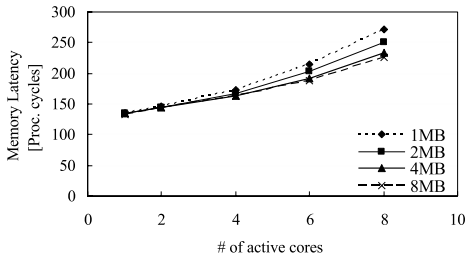
5.2 The Memory Subsystem and Its Effect

Although the rate of increase is slow, the L2$ miss rate multiplied by the number of active cores increases the frequency of memory accesses more or less linearly with the number of active cores. **Figure 14** shows the memory address bus usage and the memory access latency. The memory bus is 54.3% busy with the 6 active cores running with 4 MB L2$. This high busy rate results in longer memory latency, as shown in Fig. 14 (b). This is the major cause of performance roll-off as the number of active cores increases.

To reduce the memory latency degradation due to memory bus congestion caused by accesses from multiple cores, the bandwidth of the memory subsystem is increased by using 266 MHz DDR2 DIMMs (533 Mbit/s data transfer) and having 2 sets of address and data buses. This change is appropriate since the faster data rate of DDR2 allows only 2 DIMMs in a string (where DDR allows 4 DIMMs in a string) and necessitates two sets of address and



(a) Memory address bus busy ratio



(b) Memory access latency

Fig. 14 DDR DRAM memory address bus usage and latency.

Table 3 Memory subsystem parameters.

	DDR model	DDR2 model
Clock	133 MHz	266 MHz
Data Rate	266 MHz	533 MHz
# of DIMMs	4	4
# of Banks	4	4
# of Address buses	1	2
# of Data buses	1× 16 B	2 × 16 B

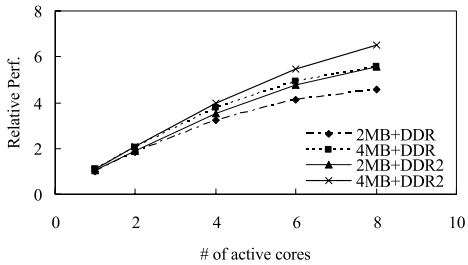
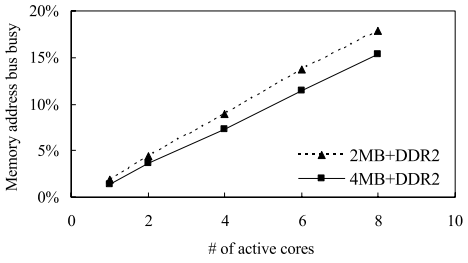


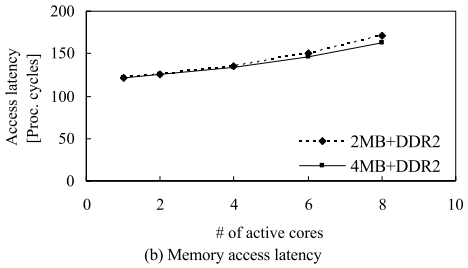
Fig. 15 TPC-C scalability with improved DDR2 memory compared with DDR memory.

memory buses. The additional cost of this configuration is the pins and associated circuits (including the second DLL for DRAM interface) in a CMP chip for the second address and data bus pair.

This 4 times increase in memory bandwidth matches with the 3∼4 times performance increase with the 6∼8 cores. The specification of this memory subsystem is compared against the original DDR in **Table 3**. **Figure 15** shows the performance with the improved DDR2 memory subsystem compared with the DDR memory subsystem. A system with 2 MB L2$ + DDR2 delivers about the same performance as a system with 4 MB L2$ with DDR. The performance of the 4 MB L2$ + DDR2 system exceeds the performance of the 8 MB L2$ system with DDR. **Figure 16** (a) shows the memory address bus busy rate and Fig. 16 (b) shows the memory access latency. The DDR2 memory system with the dual address buses reduces the bus busy time to below 20%, even with 8 active cores, and mitigates the access latency increase.



(a) Memory address bus busy rate



(b) Memory access latency

Fig. 16 DDR2 memory address bus usage.

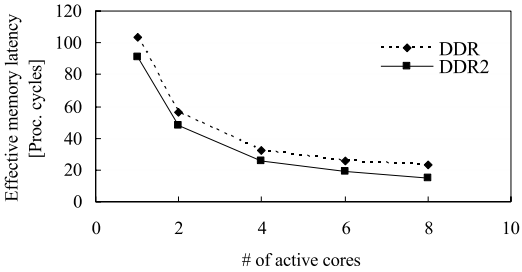


Fig. 17 Effective memory latency. Latency hiding with multiple core execution.

Figure 17 shows the effective memory latency vs. the number of active cores. Effective memory latency is defined as the aggregate L2$miss CPI for all active cores divided by the L2$ miss rate. The effective memory latency of the single core system is about 100 processor cycles, which is 77%(1/1.3) of the physical memory latency shown in Fig. 14 (b) and Fig. 16 (b). This 1.3 coefficient corresponds to the parallel

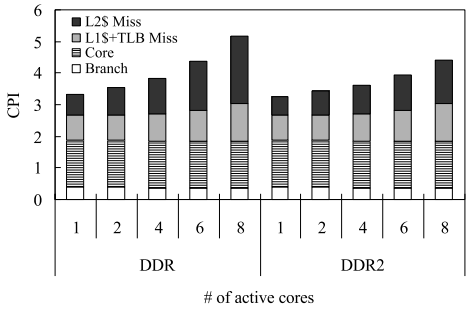


Fig. 18 Instruction execution cycle breakdown (per processor core).

memory accesses obtained from prefetch, speculative execution and out-of-order memory access combined for a single thread execution. A multi-thread executing CMP achieves much higher parallelism in memory access and hides memory latency. The effective memory latency is reduced to less than 30 processor cycles when 6 or more cores are active.

The memory access latency problem is mitigated by the CMP, but it requires more memory bandwidth because multiple cores generate more accesses. This bandwidth problem is solved with the use of two DDR2 channels.

5.3 Analysis of Scalability

Figure 18 shows the breakdown of the instruction execution time (CPI) of a processor core with the DDR and DDR2 memory subsystem. The branch and processor core components remain constant as the number of active cores increases.

The L2$ access congestion and resulting increase in latency was one of the major concern for the CMP. As shown in Fig. 18 (L1$+TLB miss component), the effect of L2$ congestion is noticeable especially for a CMP with 6 or more cores. But, the CPI increase with this effect is small compared to the total CPI, and is acceptable.

The other concern for CMP was memory access congestion and resulting increase in access latency. The effect on CPI is shown as the L2$ miss component in Fig. 18. It is the single largest component of the CPI increase as the number of active cores increases. However, we have found that constructive hits in the L2$ occupy about 60% of the total instruction accesses and 15% of the total data accesses for the 6 core CMP, as shown in Fig. 13 (a) and (b). These constructive hits help to keep the L2$ miss rate relatively flat with the increasing number of cores, as shown in Fig. 12. The memory access component of CPI is reduced with the use of a higher bandwidth memory subsystem using DDR2 memory. The DDR2 memory subsystem reduced the L2$ miss component by 27% and improved total performance by 10% over the same 6 core CMP using the DDR memory subsystem.

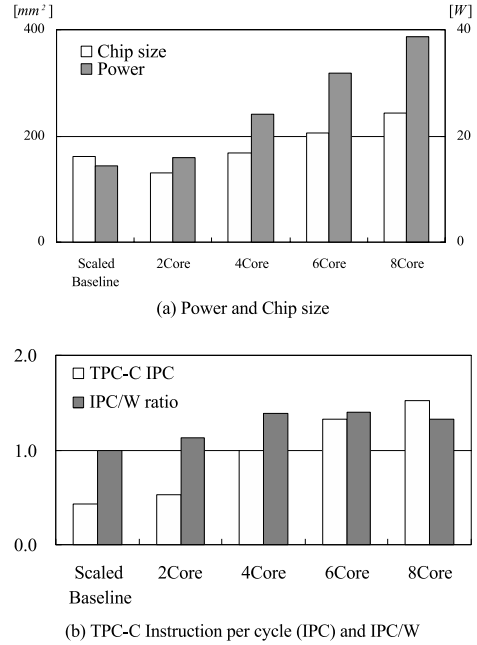


Fig. 19 Small core CMP chip size, power dissipation, TPC-C performance in aggregate IPC and relative IPC/Watt.

6. Energy Efficiency and Comparison to the Scaled Baseline Processor

The silicon technology assumed is Fujitsu's 90 nm CMOS process [17], designed for a nominal Vdd of 1.0 V. But we have reduced the power supply voltage to 0.8 V to reduce power consumption while still running at 1.33 GHz clock, because the 90 nm semiconductor process improves circuit speed over the 130 nm process used for the baseline processor. We compared the instructions per cycle (IPC), power consumption, normalized IPC/W and chip size (in the 90 nm process) as shown in **Fig. 19**. A $MIPS^2/W$ (or IPC^2/W as we compare at the same clock frequency) is a more commonly used metric, but, this energy-delay product metric is easily improved by just replicating the baseline core N times, as the energy per instruction stays constant and the time to completion becomes 1/N. Hence we have chosen IPC/W as our metric. We believe this metric fits with the

nature of transaction processing, which can use more servers to get higher IPC performance.

A *scaled baseline* is the baseline processor scaled into the 90 nm process with its L2$ size increased to 4 MB. 2 to 8 core CMPs in Fig. 19 use small cores with 4 steps of reduction as described in section 4. All configurations use the same DDR2 memory subsystem.

The two core CMP delivers 24% higher aggregate TPC-C performance and consumes 9% higher power compared to the scaled baseline processor. The resulting IPC/W is 13% higher than the scaled baseline. Since the silicon area of the small core is 40% of the baseline core, a two core CMP chip is smaller than the baseline chip.

The four core CMP chip size is about equal to the size of the baseline chip, since the use of custom macros reduces the area of the L2$ unit (TLB CAM and the small register files for data buffers) and compensates for the increase in core area. This design delivers 2.3 times higher TPC-C performance with 67% more power dissipation. It achieves 38% higher IPC/W compared to the scaled baseline chip.

As the number of processor cores on a CMP is increased to 6, IPC/W becomes 40% higher than the baseline. But, it degrades with the 8 cores since the increase in IPC is limited by the capability of the L2$ and memory subsystem. The resulting IPC improvement is smaller than the power dissipation increase from the additional core.

As shown in Fig. 19 (b), the CMP delivers higher IPC/W than the scaled baseline chip. This means that the CMP consumes less power when the same IPC is required. For example, a two core CMP delivers 24% higher IPC than the scaled baseline. For the scaled baseline to match this performance, the clock frequency need to be increased by 24%, and the supply voltage need to be raised to 1.0 V. This increases the power dissipation of the scaled baseline chip by 94% which is much higher than the 2 core CMP power dissipation (only 9% higher than the 0.8 V scaled baseline).

7. Conclusion

We have derived a smaller processor core from a commercial high end processor (SPARC64 V) using a four-step resource reduction procedure. We showed that a CMP built with four smaller cores and with a total area equal to the scaled high-end processor (this gives roughly the same manufacturing cost) has 2.3 times higher TPC-C performance and 38% higher aggregate performance per watt (IPC/W). The power dissipation of the CMP can be significantly lower compared to the single core scaled baseline processor when the higher performance is not required. This design approach results in more efficient processors compared to the current high-end uniprocessors designed for highest single-thread performance.

Processor power dissipation is rapidly increasing with semiconductor scaling. Power and heat removal are becoming critical design issues. The high energy efficiency of a smaller core CMP working on highly parallelizable application like OLTP contribute significantly to the design of compact servers that can be powered and cooled efficiently.

A smaller processor core requires less design effort than a large processor core. Although this benefit was not evaluated in this case study, it also can run at higher clock frequency since the wires are shorter and fan-outs are less. These benefits make the proposed design approach more attractive from engineering cost, competitiveness, and time-to-market viewpoints.

Acknowledgments The authors acknowledge the support and encouragement from Takashi Aoki and Yasunori Kimura for conducting this study. This study would not have been possible without the processor design data and the discussion with the SPARC64 V processor design team headed by Aiichiro Inoue.

References

1) Hammond, L., et al.: A Single-Chip Multiprocessor, *IEEE Computer*, Vol.30, No.9, pp.79–85 (1997).
2) Barrosso, L.A., et al.: Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, *Proc. 27th International Symposium on Computer Architecture*, pp.282–293 (2000).
3) Hammond, L., et al.: The Stanford Hydra CMP, *IEEE Micro*, Vol.20, No.2, pp.71–84 (2000).
4) Codrescu, L., Wills, D. and Meindl, J.: Architecture of the Atlas Chip-Multiprocessor: Dynamically Parallelizing Irregular Applications, *IEEE Trans. Comput.*, Vol.50, No.1, pp.67–82 (2001).
5) Nayfeh, B., Hammond, L. and Olukotun, K.: Evaluation of Design Alternatives for a Multiprocessor Microrocessor, *Proc. 23rd Annual International Symposium on Computer Archi-*

tecture, pp.66–77 (1996).
6) Tendler, J.M., et al.: POWER4 system microarchitecture, *IBM Journal of R.& D.*, Vol.46, No.1, pp.5–25 (2002).
7) Alameldeen, A.R., et al.: Evaluating Non-deterministic Multi-threaded Commercial Workloads, *Proc. Fifth Workshop on Computer Architecture Evaluation using Commercial Workloads* (2002).
8) Inoue, A.: Fujitsu's New SPARC64 V for Mission-Critical Servers, *Microprocessor Forum 2002* (2002).
9) Ando, H., et al.: A 1.3 GHz Fifth Generation SPARC64 Microprocessor, *IEEE Journal of Solid-State Circuits*, Vol.38, No.11, pp.1896–1905 (2003).
10) Pollack, F.: New Microarchitecture Challenges in the Coming Generations of CMOS, *Keynote of the 32nd Annual International Symposium on Microarchitecture* (*MICRO-32*) (1999).
11) Codrescu, L., et al.: Exploring Microprocessor Architectures for Gigascale Integration, *Proc. 20th Anniversary Conference on Advanced research in VLSI*, pp.242–245 (1999).
12) Cmlick, R. and Keppel, D.: Shade: A Fast Instruction-set Simulator for Execution Profiling, Technical Reports TR-93-12, Sun Labs (1993).
13) Barrosso, L.A., et al.: Memory System Characterization of Commercial Workloads, *Proc. 25th Annual International Symposium on Computer Architecture*, pp.3–14 (1998).
14) Nanda, A.K.: Multiprocessor Architecture Evaluation using Commercial Applications, *Proc. First Workshop on Computer Architecture Evaluation using Commercial Workloads* (*HPCA-4*) (1998).
15) Sakamoto, M., et al.: Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems, *Proc. Ninth International Symposium on High-Performance Computer Architecture* (*HPCA-9*), pp.141–152 (2003).
16) Ranganathan, P., et al.: Performance of Database Workloads on Shared-Memory Systems with Out-of-Order Processors, *Proc. ASPLOS* (1998).
17) Nakai, S., et al.: A 100 nm CMOS Technology with "Sidewall-Notched" 40 nm Transistors and SiC-Capped Cu/VLK Interconnects for High Performance Microporcessor Applications, *Symposium for VLSI Technology Digest Papers*, pp.66–67 (2002).

(Received September 24, 2004)
(Accepted January 17, 2005)

Hisashige Ando received B.S. and M.S. degrees in electronic engineering from Tokyo Institute of Technology, Tokyo, Japan in 1968 and 1970 respectively. He joined Fujitsu, Kawasaki, Japan in 1970. He has been working on the development of high-end processors. He was assigned to HAL computer systems between 1992 and 1998 and led the development of first 3 generations of SPARC64 processors. Currently, he is a chief technical officer of Server systems group of Fujitsu. He is currently interested in the application of CMP for energy efficient systems and high performance computing systems.

Akira Asato received the B.S. degree in information science from the University of Tokyo, Tokyo, Japan in 1983. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1983 and has been engaged in research and development of computer architectures. He was a secretary of SIG-Architecture of IPSJ between 2000 and 2004, and has been a member of the Editorial Board of IPSJ Transactions on Advanced Computing Systems (ACS) since 2001. He is a member of the Information Processing Society of Japan (IPSJ).

Motoyuki Kawaba received the M.E. degree in Information Engineering from the University of Tokyo in 1991. In 1991, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan. His research interests include performance modeling and analysis of commercial workloads.

Hideki Okawara received the M.E. degree in Mathematical Engineering and Information Physics from the University of Tokyo in 2000. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 2000. His research interests include computer system architecture, evaluation and modeling of commercial system. He is a member of the Information Processing Society of Japan (IPSJ).

William Walker received the A.B. degree in physics and applied math in 1976, and the MSEE in 1978, both from the University of California at Berkeley. From 1978 to 1981 he was a Staff Engineer at IBM Corporation, East Fishkill, NY and from 1981–1983 at IBM in Burlington VT. At IBM he was involved in the development of the LDD MOS transistor. From 1984 to 1991 he was a Senior Engineer at Integrated CMOS Systems, Inc. in Sunnyvale CA. From 1991 to 2000, he was an Engineering Manager at Hal Computer Systems, Inc. in Campbell, CA where he developed circuits for the first 64-bit SPARC microprocessors. Since 2000, he has been employed at Fujitsu Laboratories of America, where he is currently Vice President in charge of the LSI Technology Research Laboratory. His research interests include high-speed and low-power digital circuits for microprocessors and radio-frequency CMOS circuits for telecommunications.