Exploring Core Designs for Chip Multiprocessors

Allison Holloway (ahollowa@cs.wisc.edu)
Matthew Allen (matthew@cs.wisc.edu)
CS838 Project Report

Abstract

The era of billion transistor chips is fast approaching, and the emerging trend is to use these transistors to integrate multiple processors onto a single chip. In this paper, we explore the core design for a chip multiprocessor (CMP). We have found that out-of-order cores provide better absolute performance than in-order cores, both for commercial and scientific workloads. In general, it takes twice as many in-order cores to equal the performance of out-of-order cores. However, when we consider the die area required for each type of core, we find that in-order cores perform twice as well as out-of-order cores in half the space. Therefore, we believe future CMPs should use in-order cores.

1. Introduction

As technology advances, computer architects and chip designers are able to put ever-more features on a single chip. We are now reaching the point where we can fit multiple processing cores onto a single die. Since we can now include multiple cores, the question of how these cores should look arises, namely should they be really simple in-order cores, really complex out-of-order cores, or somewhere in-between? We investigate both two and four wide in-order and out-of-order processors, and we also vary the number of available outstanding memory instructions for in-order cores. We believe that allowing a larger number of outstanding memory operations will reduce the performance gap between in-order and out-of-order cores.

We have decided to focus on commercial workloads, with a few scientific applications. Conventional wisdom on these applications is that they have little instruction-level-parallelism

and large memory footprints [2]. Hence, one of our hypotheses is that commercial workloads will not benefit from the extra complexity and size of out-of-order, wide-issue cores. The conventional wisdom seems to be the opposite for many scientific codes: that they have a lot of ILP and smaller footprints. Our second hypothesis is that the scientific workloads will greatly benefit from out-of-order, wide issue cores.

We realize that thinking of returning to in-order cores seems to buck the current microprocessor trend; however, it generally makes sense to re-evaluate your assumptions frequently, especially when there is a revolutionary leap forward in technological capabilities, as in the case with chip multiprocessors. Another important issue that will need to be addressed is how many cores the CMP should have. This question brings about an interesting trade-off: is it better to have more in-order cores or fewer out-of-order cores? Our instincts suggest the former, and we hypothesize that as we add more processors, the performance advantage of out-of-order processing will diminish.

The idea of having a CMP has been around for a number of years, so we will discuss some related work. Then, we will provide our methodology for testing our hypotheses and present our results. Finally, we will give our conclusions and suggest what we believe would be most appropriate for a future CMP.

2. Related Work

Many academic and industrial researchers have proposed CMP designs. One of the first CMPs was Multiscalar [8]. Multiscalar executed a single thread (broken into several *speculative* threads), so it differs from a general multiprocessor on a chip that executes independent threads.

One of the first large-scale research projects to look into SMPs on a chip was Hydra [7, 3, 6]. The Hydra group did a lot of work to make the case for CMPs and evaluate different cache alternatives, but they have not presented a side-by-side analysis of in-order versus out-of-order cores and did an area analysis to determine the optimal configuration.

Compaq's Western Research Lab has also investigated CMPs, but again none of the work directly compares in-order and out-of-order cores to figure out the optimal configuration [2, 1]. Some of the work finds that integrating different features affects in-order processors as much as it affects out-of-order processors, but does not directly compare them [2]. Their Piranha CMP chip uses eight in-order cores but does not provide extensive justification for their use [1].

One of the few general-purpose CMPs in production is IBM's Power4 [9, 12]. The Power4 uses two complex out-of-order 8-wide cores, however they give no analysis of why they chose two out-of-order cores rather than more in-orders, except to say they want high throughput.

The final related work we will mention is *Exploring the Design Space of Future CMPs* [4]. This paper includes an analysis of in-order versus out-of-order cores and decides that out-of-order

cores perform better and are more efficient. However, there are some caveats. Their CMP is designed so that only the interconnection network and any off-chip resources are shared: each processor has its own L1 and L2 caches. In addition, they focus on independent single-threaded SPEC benchmarks without operating system support in their simulator. To evaluate the SPEC benchmarks in the context of a CMP they just have each processor run its own copy of the same benchmark. In general, their methodology and design assumptions are completely different from ours. However, we use their cache byte equivalent area analysis to determine our area tradeoffs.

3. Methodology

3.1 Simulation Infrastructure

Our simulations were performed using the Multifacet simulator. This simulator is based on the Virtutech Simics [11] full-system simulator. Multifacet augments Simics with Ruby, a detailed memory system model, and Opal [5], a cycle-accurate timing model of an out-of-order superscalar processor.

Since the Multifacet simulator does not have an in-order processor timing model, we needed to build one for the project. We modified Opal to create a version that issues all instructions in order. We also looked at using the Simics functional simulator without Opal, which would give us a 1 IPC machine, with Ruby providing detailed timing for the memory system. We found the results to be erratic—they were sometimes better than our best Opal configuration, and sometimes worse than our worst Opal configuration. We conclude that the Simics functional simulator is not comparable with a detailed processor model, and omit the results from this paper.

Our modifications to Opal model a processor that issues instructions in-order, but retires them out-of-order. This prevents long-latency instructions from delaying later instructions after they have issued. To restrict Opal to in-order issue, we modified the way dynamic instructions are handled. Dynamic instructions go through several stages before they are issued. Each stage represents an input to the instruction, and the instruction may only proceed to the next stage when that input is ready. We added another stage before issue so that once all operands are ready, an instruction waits in the new stage until the instruction preceding it in the instruction window begins execution. We also had to modify several other parts of the simulator to make this work correctly, particularly the parts dealing with squashing mis-speculated instructions.

There are a few shortcomings to our in-order timing model. Due to the complications caused by the register windows in the SPARC architecture, the Opal code that handles register renaming is very complex, and we did not have time to remove it from our model. However, given that issue is restricted to in-order, we do not believe that register renaming gives our timing model much of an advantage over a more realistic in-order implementation.

Recent implementations of in-order processors, such as the Ultra SPARC [10], rely on sophisticated compiler scheduling to maximize utilization of superscalar resources. For example, the Ultra SPARC can issue branches and stores in the same cycle as instructions they depend on, because branches and stores are resolved later in the pipeline than the execution of other instructions. We did not have access to the source code for most of our workloads, so we were not able to recompile them. It would be impractical to make the substantial modifications

to the simulator necessary to make it match the issue rules assumed in our workloads, so we did not consider that option.

3.2 Baseline Configuration

Table 1 gives the baseline configuration used for our simulations.

Issue Policy	In-order, Out-of-order
Issue Width	2, 4
Pipeline Depth	10
Integer Registers	160 logical + 64 rename
Floating Point Register	64 logical + 128 rename
Instruction Window Size	32 (2-wide), 64 (4-wide)
ROB Size	64 (2-wide), 128 (4-wide)
L1 Cache	64KB I/D
L2 Cache	2MB shared

Table 1: Baseline processor configuration for simulations.

3.3 Workloads

Our benchmarks consist of four commercial workloads and two scientific benchmarks. Our commercial benchmarks are Apache (1,000 transactions), SPECjbb (10,000 transactions), OLTP (50 transactions), and Zeus (1,000 transactions). Our scientific benchmarks are Barnes-Hut (16,000 problem size) and Ocean (258 problem size).

4. Results

Figure 1 summarizes the performance of all benchmarks vs. the number and configuration of cores on the CMP. We see that out-of-order issue provides more benefit (about 25%) than increasing the issue width from two to four. We note that for the in-order models we see little benefit (2-5%) from increasing the issue width, while out-of-order models benefit more (about 10%). We believe this is due to our lack of aggressive compiler scheduling for the in-order model.

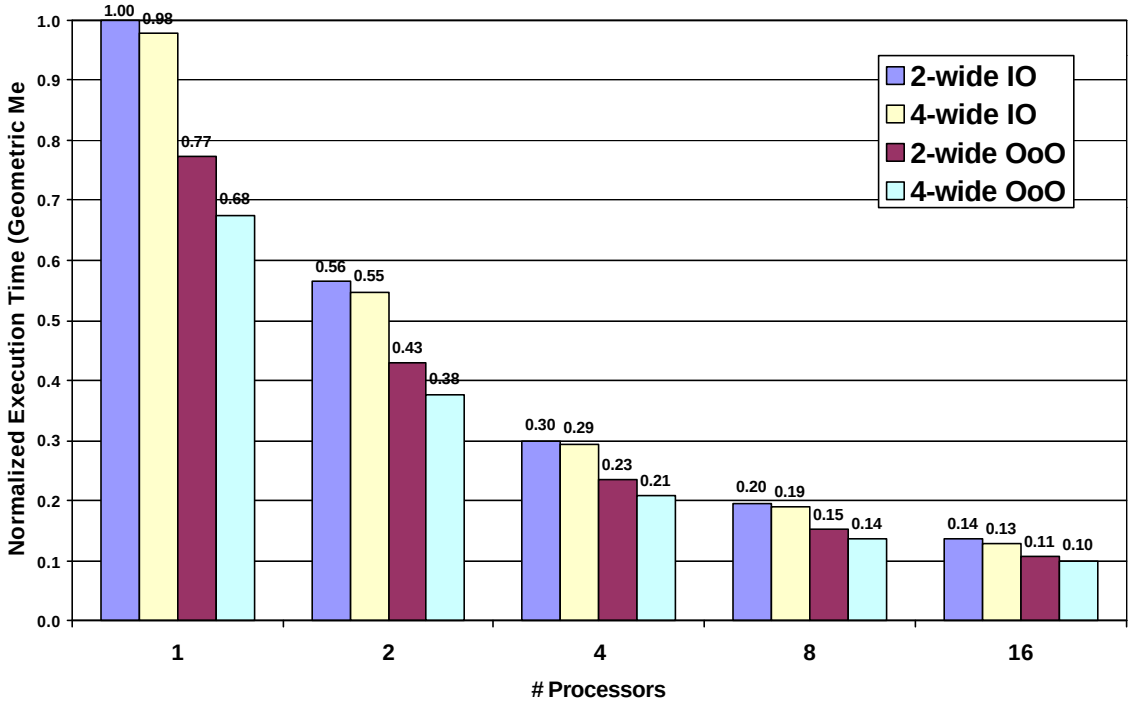

Figure 1: Geometric mean of normalized execution time of all benchmarks vs. number of processors by processor configuration.

4.1 Commercial vs. Scientific Workloads

We hypothesized that commercial workloads would receive little to no benefit from out-of-order cores due to their lack of ILP. We also hypothesized that scientific workloads would receive significant benefits from out-of-order issue. Figure 2 shows the results of our experiment.

Commercial workloads receive an 18-24% improvement from out-of-order issue. Scientific workloads benefit more—25-42%. Thus, commercial workloads do benefit from out-of-order cores, but less so than scientific workloads.

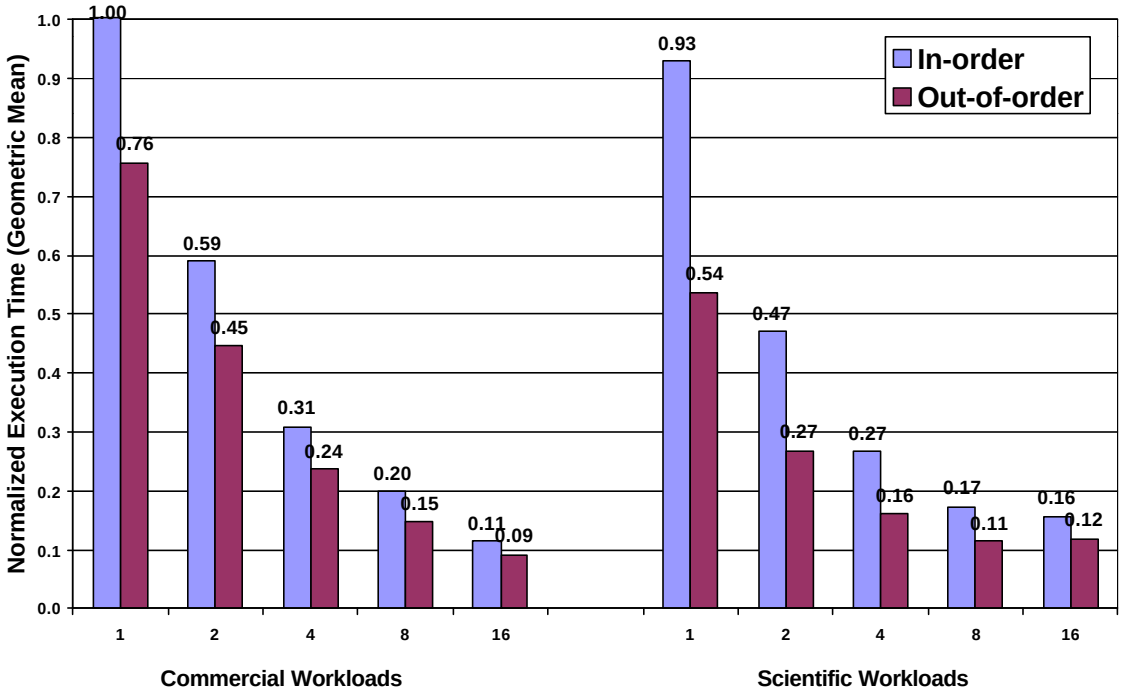

Figure 2: Geometric mean of normalized execution time
for a 4-wide core, by processor configuration and workload type.

4.2 Scale in a Chip Multiprocessor

Our third hypothesis was that as the number of processors in the CMP increases, the performance advantage of out-of-order cores would decrease. We thought that the increased coherence traffic in a larger-scale system would diminish the out-of-order processor's ability to tolerate latency. Figure 3 is a log-log graph that shows the scaling of our workloads vs. number of processors. We see that overall the workloads scale very well as the number of processors increases. We also see that the performance advantages of out-of-order issue and larger issue width are basically constant as the number of processors in the system grows.

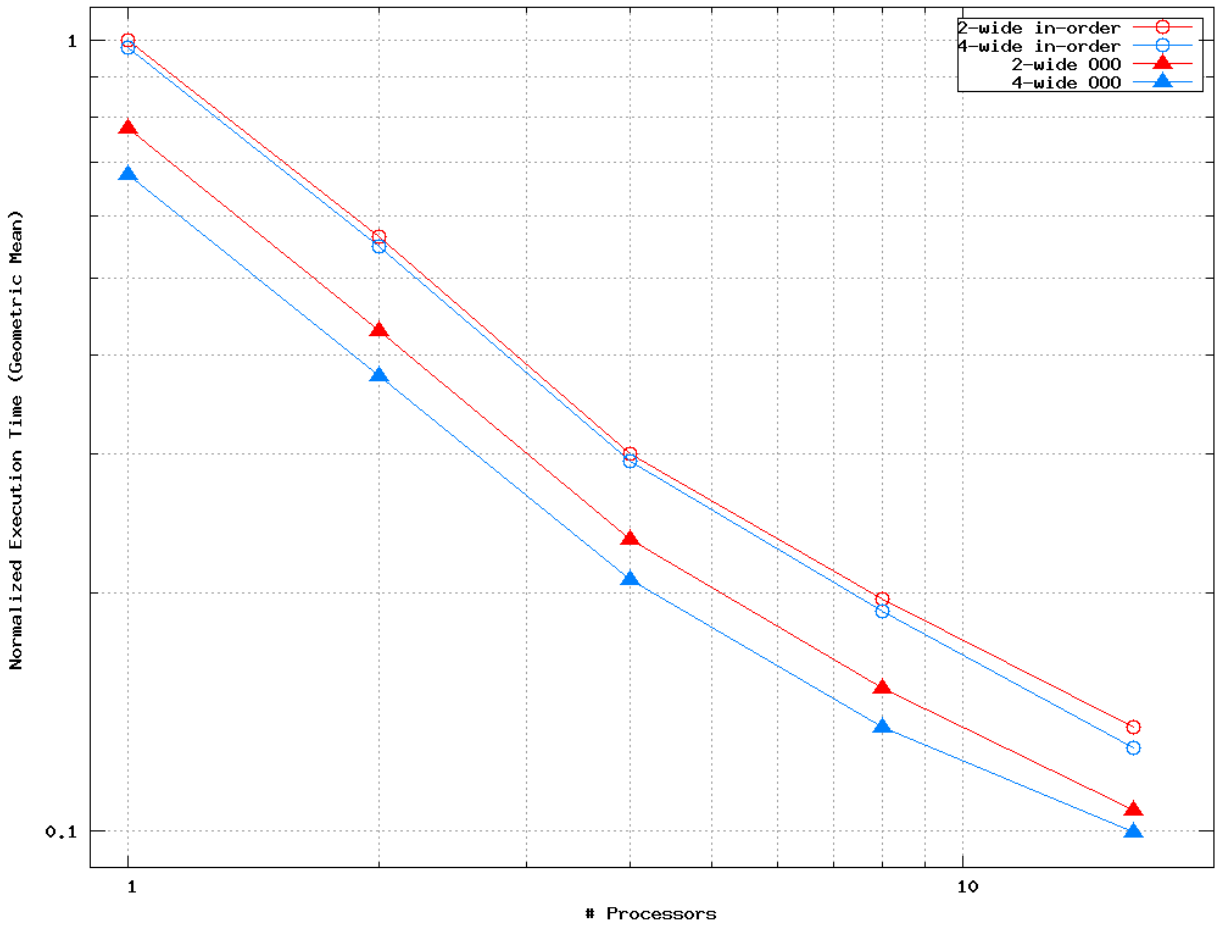

Figure 3: Geometric mean of normalized execution times vs. number of processors in the CMP.

We also examined the scaling of each workload separately. All benchmarks except for Ocean scaled well, as shown in Figure 3. Ocean performed best with eight processors, and actually took longer to execute with sixteen processors (Figure 4). This may indicate that our L2 cache was too small, causing destructive interference among the threads, or it could indicate that the application is not structured to take advantage of a larger number of processors.

Another interesting observation that can be made from Figure 3 is a comparison of execution times for different configurations of processors. For instance, 4 out-of-order cores have roughly

the same performance as 8 in-order cores. In terms of absolute performance, we can conclude that out-of-order cores are better; however, Section 4.4 re-examines this issue, taking area into account.

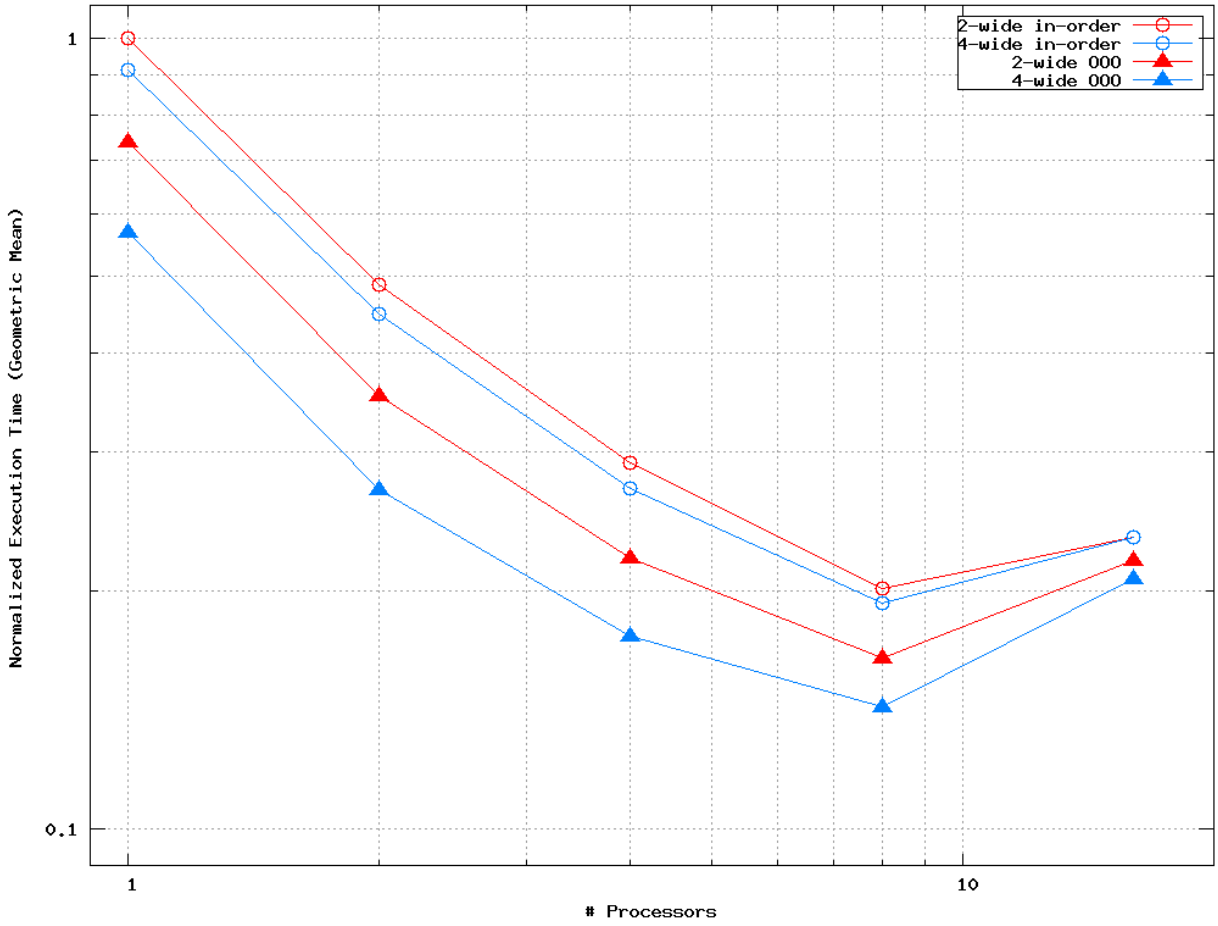

Figure 4: Scaling of the Ocean benchmark.

4.3 Number of Outstanding Memory Requests

Another parameter we investigated was the number of simultaneous outstanding memory requests a processor could have. Because our in-order model can complete instructions out-of-

order, it may benefit from having multiple outstanding requests to the memory hierarchy (for reference, the Ultra SPARC-I processor allows five outstanding requests [10]).

Figure 5 shows average performance for all workloads on our 4-wide in-order processor, varying the allowed number of outstanding memory requests. We see a significant improvement of 14% from increasing the number of requests from one to two, and little benefit from allowing more than two requests.

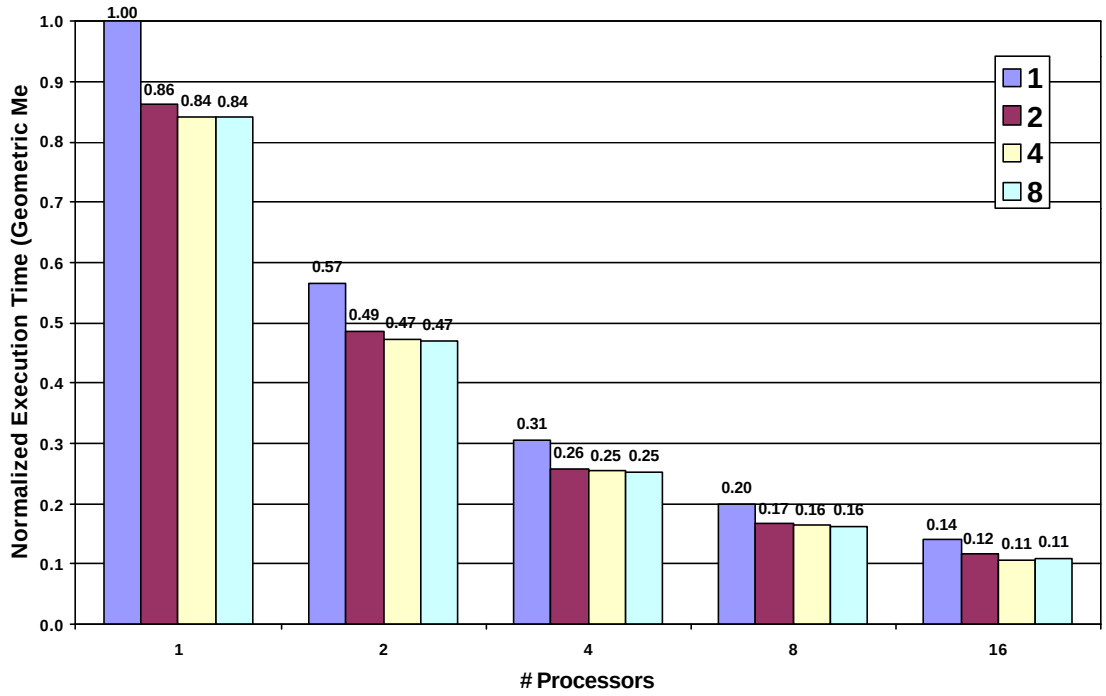

Figure 5: Performance of 4-wide in-order processor by number of allowed outstanding memory requests.

4.4 Area Efficiency of Cores in a CMP

We have seen that out-of-order processors provide a significant performance advantage over in-order cores. However, in-order cores require less space on a die, and therefore a chip multiprocessor may achieve higher performance by using a larger number of in-order cores. To

determine the trade-off between core complexity and die area, we used area estimates published in an earlier design space study of CMP cores [4] to evaluate the performance trade-off with die size.

Huh et al. [4] normalize area to the amount consumed by a byte of SRAM cache. They estimate that a 2-wide in-order core consumes 50 KB cache-byte-equivalents (CBE) and that a 4-wide out-of-order core consumes 250 KB CBE. Combining these estimates with our performance data, we were able to plot performance vs. die size, shown in Figure 6.

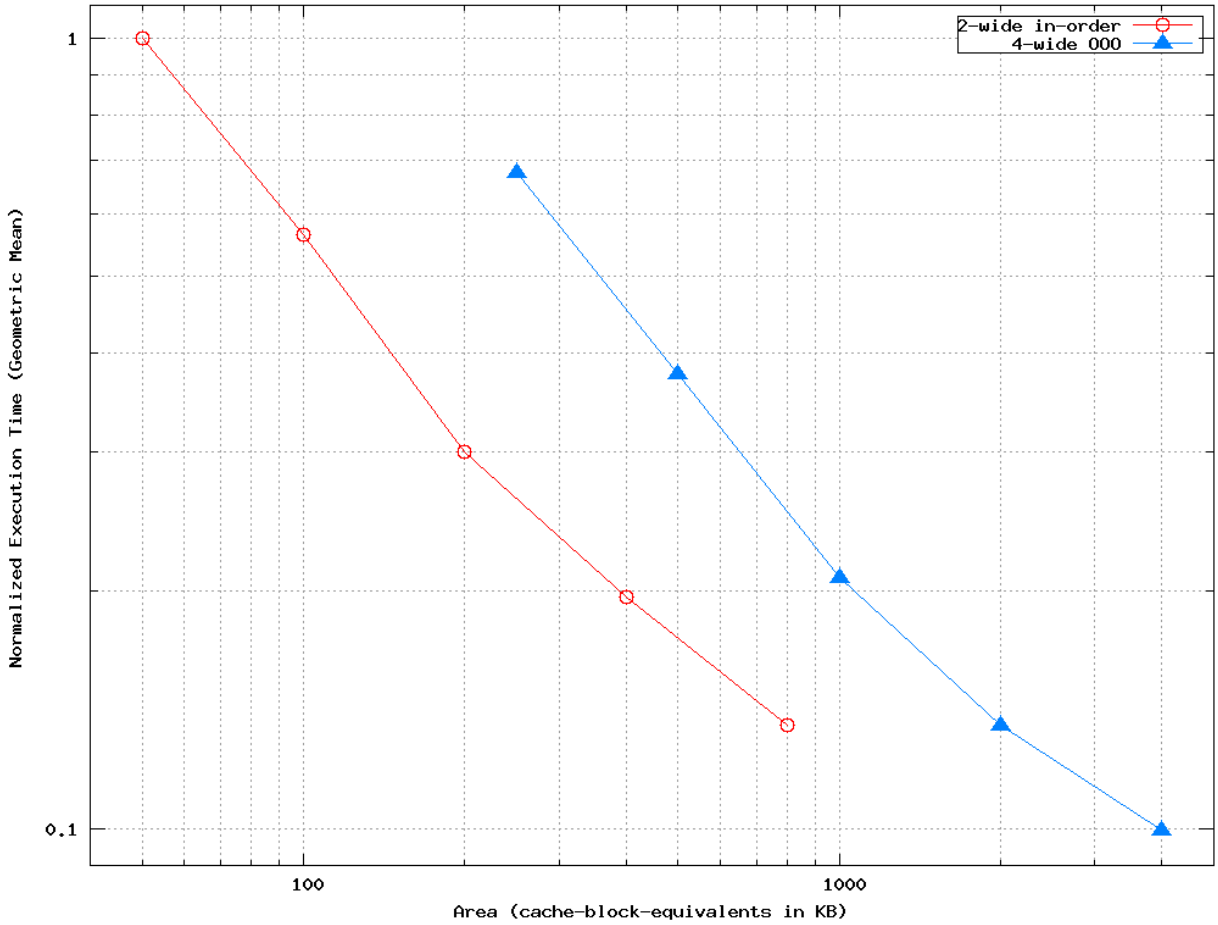

Figure 6: Performance vs. area for simple and complex CMP cores, on a log-log scale.

Figure 6 shows that for a given die area, a CMP using simple 2-wide in-order cores will perform significantly better than one using complex 4-wide out-of-order cores. The figure shows that two in-order cores provide about the same performance as one out-of-order core in roughly half the area, and that this performance difference holds true as the number of cores on the CMP increases.

5. Conclusions

Our measurements have shown that out-of-order processor cores provide significant performance advantages over in-order cores: 18-24% for commercial workloads and 25-42% for scientific workloads. This performance advantage is seen for all system sizes we examined, up to 16 processors.

All the workloads we measures scaled well with increasing number of processors, except for Ocean. Ocean performed best with 8 processors, and performance degraded in the 16 processor configuration.

For in-order cores, we found that allowing two simultaneous requests to the memory system improved performance by 14%. Allowing more simultaneous requests yielded only marginal improvement.

Although out-of-order cores provide better performance than in-order cores, this does not mean that they are better for chip multiprocessors. Out-of-order cores consume more power and area, and are much more complex than in-order cores. We must account for this complexity when

making our design decisions. We evaluated the performance-area tradeoff and found that two in-order cores provided roughly the same performance as one in-order core in half the area. This rule of thumb holds for all system sizes we tested, so a 16 in-order processor CMP would perform as well as an 8 out-of-order processor CMP in half the area.

In conclusion, we have shown that wide-issue out-of-order superscalar cores that perform well for uniprocessor workloads are not the best fit for CMPs running parallel workloads. For a fixed die size, in-order processors provide better performance and less complexity.

References

[1] L.A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing. In *Proceedings of the 27th ACM International Symposium on Computer Architecture*, June 2000.

[2] L.A. Barroso, K. Gharachorloo, A. Nowatzyk, B. Verghese. Impact of Chip-Level Integration on Performance of OLTP Workloads. In *Proceedings of the Sixth International Symposium on High-Performance Computer Architecture (HPCA-6)*, January 2000, Toulouse, France

[3] L. Hammond, B.A. Nayfeh and K. Olukotun. A Single-Chip Multiprocessor. *IEEE Computer* Special Issue on "Billion-Transistor Processors," September 1997.

[4] J. Huh, D.C. Burger, and S.W. Keckler. Exploring the Design Space of Future CMPs. In *International Conference on Parallel Architectures and Compilation Techniques* (PACT), September, 2001.

[5] C. J. Mauer, M. D. Hill, D. A. Wood: Full System Timing-First Simulation. In *Proceedings of the 2002 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, June 2002.

[6] B.A. Nayfeh , L. Hammond and K. Olukotun. Evaluation of Design Alternatives for a Multiprocessor Microprocessor. In *Proceedings of the 23rd International Symposium on Computer Architecture*, May 1996.

[7] K. Olukotun, B.A. Nayfeh, L. Hammond, K. Wilson and K.Y. Chang. The Case for a Single-Chip Multiprocessor. In *Proceedings of the Seventh International Symposium on Architectural Support for Parallel Languages and Operating Systems*, October 1996.

[8] G. S. Sohi, S. Breach, and T. N. Vijaykumar. Multiscalar Processors. *In Proceedings of the* 22nd International Symposium on Computer Architecture (ISCA-22), 1995.

[9] J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy. POWER4 system microarchitecture. *IBM J. of Research and Development*, vol. 46, no., pp 5.

[10] M. Tremblay and J.M. O'Connor; *UltraSparc I: A Four-Issue Processor Supporting Multimedia*: IEEE Micro, Vol. 16, No. 2, March-April 1996; Pages 42-50.

[11] Virtutech Simics AB: Simics Full-System Simulator. http://www.simics.com

[12] J. D. Warnock, J. M. Keaty, J. Petrovick, J. G. Clabes, C. J. Kircher, B. L. Krauter, P. J. Restle, B. A. Zoric, and C. J. Anderson. The circuit and physical design of the POWER4 microprocessor. *IBM J. of Research and Development*, vol. 46, no., pp 27.