# Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors[1]

Tomer Y. Morad† Uri C. Weiser‡ Avinoam Kolodny† Mateo Valero* Eduard Ayguadé*

†Department of Electrical Engineering
Technion, Haifa, Israel
{tomerm@tx , kolodny@ee}.technion.ac.il

‡Intel Corporation
Petach Tikva, Israel
uri.weiser@intel.com

*Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya, Barcelona, Spain
{mateo, eduard}@ac.upc.es

*Abstract*—**This paper evaluates asymmetric cluster chip multiprocessor (ACCMP) architectures as a mechanism to achieve the highest performance for a given power budget. ACCMPs execute serial phases of multithreaded programs on large high-performance cores whereas parallel phases are executed on a mix of large and many small simple cores. Theoretical analysis reveals a performance upper bound for symmetric multiprocessors, which is surpassed by asymmetric configurations at certain power ranges. Our emulations show that asymmetric multiprocessors can reduce power consumption by more than two thirds with similar performance compared to symmetric multiprocessors.**

*Index Terms*—**ACCMP, Chip Multiprocessors, Power Efficiency.**

## I. Introduction

Achieving high performance within a given power envelope is a major concern for microprocessor architects. In the past, the constant decrease in feature sizes has enabled to increase uniprocessor performance by packing more transistors in the same die area. Nowadays, due to the complexity of current state of the art microprocessors, a large increase in power and area results in only small performance improvements. Empirical evidence suggests that the performance of uniprocessors is proportional to the square root of their area [12],[5]. These trends favor the use of Chip Multi Processors (CMP) [8].

Software applications contain serial phases as well as parallel phases. The lowest execution time for the serial phases is achieved by the highest performance uniprocessor available, since the serial phases can be executed on one processor only. On the other hand, parallel phases can be executed on numerous processors in parallel. Therefore, the lowest execution time for the parallel phases is achieved by executing them on many simple processors that consume less energy per instruction (EPI) [7]. We claim that a choice of symmetric cores is suboptimal due to the contradicting requirements of the serial and parallel phases within the same application.

We propose placing clusters of different cores on a single die. All of the cores within the Asymmetric Cluster Chip Multi-Processor (ACCMP) share the same instruction set architecture and memory address space in order to let threads migrate from core to core. The larger and faster cores will execute single-threaded programs and the serial phases of multithreaded programs for high EPI, whereas the smaller and slower cores will execute the parallel phases for lower EPI. In the general case, ACCMPs will include clusters of different cores, since numerous multithreaded applications with varying parallelism execute in parallel. An example of an ACCMP with three clusters is shown in Fig. 1.

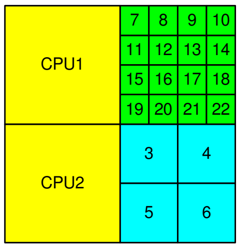

Fig. 1. An ACCMP floorplan with 22 general-purpose cores. On the left is a cluster of large high performance cores that consume high EPI. On the right are two clusters of smaller cores that consume less EPI.

Placing heterogeneous cores on a single die has been proposed in recent research. Kumar et al. [9] have shown how a heterogeneous multiprocessor could achieve similar performance to a homogeneous multiprocessor for less power and area. Grochowski et al. [2], [7] have proposed and demonstrated an asymmetric multiprocessor by employing voltage and frequency scaling on a symmetric multiprocessor. Menasce et al. [10] have shown the analytic benefit of heterogeneous systems using queuing models. Moncrieff et al. [11] have shown that heterogeneous multiprocessors perform better when parallelism varies during execution.

Our contributions to previous work are as follows. We propose a simple theoretical model for the performance of symmetric chip multiprocessors. We then find an upper bound for the performance per unit power (power efficiency) of such multiprocessors. By comparing this upper bound to the power efficiency of asymmetric chip multiprocessors we conclude that asymmetric structures can achieve higher power efficiency than any symmetric chip multiprocessor. Finally, we validate our findings by experimental emulation of ACCMP structures.

## II. Analysis

### A. *Symmetric Chip Multiprocessors (SCMP)*

Software applications have two types of phases, serial phases and parallel phases. The serial phases can be executed

[1] The work has been performed under the Project HPC-EUROPA (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme. This work was also supported by a grant from Intel Corporation. We acknowledge the European Center for Parallelism of Barcelona (CEPBA) for supplying the computing resources for this research.

Manuscript received: 27 Apr. 2005. Manuscript accepted: 14 July 2005. Final manuscript received: 22 July 2005.

only on a single processor whereas the parallel phases can be executed on more than one processor. The variable λ denotes the fraction of dynamic instructions that reside in the parallel phases out of the total dynamic instructions.

For the purpose of mathematical analysis, we evaluate multithreaded applications whose parallel phases can be executed by up to the number of available hardware processors, denoted by *n*. We assume that the performance of a uniprocessor is a function of its area [12], [5]. Therefore, a processor with area size *a* will have performance of Perf(*a*), measured in instructions per second. We assume constant cycles per instruction per processor throughout the workload.

The number of dynamic instructions in the parallel phases of a program with *M* total dynamic instructions is *λM*. When interactions between cores are neglected, Amdahl's law for symmetric parallel processors [1] may be written as following:

$$\mathrm{T}_{SCMP} = \mathrm{T}_{serial} + \mathrm{T}_{parallel} = M\left(\frac{(1-\lambda)}{\mathrm{Perf}(a)} + \frac{1}{n}\cdot\frac{\lambda}{\mathrm{Perf}(a)}\right). \tag{1}$$

The performance of an SCMP can thus be expressed as

$$\mathrm{Perf}_{\mathrm{SCMP}} = \frac{M}{\mathrm{T}_{SCMP}} = \mathrm{Perf}(a)\frac{1}{\lambda/n + (1-\lambda)}. \tag{2}$$

We estimate Perf(*a*) for a uniprocessor of size *a* by incorporating Pollack's rule [12], [5] which states that performance grows with the square root of area, that is

$$\mathrm{Perf}(a) = \eta\sqrt{a}, \tag{3}$$

where $\eta$ is a constant. For simplicity, we assume that total chip power is proportional to the area of the die [12],

$$P = \gamma n a. \tag{4}$$

SCMP performance (2) can also be expressed as a function of total chip power and area of a single core by using (3)-(4),

$$\mathrm{Perf}_{\mathrm{SCMP}}(P,a) = \mathrm{Perf}(a)\frac{1}{\lambda\left(\frac{\gamma a}{P}\right) + (1-\lambda)}. \tag{5}$$

## B. Performance Upper Bound of Symmetric CMP

For a given power budget, the highest performance ($\partial\,\mathrm{Perf}_{\mathrm{SCMP}}(P,a)/\partial a = 0$) of SCMPs is achieved when

$$n_{opt} = \frac{\lambda}{1-\lambda}, \quad a_{opt} = \frac{P}{\gamma n_{opt}}. \tag{6}$$

The above results imply that maximum performance for symmetric multiprocessors is achieved when the execution time of the parallel and serial phases are equal. The maximum performance as a function of total SCMP power is thus:

$$\mathrm{Perf}_{\mathrm{SCMP,max}}(P) = \mathrm{Perf}\left(a_{opt}(P)\right)\frac{1}{2(1-\lambda)}. \tag{7}$$

## C. Asymmetric Cluster Chip Multi-Processors (ACCMP)

We focus on a special case of ACCMP that contains one large core of area *βa* and *n-1* small cores of area *a*, where *β*>1. We assume that the serial phases of programs are executed on the large core, whereas the parallel phases are executed on all *n* available (large and small) cores. The performance of the ACCMP is thus:

$$\mathrm{Perf}_{ACCMP} = \mathrm{Perf}(\beta a)\frac{1}{\lambda\left(1+\frac{n-1}{\sqrt{\beta}}\right)^{-1} + (1-\lambda)}. \tag{8}$$

The performance of ACCMP can also be expressed as a function of total chip power by using

$$P = \gamma a(n-1+\beta). \tag{9}$$

Fig. 2 shows the performance versus power of an ACCMP compared with SCMPs for *λ*=0.75. Power and performance increase in Fig. 2 as additional cores are added to the system. However, performance saturates when the core count is high due to Amdahl's law. The shown ACCMP achieves higher performance than the symmetric upper bound given by (7) at a certain power envelope.

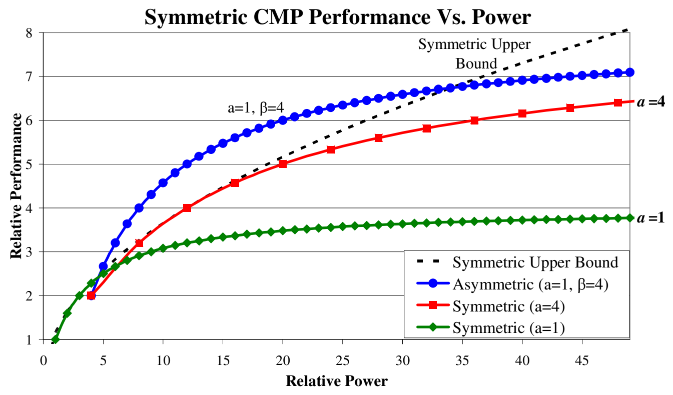

Fig. 2. The performance versus power for SCMP and ACCMP for *λ*=0.75, with no interaction overheads. Each data point represents a multiprocessor with a different number of cores. The graph shows an ACCMP that crosses the upper bound of symmetric multiprocessors at a certain power envelope.

## D. Overhead Model for Symmetric CMP and ACCMP

The interactions between multiple parallel processors incur performance overheads. These overheads are a result of synchronization, communication and coherence costs, and are modeled as a time penalty [6],

$$T_{SCMP} = T_{serial} + T_{parallel} + T_{SCMP,overhead}. \tag{10}$$

In the general case, interaction overheads are a function of the number of processors in the system [6],

$$T_{SCMP,overhead} \propto \left(k_1 + k_2 n + k_3 n^2 + \cdots\right). \tag{11}$$

For our analysis, we assume that all coefficients except $k_1$ and $k_2$ are zero. The coefficient $k_1$ could account for the effects of manipulations to shared locks that are proportional to the problem size. The coefficient $k_2$ could account for the effects of shared lock manipulations required for workload distribution among the *n* available processors. The coefficients are measured in interactions per instruction.

The time required to complete an interaction between two processors depends on the distance information needs to travel on the die. We assume that this distance is roughly proportional to $\sqrt{na}$, which is the diameter of the die. Therefore, the latency is proportional to $v^{-1}\sqrt{na}$, where *v* is the effective signal propagation velocity. The time penalty due to the interaction overheads for SCMP is thus:

$$T_{SCMP,overhead} = \lambda M\left(k_1 + nk_2\right)v^{-1}\sqrt{na}. \quad (12)$$

Therefore, the performance of SCMPs is

$$\text{Perf}_{\text{SCMP}} = \frac{\text{Perf}\left(a\right)}{\lambda/n + \left(1-\lambda\right) + \eta\sqrt{a}\lambda\left(k_1 + nk_2\right)v^{-1}\sqrt{na}}. \quad (13)$$

For mathematical simplicity only, we assume $k_2 \rightarrow 0$ when finding the performance upper bound of SCMPs. This results in a higher upper bound:

$$\text{Perf}_{\text{SCMP,max}}\left(P\right) = \text{Perf}\left(a_{opt}\left(P\right)\right)\frac{1}{2\left(1-\lambda\right) + \eta\sqrt{a_{opt}}\lambda k_1 v^{-1}\sqrt{a_{opt}n_{opt}}}. \quad (14)$$

When only one core is used, there is no performance loss due to interaction overheads. The performance of one core is given by substituting (4) into (3) where $n$=1,

$$\text{Perf}_{uniprocessor} = \eta\sqrt{\frac{P}{\gamma}}. \quad (15)$$

The upper bound for symmetric systems is the maximum between equations (14) and (15). Uniprocessors (15) achieve more performance for a given power budget than multiprocessors (14) when $n_{opt}$<1 (i.e. $\lambda$<½) or when $P$ is sufficiently large:

$$P \geq \gamma v\frac{1-2\sqrt{\lambda\left(1-\lambda\right)}}{\lambda\eta k_1}, \quad \lambda \geq 1/2. \quad (16)$$

Fig. 3 shows the performance versus power with interaction overheads for $\lambda$=0.75. When the core count is high, performance degrades due to the interaction overheads. The SCMPs are plotted with $k_2 \neq 0$, and do not cross the theoretical upper bound given by equations (14)-(16).

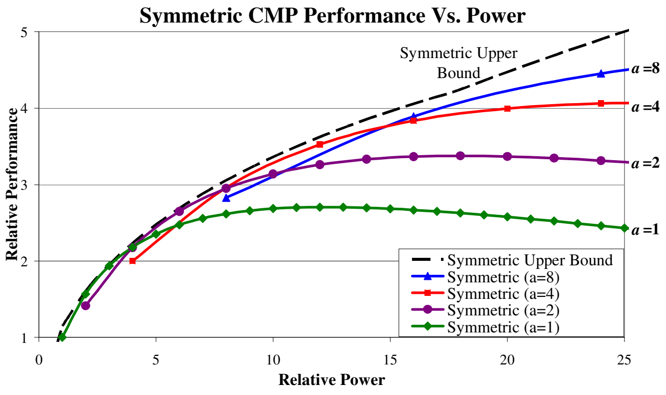

Fig. 3. The performance versus power for SCMPs for $\lambda$=0.75, $k_1$=$10^{-2}$, $k_2$=$10^{-3}$, $\gamma$=1, $\eta$=1, $v$=1. Four SCMPs are modeled, with normalized core areas of 1, 2, 4, and 8 respectively.

The interaction overhead for ACCMP is given by

$$\text{T}_{ACCMP,overhead} = \lambda M\left(k_1 + nk_2\right)v^{-1}\sqrt{\left(n-1+\beta\right)a}. \quad (17)$$

The performance of ACCMP is thus

$$\text{Perf}_{ACCMP} = \frac{\text{Perf}\left(\beta a\right)}{\lambda\left(1+\frac{n-1}{\sqrt{\beta}}\right)^{-1} + \left(1-\lambda\right) + \eta\sqrt{\beta a}\lambda\left(k_1+nk_2\right)v^{-1}\sqrt{\left(n-1+\beta\right)a}}. \quad (18)$$

Performance versus power predictions for $\lambda$=0.75 is shown in Fig. 4. The curves show two asymmetric structures modeled according to (18), as well as the upper bound for SCMPs given by equations (14)-(16). As can be seen, the ACCMPs exceed the performance versus power achieved by SCMPs at certain ranges of power. This is because ACCMPs optimize the parallel phases and serial phases separately, whereas SCMPs optimize both kinds of phases together.

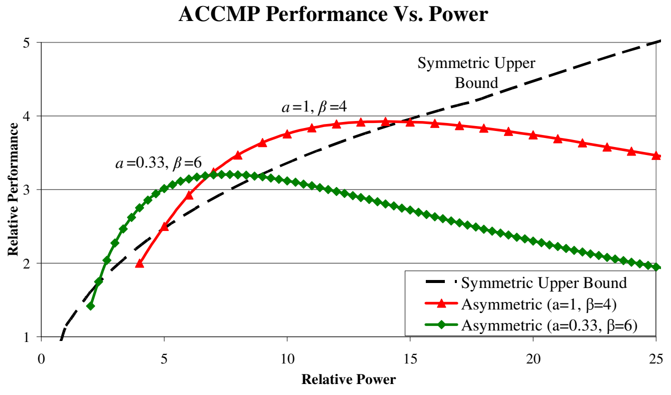

Fig. 4. The performance versus power for SCMP upper bound (with $\lambda$=0.75, $k_1$=$10^{-2}$, $k_2$=0, $\gamma$=1, $\eta$=1, $v$=1) and ACCMP (with $k_2$=$10^{-3}$). The graph shows two cases where an ACCMP crosses the upper bound of SCMPs.

## III. Emulation Environment

We validate our theory by emulations. The emulations were done using one node of 16 Power3 microprocessors of a 162 processor IBM RS-6000-SP [4]. Different core sizes (performance) are modeled with the help of a "leech" program that executes in the background and degrades specific microprocessor performance. For example, in order to model 80% of the Power3 processor, the leech process would have to consume 20% of the CPU cycles. By using different parameters for the leech program, processors of different performance can be emulated. Software threads are bound to specific processors with the help of operating system calls.

### A. *Synthetic Benchmark*

In order to test the emulation environment we have developed a synthetic benchmark using OpenMP, with $\lambda$=0.75. The synthetic benchmark contains simple computation loops with accesses to shared locks. The loop iterations in the parallel regions were distributed dynamically among the processors in chunks of 64 iterations, on a first come first serve basis. Measurements of the emulation environment show accuracy within 5% of the target core performance.

### B. *SPEC OMP Benchmarks*

Apart from our synthetic benchmark, we have run a number of SPEC OMP [3] benchmarks on our emulator. Unless specified otherwise, OpenMP compilers divide parallel work evenly among available processors. When the same amount of work is distributed between a fast core and a slow core, the net performance is reduced to that of two slow cores. Dynamic loop scheduling with a small "chunk" size (i.e. iterations) solves this problem since the maximum join waiting time will be the time required to execute "chunk" iterations by the slowest processor. The SPEC OMP benchmark sources were thus augmented to include dynamic workload distribution. These changes increase the runtime of the benchmarks since now $k_2$ is larger. Additionally, processor binding code was added at the beginning of each benchmark.

## IV. Emulation Results

The emulation results for the synthetic benchmark are shown in Fig. 5. For certain power regions, the three emulated ACCMPs demonstrate better performance versus power than any SCMP emulated, in accordance with the theoretical equations and initial intuition. Table 1 shows a comparison of the interesting SCMP and ACCMP configurations in Fig. 5.

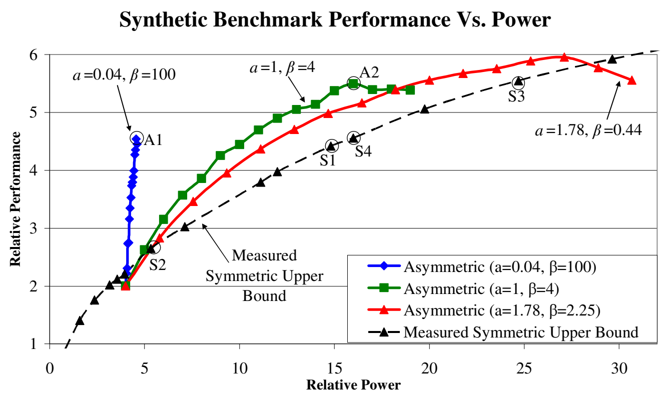

Fig. 5. Synthetic benchmark performance versus power ($\lambda$=0.75). Various symmetric configurations were emulated in order to plot the symmetric upper bound. The asymmetric multiprocessors cross this boundary.

The results for the SPEC OMP Wupwise benchmark with the "test" input set are shown in Fig. 6. Extrapolation from the emulation results reveal that $\lambda_{Wupwise,test}$=0.71, $k_1$=4.7*10$^{-3}$, $k_2$=1.2*10$^{-3}$. In the shown power range, the ACCMPs surpass the performance of every SCMP emulated.

Table 1 – Comparison of Design Points.

| ACCMP | | | | Symmetric CMP | | | Comparison | |
|---|---|---|---|---|---|---|---|---|
| Pt. | *a* | *α* | # Cores | Pt. | *a* | # Cores | Perf. | Pow. |
| A1 | 0.04 | 100 | 15 | S1 | 4.96 | 3 | +3% | -69% |
| A1 | 0.04 | 100 | 15 | S2 | 1.78 | 3 | +71% | -14% |
| A2 | 1 | 4 | 13 | S3 | 4.96 | 5 | -1% | -35% |
| A2 | 1 | 4 | 13 | S4 | 4 | 4 | +20% | 0% |
| A3 | 1 | 4 | 7 | S5 | 4 | 4 | +2% | -31% |

The Wupwise benchmark with the "train" input set required over an hour to complete on a single Power3 processor. Since this benchmark is highly parallel ($\lambda$=0.95 extrapolated), we have not seen significant benefits from using ACCMP over SCMPs. This is because the time required to execute the serial phases is negligible compared with the total execution time.

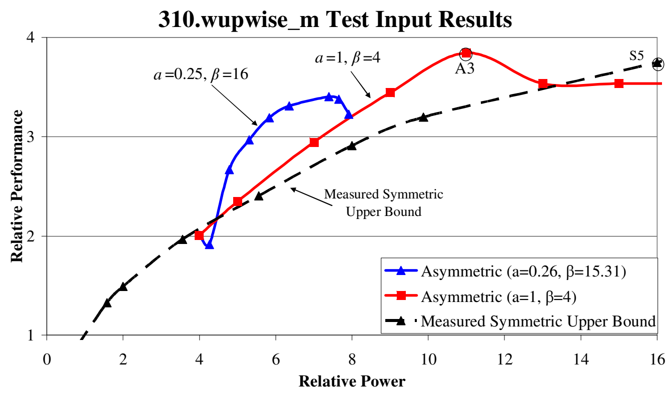

Fig. 6. Wupwise performance versus power with the "test" inputs ($\lambda$=0.71 extrapolated). Various symmetric configurations were emulated in order to plot the symmetric upper bound. The ACCMPs cross this boundary.

Most SPEC OMP benchmarks exhibited degradation when small cores were added next to large cores. The reasons for this stem from the fact that these benchmarks were written for symmetric systems. For example, the Galgel benchmark frequently calls the MATMUL function, which is a matrix multiplication function. This function was written for symmetric systems, so when small cores are introduced beside larger cores, performance degrades to the level of the small cores. Therefore, in order to fully realize the theoretical benefits of ACCMP, asymmetry aware performance tuning must be applied, which is a fertile area for future research.

## V. Discussion

We have shown by theoretical analysis that at certain power ranges, ACCMPs can achieve higher performance for multithreaded applications than any SCMP configuration. Our emulations of ACCMP structures show reduction of more than two thirds in power for similar performance, as well as more than 70% higher performance for the same power budget, in accordance with the theory and the initial intuition. ACCMPs excel in executing multithreaded programs in power constrained environments since the parallel and serial phases of software applications are executed on different core types.

Placing multiple asymmetric cores on a single die introduces many new challenges on the practical side. Scheduling in operating systems becomes more complex when processing cores are not homogeneous. Compilers must support asymmetric load balancing for parallel constructs. Additionally, ACCMP architectures must enable fast thread migration between cores, efficient inter-core communications and a scalable coherent memory system. The effects of finite and varying parallelism of software applications on ACCMP performance is left for future work.

## References

[1] G. Amdahl. "Validity of the single processor approach to achieving large scale computing capability." In Proc. AFIPS Spring Joint Computer Conf., 1967, pp. 483-485.

[2] M. Annavaram, E. Grochowski, and J. Shen. "Mitigating Amdahl's Law Through EPI Throttling." In proc of the 35th ISCA, June 2005.

[3] V. Aslot et. al. "SPEComp: A New Benchmark Suite for Measuring Parallel Computer Performance." In Proc. of WOMPAT 2001.

[4] M. R. Barrios et al. "The RS/6000 SP Inside Out." IBM, International Technical Support Organization, May 1999.

[5] S. Borkar. "Getting Gigascale Chips: Challenges and Opportunities in Continuing Moore's Law." In ACM Queue vol. 1, no. 7 - October 2003.

[6] H. P. Flatt. "Performance of Parallel Processors." In Parallel Computing, Vol. 12, No. 1, 1989, pp. 1-20.

[7] E. Grochowski, R. Ronen, J. Shen, and H. Wang. "Best of Both Latency and Throughput." In proc. of the 22nd ICCD, October 2004.

[8] L. Hammond, B. A. Nayfeh, and K. Olukotun. "A Single-Chip Multiprocessor." In IEEE Computer, September 1997 (Vol. 30 No. 9).

[9] R. Kumar, D. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas. "Single-ISA Heterogeneous Multi-core Architectures for Multithreaded Workload Performance." In proc. of the 31st ISCA, June 2004.

[10] D. Menasce and V. Almeida. "Cost-performance analysis of heterogeneity in supercomputer architectures." In Proc. of ICS, 1990.

[11] D. Moncrieff, R. E. Overill, and S. Wilson. "Heterogeneous Computing Machines and Amdahl's Law." In Parallel Computing, vol. 22, 1996.

[12] F. Pollack. "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies." In Micro 32, 1999. http://www.intel.com/research/mrl/Library/micro32Keynote.pdf