ACCMP - Asymmetric Cluster Chip Multi-Processing

Tomer Y. Morad
Department of Electrical Engineering
Technion, Israel
tomerm@tx.technion.ac.il

Uri C. Weiser
Intel Corporation, Israel
uri.weiser@intel.com

Avinoam Kolodny
Department of Electrical Engineering
Technion, Israel
kolodny@ee.technion.ac.il

Abstract— Achieving high performance within a power envelope becomes extremely difficult as feature sizes decrease. Uniprocessor architects encounter diminishing returns when trying to attain higher performance in exchange for area and power. For multithreaded programs, symmetric chip multiprocessing (CMP) offers higher throughput and power efficiency than uniprocessors. However, symmetric CMP does not scale well with technology. In this paper we explore the theoretical advantages of placing asymmetric core clusters in multiprocessor chips. All of the cores on the ACCMP die have the same instruction set architecture, but may have a completely different micro-architecture. We show that asymmetric core clusters are expected to achieve higher performance per area and higher performance for a given power envelope.

***Index Terms*— ACCMP, Multiprocessing.**

I. Introduction

Power consumption is becoming extremely important as microprocessors become more complex. The constant decrease in feature sizes enables processor architects to improve processor performance by using the extra silicon real estate. Due to wire delays and increased architectural complexity, uniprocessor performance does not scale well with the increase of the effective processor die area. Additionally, uniprocessors are becoming extremely power inefficient, as 1% of performance increase costs approximately 3% in additional power consumption. Since the power consumption envelope and the power density of current microprocessors are approaching or have even reached in some cases their limits, a new design paradigm must be employed.

Symmetric chip multiprocessing (CMP) aims to increase performance while increasing the power and area proportionally. CMP processors divide the threads of multithreaded applications among the symmetric cores, exploiting their parallelism. Although single-threaded applications may run slower on a CMP, modern desktop and server operating systems run multiple threads, making the overall performance higher. Most leading microprocessor manufacturers are either already offering CMP processors or have announced plans to do so [1].

When adding an additional core to a symmetric CMP chip, 1% of power and 1% of area are exchanged for roughly 1% of performance, given that we ignore thread contention. This can be seen in Figure 1. However, it is possible to pay less than 1% of power for each one percent of performance, by placing asymmetric [2] core clusters on one die. General purpose threads will be executed on the larger core clusters, while the smaller core clusters will be highly tuned for specific application domains, such as streaming applications (media, data encryption...), virus checking, data mining, etc. Due to their smaller size and their tuned architecture, addition of smaller cores can provide a performance advantage of better than 3% for each 1% in power. This can be seen in Figure 2.

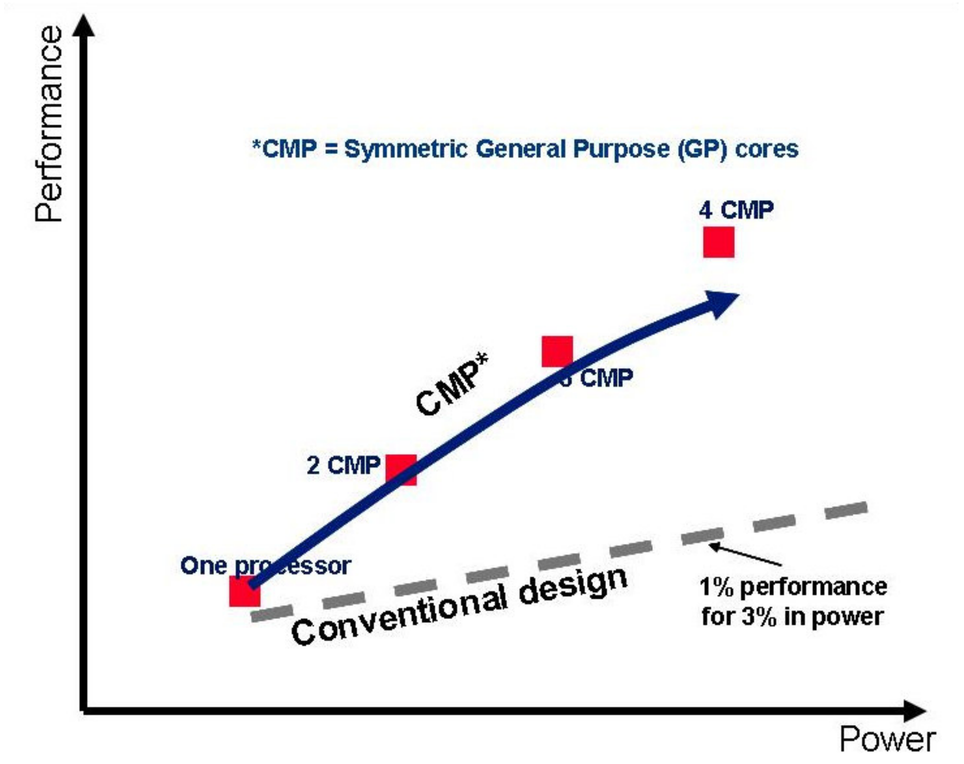

Figure 1 – Performance Vs. Power of a conventional uniprocessor and a CMP processor

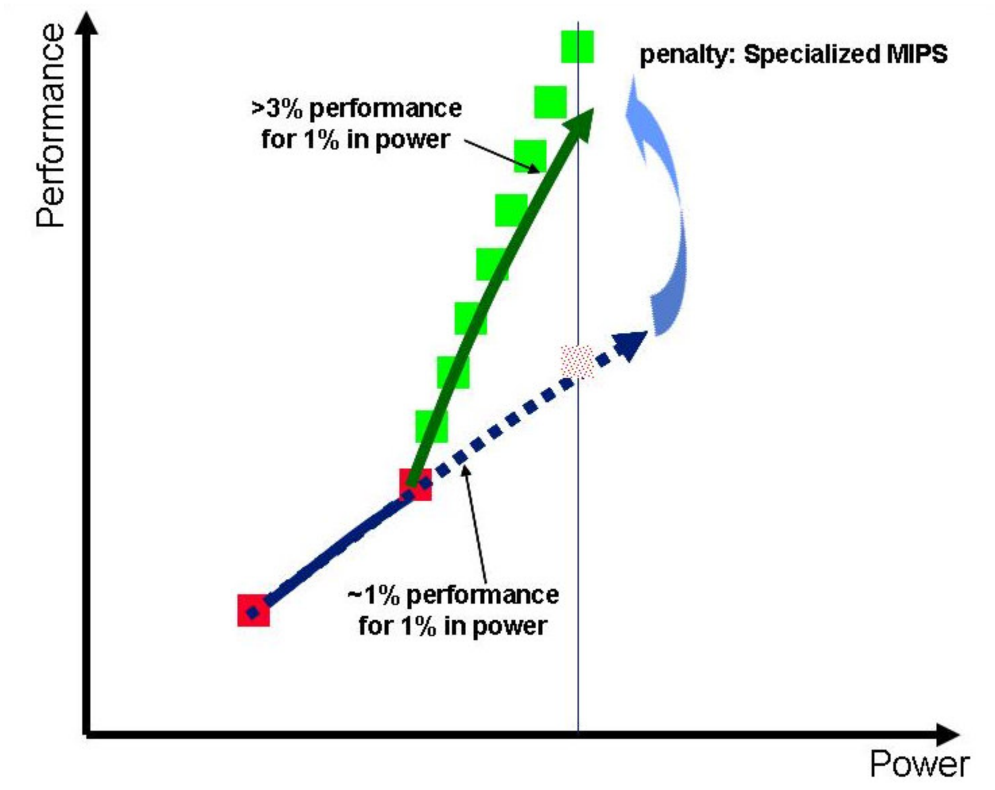

Figure 2 – Performance Vs. Power of ACCMP and CMP, near the power wall.

II. Symmetric Chip Multi Processing

We will now compare the performance of a multi-core processor to the performance of a single-core processor using a simple model. The analysis is done for an endless number of threads and for a constant die area. For a multi-core processor with N symmetric cores, each core will have an area smaller by a factor of N. Determining the average IPC and the frequency of the smaller core based only on its area is not trivial. Architectural features such as out of order execution, cache sizes, branch prediction and the number of pipeline stages have different effects on the overall IPC and frequency. For our model, we assume that the performance of each small core is $\frac{1}{\sqrt{N}}$ of that of the uniprocessor. This takes into account the fact that there are diminishing returns when trying to exchange area for performance. The diminishing returns stem from the interconnect delay which is linearly proportional to the wire lengths. Given these assumptions, we get:

$$\frac{Performance_{Core}}{Performance_{Uniprocessor}} = \frac{1}{\sqrt{N}}$$

Assuming that the amount of network traffic on the die grows exponentially with N, each core's performance will be multiplied by $c^{\alpha N}$, where $0 < c < 1$ and $\alpha > 0$. We now get the following performance comparison:

$$\frac{Performance_{multi-core}}{Performance_{uni-processor}} = \frac{N\frac{1}{\sqrt{N}}c^{\alpha N}}{1} = c^{\alpha N}\sqrt{N}$$

Although the performance of N independent small cores increases as N grows, the exponential growth of the network traffic limits the multi-core processor's performance. However, for independent threads, the network traffic will be negligible.

III. Asymmetric Chip Multi Processing

Using a simple model, we were able to roughly evaluate multi-core processor performance. However, we are not limited to having only identical cores. If the area were to be divided asymmetrically using these ratios with M+N-1 cores:

$$a_1 + a_2 + ... + a_M = \frac{N-1}{N} + \frac{M}{NM} = 1$$

The performance of each core would be:

$$\frac{Performance_{asym-core}}{Performance_{uni-processor}} = \sqrt{a_n}$$

Therefore:

$$\frac{Performance_{ACCMP}}{Performace_{uniprocessor}} = \sum_{n=1}^{M+N-1} \sqrt{a_n}c^{\alpha(M+N-1)} = c^{\alpha(M-1)}c^{\alpha N}\left(\sum_{n=1}^{N-1}\sqrt{\frac{1}{N}} + \sum_{n=1}^{M}\sqrt{\frac{1}{MN}}\right) =$$

$$= c^{\alpha(M-1)}c^{\alpha N}\left(\frac{N-1}{\sqrt{N}} + \sqrt{\frac{M}{N}}\right) = c^{\alpha(M-1)}c^{\alpha N}\left(\sqrt{N} + \frac{\sqrt{M}-1}{\sqrt{N}}\right)$$

We are interested when this expression is larger than the performance of CMP. This happens when:

$$\frac{Performance_{ACCMP}}{Performace_{uniprocessor}} = c^{\alpha(M-1)}c^{\alpha N}\left(\sqrt{N}+\frac{\sqrt{M}-1}{\sqrt{N}}\right) \geq c^{\alpha N}\sqrt{N} = \frac{Performance_{CMP}}{Performance_{uni-processor}}$$

$$c^{\alpha(M-1)}\left(1+\frac{\sqrt{M}-1}{N}\right) \geq 1$$

This equation is true for the following example values:

$$c = 0.5$$

$$\alpha = 1/30$$

$$M = 16$$

$$N = 3$$

$$0.5^{1/30(16-1)}\left(1+\frac{\sqrt{16}-1}{3}\right) = \frac{1}{\sqrt{2}}2 = \sqrt{2} \geq 1$$

Evident from the above analysis, given a constant die area, better performance can be achieved by dividing the die into asymmetric cores than by dividing it into symmetric cores. Since power consumption is roughly proportional to the die area for a specific technology, the ACCMP will perform much better than CMP or uniprocessor designs for a specified power envelope.

The ACCMP processor will contain clusters of different cores, but with the same instruction set. Using the same ISA in all of the cores enables the migration of threads between the cores. Threads migrate in response to core utilization, power envelope requirements and performance goals. An example of an ACCMP processor can be seen in Figure 3.

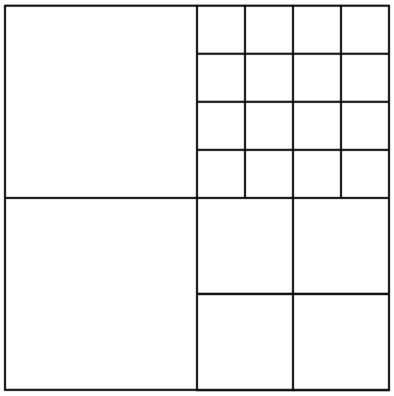

Figure 3 – Example of an ACCMP processor. The above processor has two general purpose cores, and two clusters of different smaller cores.

IV. Asymmetric Hardware and Software Schedulers

Since the ACCMP processor contains clusters of different cores, an efficient mechanism must be developed in order to schedule threads to the correct cores. For example, streaming media threads need to perform some kind of computation within a specific time frame. Executing media threads on the big general purpose cores is not efficient in terms of power, as the extra performance of the GP cores will not be used by the media thread.

The cost function for determining the best matching of threads and cores is dependent on the type of cores available, the type of threads currently running, and power/performance considerations. Since programs have different phases during their runtime [3], the schedulers must dynamically evaluate every once in a while the current requirements and calculate the best configuration. Thread migration should not incur a high overhead, if implemented correctly.

An efficient scheduler should schedule special purpose threads to their tuned cores. For example, media threads should execute on media-tuned cores. The scheduler can identify the type of the running thread according to hints supplied from the thread, or by sampling the runtime properties of the thread. The runtime properties can include IPC, memory usage, power consumption and the ISA footprint. The scheduler should also take into account the required time for completion for each thread. This should be supported by the compiler and the operating system.

V. On-Chip Communication and Memory Hierarchy

The ACCMP cores share the same memory space, therefore an efficient memory hierarchy and communication protocol must be used [4]. For design simplicity, cores should communicate via a Network on Chip [5]. The memory hierarchy should include a level-1 cache in each core, an intra cluster level-2 cache, and an inter cluster level-3 cache.

VI. Asymmetric Software Architecture

Software for the ACCMP should be multithreaded in order to take advantage of its parallelism. The programming methodology should shift from single-threaded applications to multithreaded applications. Web servers, database servers and application servers which serve multiple requests concurrently should benefit greatly from the use of the ACCMP.

Some of the clusters of the ACCMP will be tuned for specific applications. The compiler should know which core it should tune for, not just the ISA. For example, compiling an application for an in-order core is quite different than compiling the same application for an out of order core. Additionally, tuned cores will have different throughput for different instruction sequences.

VII. Cluster Mixture and Compatibility

The mixture of the types of cores and clusters required for high performance differs for each application. Therefore, the ACCMP should always include high performance general purpose cores. Other cores will be tuned for specific application domains, and their types and quantity will differ for each technology generation. Thanks to the constant ISA among the cores, threads that were tuned for a specific core will still be able to execute on other cores, but will take a little more time to complete.

VIII. Conclusions

In this paper we have shown the advantages of asymmetric clustered chip multiprocessors over symmetric CMP and uniprocessor architecture. The advantages include higher performance per area, and higher performance per power. Asymmetric schedulers become much more complex than their symmetric counterparts.

References

[1] Kalla, R. Sinharoy, B. Tendler, J.M., "IBM power5 chip: a dual-core multithreaded processor", IEEE Micro, Volume24, Issue 2, April 2004.

[2] R. Kumar and D. M. Tullsen, P. Ranganathan, N. P. Jouppi, K. I. Farkas, "Single-ISA Heterogeneous Multi-Core Architecture Designs for Multithreaded Workload Performance", ISCA 2004

[3] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder. "Discovering and Exploiting Program Phases". IEEE Micro: Micro's Top Picks from Computer Architecture Conferences, Nov./Dec. 2003.

[4] L. Ivanov, R. Nunna, "Modeling and verification of cache coherence protocols", ISCAS 2001. Volume: 5 , 6-9 May 2001

[5] W.J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", DAC 2001, June 18–22, 2001, Las Vegas, Nevada, USA