

# SMP-SoC is the Answer if You Ask the Right Questions

PHILIP MACHANICK

National ICT Australia and School of ITEE, University of Queensland

---

With growing interest in multiprocessor system-on-chip (SoC) designs and the increasing number of multicore designs at the high to medium end of the desktop and server markets, the question arises as to whether there should be convergence of the two concepts. This paper explores design principles for SoC multiprocessors and relates them to more general system requirements. It makes a case for focusing design efforts on symmetric multiprocessor (SMP) SoC designs, which have the best chance of making an impact in a wide range of markets, rather than designs for very specific niches, such as the IBM-Sony-Toshiba Cell processor. If SMP-SoC designs are successful, they could represent the next change in packaging in the general computer market: the next step after the microprocessor. A case is made not only for designing a SMP-SoC alternative to the Cell, but for using such a design as an alternative to an aggressively-pipelined uniprocessor.

Categories and Subject Descriptors: C.1.2 [Processor Architectures]: C.1.2 Multiple Data Stream Architectures (Multiprocessors)—*Multiple-instruction-stream, multiple-data-stream processors (MIMD)*; C.4 [Processor Performance of Systems]: —*Design Studies*

General Terms: Measurement, Performance, Design

Additional Key Words and Phrases: chip multiprocessor, system on chip, high performance computing, embedded systems

---

## 1. INTRODUCTION

The growth in complexity of single-chip devices creates the potential for another packaging revolution, following in the footsteps of the emergence of the minicomputer, followed by microprocessors. The joint announcement in 2004 by IBM, Sony and Toshiba of the Cell [Krewell 2005a], a multicore device with a PowerPC core, 8 vector cores and 2 Mbytes of memory distributed between the vector cores indicates the emergence of relatively sophisticated single-chip devices. Microprocessors, originally designed for simple embedded applications, have become the mainstream processor in conventional computers. System on Chip (SoC) multiprocessors have the potential to create a new packaging revolution. An SoC design is generally a complete computer minus peripherals and power supply on one chip (i.e., processor, memory and at least rudimentary I/O are included). Cell itself is not strictly an SoC solution since it does not have a general-purpose main memory in its initial form, but it illustrates a trend. In the not too distant future, there should be a wide variety of increasingly powerful single-chip multiprocessors aimed at niche but high-volume markets.

The Cell design indicates what is possible by virtue of the scale of the design. As a general-purpose computer part, its value is doubtful, as programming specialized vector units is unlikely to result in anywhere close to the rated peak throughput in all but a limited number of special cases<sup>1</sup>. This paper explores design principles for a more useful SoC part – one which could not only play a role in the embedded space, but also become a building block for more general systems.

The idea of chip multiprocessors (CMP) been explored as an alternative to high-end processors for super-computer workloads [Olukotun et al. 1996] and have in recent times started to migrate to more general-purpose designs (e.g., the Intel Core Duo and AMD Opteron Dual-Core) This paper argues for SMP-SoC as an alternative to commodity microprocessors running multitasking workloads, as the next logical step. The high-end computation market is the best case for aggressive pipelines, because floating point generally has better instruction-level parallelism. Typical consumer applications are less likely to have the same potential to exploit instruction-level

---

<sup>1</sup>A matrix multiply example available from IBM's developer site <http://www-128.ibm.com/developerworks/> is over 1200 lines of code, and does not cover the general case. Compare that with a simple matrix multiply, which is three nested for loops containing an add and a multiply.

---

Author Address:

P Machanick, School of ITEE, University of Queensland, St Lucia, Qld 4072, Australia; philip@itee.uq.edu.au

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, that the copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than SAICSIT or the ACM must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2006 SAICSIT

parallelism. However, many consumer-level applications are multithreaded (if only at the level of operating system services like the user interface), and most consumer operating systems are inherently multitasking, with many processes simultaneously active. An SMP-SoC part could therefore be a useful alternative to an aggressively pipelined uniprocessor in some situations. The trick is to design an overall combination that could fit both the embedded market (which wants a low chip count) and the general-purpose market, where several parts covering needs like main memory and

This paper starts from identifying questions about factors which may lead to or inhibit success of a new packaging revolution. From there, some specific questions which need to be addressed in the context of SoC multiprocessors are identified. These questions are used as a base for an evaluation of the prospects for an SMP-SoC design, both as an alternative to the Cell and as an alternative to an aggressively pipelined uniprocessor. The Cell is chosen as an example of the trend in SoC multicore designs; much of what is written here as the “alternative” case could apply to the Xbox 360, which has a 3-way SMP multicore Power design [Brown 2005].

The remainder of this paper outlines the background to the problem. Next, it presents a series of questions and answers which lead to SMP-SoC as an interesting development to pursue. Finally, the paper wraps up with conclusions.

## 2. BACKGROUND

### 2.1 Introduction

There are three issues to consider in deciding whether the SoC multiprocessor concept will make it to the conventional computer market. First, there is the kind of multiprocessor which is likely to be successful as a general-purpose design. Second, there is the question of what kind of multiprocessor can be implemented on a single chip, and whether it can be performance-competitive with single-processor alternatives (used singly or multiply). Finally, it is useful to examine where existing developers of SoC multiprocessors are focusing their efforts.

This section considers each of these issues in turn, and summarises with conclusions.

### 2.2 Symmetric Multiprocessors

In the early days of large-scale multiprocessors, various heterogeneous design alternatives were explored, such as master-slave designs, multicomputer systems and symmetric multiprocessors [Enslow 1974].

Shared-memory symmetric multiprocessors (SMP) are the common organisation today because they are more general than the other designs, which can all be realised on a shared-memory SMP machine. For example, a multicomputer can be approximated on a shared-memory machine by implementing message passing through the memory system. A master-slave design can be implemented by having one process or thread which allocates work to others (without any necessity for a physical processor to be designated as the “master”).

Further, shared-memory SMPs work well for multitasking workloads (subject to scalability of the operating system) whereas scheduling a general workload across the other models is a considerable challenge.

### 2.3 Chip Multiprocessors

Chip multiprocessors have originally been proposed as a solution to the decreasing gains of aggressive instruction-level parallelism [Olukotun et al. 1996]. The logic is that a simpler processor could be replicated in the same die area as a single complex processor. While the peak throughput rate of each processor would be lower than that of the single complex processor, a higher fraction of the peak throughput of each individual processor would be realisable, resulting in higher overall performance. While compilers capable of finding threads in non-threaded code would be needed to exploit all potential parallelism, such designs could potentially perform well with existing multithreaded code.

Applications with significant instruction-level parallelism are not the obvious target for such a design, as they already are the best case for aggressive pipelines.

Consequently, the emphasis on chip multiprocessor work targeting high-end computation, in the form of the Stanford Hydra project [Hammond et al. 2000], has created a perception that the viability of the concept relies on tools like parallelising compilers. The first commercial chip multiprocessors were high-end designs, such as the top of the IBM Power range, with a small number of cores per die, each still with an aggressive pipeline.

However, in consumer workloads, multithreaded code is common (e.g., in commodity windowing systems), and most real workloads are heavily multitasking. Further, such workloads are not well suited to an aggressive pipeline, with the well-known property of typical integer code of small basic blocks [Wall 1991]. While some commodity architectures have adopted simultaneous multithreading (SMT) [Lo et al. 1997] – called HyperThreading by Intel – as a way of improving throughput on aggressive pipelines, this solution does not address one of the

other claimed advantages of CMP: faster design cycles. Further, multiple threads competing for the same cache increase cache misses [Muller et al. 1996].

There is therefore to be a case to re-evaluate the chip multiprocessor as an option for multithreaded code or multitasking workloads. This case has obviously been considered by major vendors such as Intel, Sun, AMD and IBM, all of whom at time of writing have designs aimed at the medium to high end of the mass and server markets. However, little research has been published on consumer workloads on this class of processor.

## 2.4 SoC Multiprocessors

While SMP has become the mainstream in large-scale multiprocessor systems, heterogeneous multiprocessors are considered the norm in embedded applications [Wolf 2005] – to the extent that a whole book on multiprocessor systems-on-chips has been written, with the emphasis on heterogeneous designs [Jerraya and Wolf 2005a].

What are the arguments for heterogeneous designs in this context? Are the requirements sufficiently different from historical multiprocessor designs?

The biggest issue raised in defence of heterogeneous designs is the *real-time argument* [Jerraya and Wolf 2005b]. According to this argument, real-time deadlines have to be met the most efficient way possible and a special-purpose design can be justified to meet this need. By contrast, with a more general computer, Amdahl's Law would dictate that the average case be considered, rather than a special case.

The other issue of significance in this area is power consumption. While power is an issue for large-scale designs (dissipating 10W per processor is obviously less of a problem than 100W per processor if you have hundreds of CPUs), it is a first class concern for embedded designs. Especially as a growing fraction of embedded computing is in mobile devices, energy efficiency is an issue of growing concern. Accordingly, energy-efficient design is a significant factor in work on MP-SoC [Yang et al. 2005; Dey et al. 2005].

As with general-purpose designs, heterogeneous architectures pose significant programming challenges. The real-time argument carries some weight. While a DSP or other specialised architecture may be hard to program to anywhere close to the same fraction of peak throughput as is realisable in the general case with a conventional architecture, it may make solution of some problems possible (e.g., software radio, HDTV codecs).

However, as the processor speed and size of on-chip memory grows, it is likely that these special-case arguments will be harder to support.

## 2.5 Summary

Putting all these things together, the long-term trend in general multiprocessors has been to SMP. Chip multiprocessors are a promising idea, and are finally being targeted at a more appropriate part of the design space, lower-end applications such as consumer desktops, because multithreading and multitasking can immediately use more processors. By contrast, adapting high-end single-threaded computations to CMP is a harder ask, because missing technologies like efficient parallelizing compilers are needed.

SoC multiprocessors present a special case for heterogeneous designs because of the real-time argument, backed up by the fact that early designs are resource-constrained. However, if SMP-SoC designs could be pursued, they could fill the gap between SoC and desktop designs.

# 3. THE QUESTIONS

## 3.1 Introduction

There is a lot of momentum behind SoC multiprocessors, and increasingly also the chip multiprocessor idea [Kongetira et al. 2005; Krewell 2005b; Kgil et al. 2006]. It appears that SoC hardware designers are attempting to reinvent the history of multiprocessors, by exploring highly asymmetric designs, a concept which has failed in the past. While there are arguments that this time it's different, history carries a lot of weight. The design issues which turned out to be problematic for mainframes (too few address bits, complex instruction sets, unnecessary addressing modes) were repeated in minicomputers and subsequently microprocessors.

This section raises some questions which should be answered to decide the optimum direction for SoC multiprocessors, and provides some answers. The section concludes by identifying the most critical issues to be decided before a definitive direction can be determined.

## 3.2 Questions and Answers

It is important in determining the likelihood of success of a new packaging breakthrough which issues apply from previous generations, and which do not. The questions and answers are accordingly divided into those where they may be differences, and those where there are unlikely to be differences.

The biggest differences are likely to arise from the fact that we are looking at a larger fraction of the computer

in a single package. However, the growing importance of real-time issues may also be a factor in differentiating this change from previous packaging transitions.

### 3.2.1 *Perennial Issues*

To start, here are some questions with answers which are unlikely to change in the near future.

- (1) Why won't asymmetric and various kinds of heterogeneous multiprocessors be a success? – This idea has been well worked over in the past. The biggest factor is difficulty in programming them. With uneven capabilities, workloads have to be massaged to fit the various processors, and there are inevitably workloads which are a poor fit. A symmetric multiprocessor may be sub-optimal for some workloads, but is generally better than a specialist architecture on a wider range of workloads.
- (2) Won't CPU speed improvement overwhelm DRAM speed improvements? – Yes, but SoC packaging includes the option of including an increasingly large on-chip SRAM main memory, an idea explored already in the uniprocessor realm [Machanick and Patel 2003; Machanick 2004; Machanick et al. 1998]. Further, 3-dimensional stacking is a promising idea, which could lead to tight integration of DRAM into the CPU cores, with much lower latency [Kgil et al. 2006]
- (3) Shouldn't we separate the issues for large-scale and small-scale designs? – Current microprocessors show the value of designs applicable across a wide variety of scales. MIPS, originally designed with performance in mind, is now a big seller in the embedded market. The IA32, originally designed for small-scale systems and embedded applications, has been used in designs up to hundreds of processors. Designs which work at multiple scales multiply opportunities for economies of scale, and developing operating systems and programming tools.
- (4) Aren't multiprocessor applications hard to develop? – Yes. This issue is not likely to go away soon. However, a general-purpose SMP can execute arbitrary multitasking workloads potentially with reasonable efficiency (depending on the operating system). A significant fraction of commercial workloads is multithreaded (e.g., user interface, server code) as well.

In summary, there is a case for developing designs which work across a wide variety of scales. Some of the long-term difficulties with multiprocessor designs need to be taken into account. However, as we see next, repackaging the CPUs into a single chip introduces some potential benefits, breaking previous limitations on improving performance.

### 3.2.2 *Possibilities for Change*

Including more of the computer on a single chip has obvious cost benefits. Does an SMP-SoC design introduce potential advantages in performance as well? Let's consider some questions about issues which could change as a result of different packaging.

- (1) Haven't SMPs in the past had scaling problems? – Yes. But the biggest problem is the interconnect, which either has to be designed way too fast for the first version, or is not fast enough for a CPU upgrade. By including the interconnect on-chip, this issue goes away. A new CPU package includes the interconnect. Further, if the three-dimensional stacking idea pans out, DRAM latency will at least for some years be less of a bottleneck.
- (2) Don't real-time requirements create a special case for heterogeneous architectures? – Only in specialised cases (e.g. HDTV, where the data flow is huge and the computation extremely regular) is this true. In most other cases, multiple conventional CPUs will be easier to program at closer to their theoretical peak throughput. Specialist workloads often in any case are offloaded to specialist hardware. For example, despite the increasing trend towards multimedia extensions, graphics processing is most efficiently done on a graphics processor with its own memory. Offloading encryption to a specialist engine is another example [Kongetira et al. 2005; Krewell 2005b]
- (3) Don't special requirements of embedded systems outweigh the difficulties? – As long as embedded systems are highly resource constrained, the argument holds good. Small memories mean small hand-tuned programs. Even if the programming model is a nightmare, efficiency can be achieved with small examples. Once programs become larger, the efficiencies of hand-tuned code become harder and harder to achieve. If we are talking about devices with hundreds of millions of transistors, this kind of argument loses traction.
- (4) Why go the SoC route when we can still go a long way in developing CPUs as a single component? – It's not clear that throwing more transistors at the CPU alone is worth it. The looming memory wall [Wulf and McKee 1995] requires that we have bigger and bigger caches, and some have been arguing for some time for

moving more memory on-chip [Saulsbury et al. 1996]. The further step proposed here is to have memory and a multiprocessor on one chip.

- (5) Why go MP when there's so much investment in instruction-level parallelism? – ILP has its limits. Approaches like simultaneous multithreading (SMT) [Lo et al. 1997] have been introduced because of limits to ILP. The advantage in the SoC world of multiple simple processors rather than one aggressively pipelined CPU is that each processor is simpler and uses less power. Other arguments for chip multiprocessors (CMP) [Olukotun et al. 1996] also apply, including the potential for turning around new designs faster than is possible with aggressive pipelines.

Some of the arguments here can only be tested by building real systems. The CMP arguments have already been evaluated by producing designs based on CMPs [Hammond et al. 2000] – even if the aim has been high-end systems. The most compelling argument for exploring the SMP-SoC space is the history of designs at the low end obsoleting high-end competition: the minnows eat the sharks.

## 4. SOME DESIGN ALTERNATIVES

### 4.1 Introduction

Let's put all these ideas together into some concrete options to illustrate the possibilities. The Cell design is taken as a starting point to indicate what is possible with current technology. The Cell has 234-million transistors, 1 PowerPC core, 8 vector cores [Krewell 2005a; Pham et al. 2005], and 2MB of SRAM distributed over the vector cores [Dhong et al. 2005].

It would be useful to do a detailed design to work out exact equivalences, but an approximation to the same scale of design can be made as a starting point.

The remainder of this section outlines principles for an SMP-SoC, then presents a few details of the design – first as an alternative to the Cell then as an alternative to an aggressive uniprocessor. These options are followed by some ideas of how an SMP-SoC may be used. Finally, the key issues are summarised.

### 4.2 Principles

In keeping with the view in this paper that a convergence of the ideas of CMP and MP-SoC should be sought, the starting point is a multiprocessor design of as general an applicability as possible. That starting point implies an SMP design.

From there many different possibilities could be explored. However, in keeping with the general principles of the CMP idea, the approach proposed is to have relatively simple general-purpose processors, and a simple interconnect. The main difference from the CMP idea, which places the proposal in the SoC space, is an on-chip memory and some basic on-chip I/O.

The purpose in placing the design in the SoC space is to make it a viable option for single-chip embedded applications (game engines, sophisticated cell phones, etc.). This creates the potential for a very large market to build economies of scale. The purpose in design for generality rather than the approach advocated by others in the SoC space of heterogeneous or asymmetric designs [Jerraya and Wolf 2005b] is to produce a design which can also work for large-scale systems.

The next issue to consider is how such a design could compete with a complex uniprocessor architecture using the same chip space. As a first cut, a simple comparison can be made on the basis of a workload which contains enough processes or threads to keep a multiprocessor busy. To keep the model simple, this paper compares a 4-processor SMP-SoC design with a uniprocessor. The comparison is with a uniprocessor with 4, 8 and 16 times the throughput of the SMP design's individual processors (neglecting stalls for cache misses).

The evaluation assumes the following parameters:

- *cache miss penalty* – 60ns
- *average instruction latency* – (excluding cache misses)
  - *SMP-SoC* – 1ns
  - *uniprocessor* – 0.25ns (SMP-SoC  $\times$  4), 0.125ns (SMP-SoC  $\times$  8), 0.0625ns (SMP-SoC  $\times$  16)
- *variation in contention for multiprocessor bus* – it is assumed for simplicity that in the multiprocessor bus, either an access has no contention, or has to wait for an integral number of access times (1, 2 or 3), with variations on the fraction of contention
- *cache miss variation* – two cases are considered: relatively low miss rate of 0.001 and a higher miss rate of 0.01; in both cases, the SMP-SoC design is compared with the same and double the miss rate of the uniprocessor alternatives

—*SMP bus penalties* – the extra cost of acquiring the SMP bus is put at 20ns, and of acquiring the bus after waiting for contention at 5ns

The values for additional latency for the SMP bus are arbitrary and actual values would depend on the design. However, these latencies are likely to be better in practice, because a single-chip design would not incur the penalties of a traditional off-chip SMP bus. Consequently, results based on these numbers will not unduly favour the case for SMP-SoC.

The relative speeds of the uniprocessor and the SMP-SoC design assume a very aggressive pipeline on the uniprocessor alternatives and possibly faster clock speeds. These assumptions depend on ignoring obstacles to speeding up aggressive uniprocessors, such as known limits on instruction-level parallelism [Wall 1991].

The values for miss penalty are in line with DRAM speeds available at the time of writing. It is assumed that misses between on-chip caches will make less of a difference than misses to DRAM.

Variations on the degree of contention for the SMP bus are modelled by a simple formula varying from a high contention situation (only 10% of misses do not find the bus occupied; over 80% have to wait for all 3 other processors) to low contention (100% of misses find the bus unoccupied). Figure 1 illustrates the scenarios.

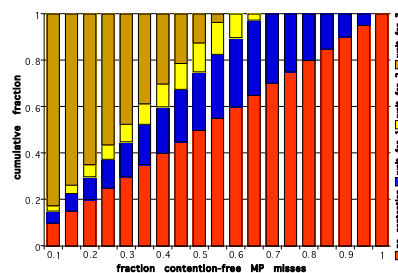


Figure 1. Modelled variations on contention for an SMP bus. Shown as a cumulative distribution for each variation on what fraction of DRAM access experience no contention.

Actual degrees of contention for the bus would of course depend on the workload; the simple model adopted here is intended as a basis for making a first cut on suitability of an SMP-SoC solution versus a uniprocessor, by presenting a range of scenarios.

These parameters are used in a simple analytical model, intended to illustrate relatively large-scale effects. More detailed modelling would require full-system simulation, given that multitasking workloads are needed for reasonable measurement.

#### 4.3 Detail of SMP-SoC vs. Cell

The starting point for the design is the amount of logic used by the Cell: 234-million transistors. The notion is to illustrate how this budget could be redeployed to create a more generally useful part.

The Power core could be simplified by removing multithreading support, and replicated, replacing the 8 vector units by 4 or more Power cores, with a shared second-level (L2) cache, an idea which has been shown to work well in large-scale multiprocessor designs [Cheriton et al. 1989; 1991]. It would be possible to vary the chip space allocated to CPUs by making them less aggressive, or decreasing L1 cache. If on the other hand the existing Power core was simply replicated, there should be space on the chip for 4 of them plus 1MB of shared L2 cache (see Figure 2(c)).

In summary, Table I illustrates the differences between the Cell and the proposed SMP-SoC design.

	Cell	SMP-SOC
General-Purpose CPUs	1	4
General-Purpose CPU ILP	multithreaded	single-threaded
Vector units	8	0
L1 cache	32KB i+32KB d	32KB i+32KB d
L2 cache	512MB	shared 1MB
other SRAM	256KB per vector unit	none

Table I. Comparison of Cell and proposed SMP-SoC. The general-purpose CPUs in both cases are dual-issue with similar pipelines.

The logic behind the proposed variation from the Cell design is that the Cell design, while exhibiting impressive peak throughput, will be a nightmare to program, if history is a guide. The SMP-SoC design, on the other hand,

represents a logical progression from existing multiprocessor designs, drawing on the ideas of CMP research. The most novel aspects of the SMP-SoC idea is the proposal to target a CMP design at the intersection between the low end of desktop and server design space and the SoC design space. This is an idea already on the horizon with the medium to high end of the personal computer space occupied by dual core designs.

The one potentially controversial idea – the point where a true SMP-SoC becomes an option – is providing an SRAM main memory as the next step up when more chip space is available, instead of increasing the size of the shared cache. This idea would only work with a small kernel. While microkernels exist in the research space [Liedtke 1996], popular operating systems use monolithic kernels. There are versions of Linux which run on microkernels such as *mkLinux* which runs on Mach, and *L<sup>4</sup>Linux*<sup>2</sup>. However, operating systems such as MS Windows and Mac OS have large kernels. Another option is to include DRAM in single package using three-deminsional stacking, as is proposed in the PicoServer design [Kgil et al. 2006].

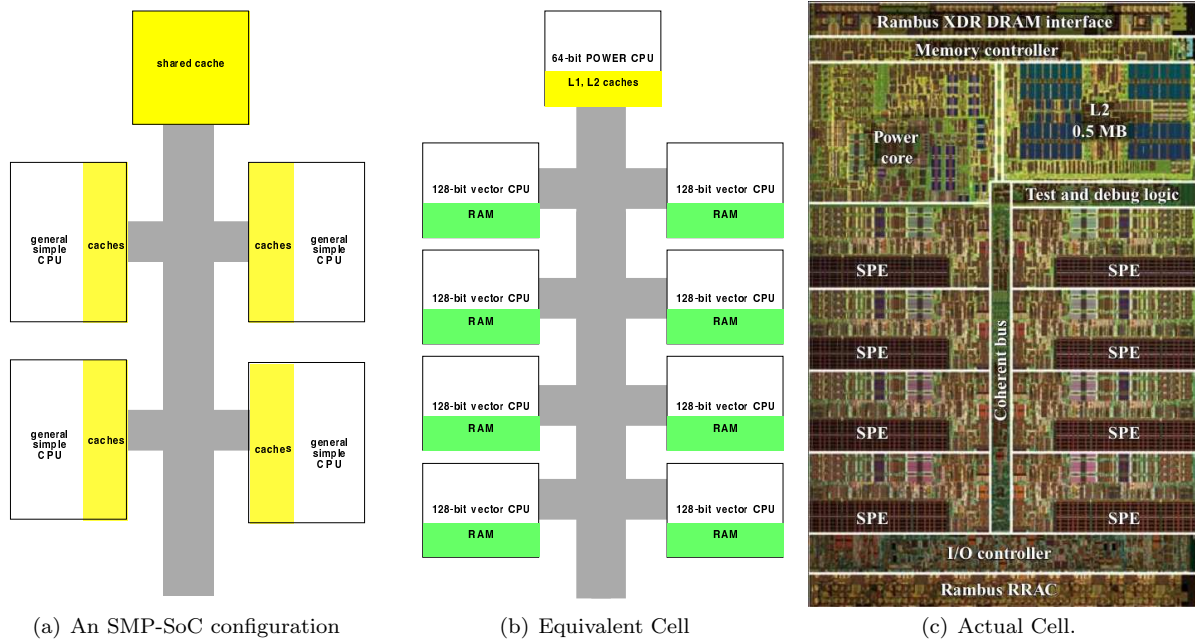


Figure 2. Alternative uses of the same chip space.

Figure 2 illustrates one possible reuse of the logic space of the Cell, neglecting some components such as I/O, which are likely to be similar. The Cell is shown conceptually, without attempting to illustrate its relatively complex interconnect. Figure 2(c) [Kahle et al. 2005] illustrates the floor plan of an implementation of the Cell, showing there is space for 4 Power CPUs if they replaced the 8 vector units, and about double the existing L2 cache.

#### 4.4 SMP-SoC vs. Uniprocessor

In the case of comparing SMP-SoC with an aggressive multiprocessor, the basis for design alternatives is wider since a competing uniprocessor could be of arbitrary complexity. For this reason, variations on uniprocessors are split into 4 times, 8 times and 16 times the throughput of each processor in the SMP-SoC design. These are relatively large differences, unlikely to be achievable given known limits on ILP. However, they do serve to mark out boundaries of the design space.

Considering the given design parameters, Figures 3 and 4 illustrate respectively the effects of a faster uniprocessor versus a 4-processor SMP, with relatively low and relatively high miss rates. Miss rates for the SMP case are averaged over all processors.

Figure 3 illustrates that with the same miss rate as compared with a uniprocessor design, the 4-processor SMP design has similar overall performance to a uniprocessor 4 times as fast. Intuitively, although the SMP miss cost is higher, the number of times the worst case (the full throughput is stalled) is lower, because each SMP processor can continue during a miss on another processor, unless there is contention for the bus. Doubling the miss rate has relatively little effect – the SMP design is slower where around 50% or fewer DRAM accesses result

<sup>2</sup><http://os.inf.tu-dresden.de/L4/Linux0nL4/>

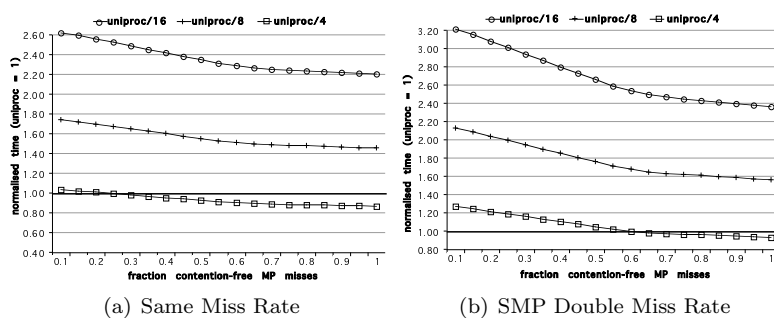


Figure 3. Speed comparison: Uniprocessor 0.001 miss rate. *Normalized to uniprocessor = 1 for the specific uniprocessor being compared in each case (“uniproc/ $n$ ” means SMP numbers as compared with a uniprocessor with  $n$  times the instruction throughput rate).*

in contention. A uniprocessor more than 4 times the speed of each processor in the SMP design is faster in all cases.

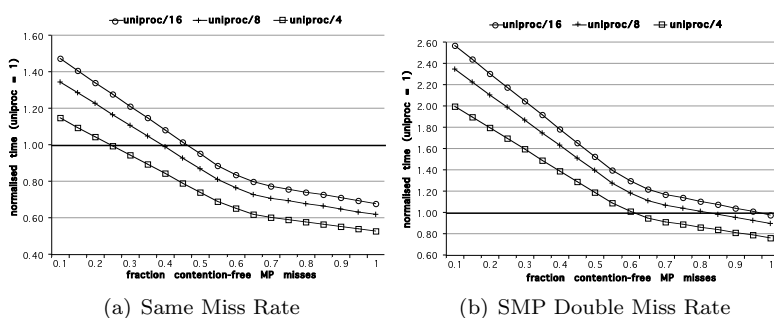


Figure 4. Speed comparison: Uniprocessor 0.01 miss rate. *Normalized to uniprocessor = 1 for the specific uniprocessor being compared in each case.*

On the other hand, with a higher miss rate as in Figure 4, the SMP design is faster across a range of cases than even a uniprocessor 16 times the speed of each processor in the SMP design. If the miss rate is the same across the two designs, only when the number of contention-free bus operations drops below about 45% is the fastest uniprocessor faster than the SMP. If the SMP has double the miss rate, a uniprocessor 4 times the speed of each SMP processor is slower for up to 60% of SMP DRAM access resulting in contention.

These results illustrate that there is a significant area of the design space where an SMP can be faster than an aggressive uniprocessor particularly when miss rates are relatively high. Increasing miss rates can be seen as a proxy for increasing the CPU-DRAM speed gap since the overall effect is the same, so these results make a case for increased effectiveness of multiprocessor systems as the CPU-DRAM speed gap grows.

#### 4.5 Possible Uses

The proposed SMP-SoC design could be used in some of the intended applications for the Cell. While the Cell would potentially be more competitive in specialist niches like HDTV, it is hard to imagine game designers being able to exercise the vector units efficiently. Graphics operations can use them, but other parts of the game implementation space including AI and simulating real-world physics would be much harder to implement efficiently on the Cell architecture. On the other hand, a more general SMP design could still be useful for graphics operations (though possibly less so than a vector unit) but much easier to program for simulating physics.

In broader use, it could be a building block for large-scale systems. In much the same way as the Stanford Hydra project builds a large-scale system out of chip multiprocessors, SMP-SoC could be a building block for similar systems. The major difference is that some rudimentary I/O capability is included in the SMP-SoC design, as well as some kind of main memory. Also, SMP-SoC is aimed at mass-market applications with potential expansion to high-end systems, whereas the CMP aspect of Hydra starts from a design for high-end computation.

As the comparison with a uniprocessor illustrates, provided the workload results in a reasonably small fraction of contending memory accesses, an SMP design could be an effective workaround for the memory wall. The kind of scenario where relatively low contention for memory access is most likely is in running dissimilar processes,

rather than a shared-memory application, which must inherently have contention as a consequence of sharing information through shared memory.

#### 4.6 Summary

The amount of logic required for the Cell design could be used to better effect for a wider range of applications if the design were reconfigured to include 4 general processors, instead of 1 general processor, and 8 vector engines. For more general use, a smaller number of processors with more cache (or a big on-chip static RAM main memory, as envisaged in the RAMpage model [Machanick et al. 1998; Machanick 2000; 2004]) could be better design trade-off. The PicoServer study has shown that a very conservative processor design can result in 8 or more cores fitting onto a modest-sized die, with very low power use, provided that advantage can be taken of DRAM in the same package [Kgil et al. 2006].

### 5. CONCLUSIONS

#### 5.1 Summary

The Cell design illustrates what can be put on one chip with current technology, with drivers like HDTV and games consoles. In contrast to whole-computer-oriented CPU design, where higher-end systems have driven development, the low end is driving designers to explore new parts of the design space.

A design of more general applicability would mean a wider range of applications, a wider range of developers and a wider range of development tools.

The proposed SMP-SoC alternative would not be any more difficult to implement than Cell. It needs less exotic memory technology [Dhong et al. 2005] since it doesn't require memory accesses to keep up with a vector engine. It does not require a novel programming model with consequent risks of failure to achieve anything close to its theoretical potential.

A design similar to that proposed here could be produced fairly quickly by the Cell design team: other than stripping out the extra logic to support multiple threads, the existing CPU design could be used, and replicated. Memory organisation, while different, is comparatively simple. A relatively simple interconnect – a shared bus – could be used, with a simple bus snooping protocol.

#### 5.2 Way Ahead

Very little work has been done on low-end multiprocessor performance issues, since multiprocessor design has historically been aimed at high-end systems.

A range of issues including speed, scalability to larger designs, power management and trade-offs such as amount of on-chip RAM versus number of processors need to be explored.

Demands of embedded and mobile devices imply different approaches to benchmarking. SPEC workloads represent typical workstation or server applications. Real-time and energy-oriented benchmarks are more applicable to these design spaces. Evaluation of SMP-SoC as a whole-computer part requires whole-system benchmarking, an area for which tools exist [Binkert et al. 2003; Magnusson et al. 2002], but where little work has been published.

Recent work on throughput engines [Kongetira et al. 2005; Krewell 2005b; Kgil et al. 2006] has illustrated the potential of this approach; all that's left is to move it to the mainstream, rather than focussing on niches like web servers or internet infrastructure.

#### 5.3 Overall Conclusions

This paper has examined possibilities for an alternative and potentially more general use of chip space to the Cell design. Several historical limitations to SMP systems could potentially be addressed by an SoC design. At the same time Cell has some design features which appear to fly in the face of logic, given historical precedents.

It will be interesting to see if the Cell manages to catch up the lead already established by the Xbox 360. If history is a guide, it will not.

Further investigation of the SMP-SoC idea appears worth pursuing.

#### ACKNOWLEDGMENTS

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

## REFERENCES

- BINKERT, N. L., HALLNOR, E. G., AND REINHARDT, S. K. 2003. Network-oriented full-system simulation using M5. In *Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*. 36–43.
- BROWN, J. 2005. Application-customized cpu design: The Microsoft Xbox 360 CPU story. In *Papers from the Fall Processor Forum 2005*. <http://www.ibm.com/developerworks/library/pa-fpfxbox/>.
- CHERITON, D., GOOSEN, H., AND BOYLE, P. 1989. Multi-level shared caching techniques for scalability VMP-MC. In *Proc. 16th Int. Symp. on Computer Architecture (ISCA '89)*. Jerusalem, 16–24.
- CHERITON, D. R., GOOSEN, H. A., AND BOYLE, P. D. 1991. ParaDiGM: A highly scalable shared-memory multicomputer architecture. *Computer* 24, 2 (February), 33–46.
- DEY, S., LAHIRI, K., AND RAGHUNATHAN, A. 2005. Design of communication architectures for high-performance and energy-efficient systems-on-chips. In *Multiprocessor Systems-on-Chips*, A. Jerraya and W. Wolf, Eds. Morgan Kaufman, San Francisco, 187–222.
- DHONG, S. H., TAKAHASHI, O., WHITE, M., ASANO, T., NAKAZATO, T., SILBERMAN, J., KAWASUMI, A., AND YOSHIMURA, H. 2005. A 4.8GHz fully pipelined embedded SRAM in the streaming processor of a CELL processor. In *ISSCC 2005 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*. San Francisco, 486–487,612.
- ENSLow, P. H., Ed. 1974. *Multiprocessors and Parallel Processing*. John Wiley, New York.
- HAMMOND, L., HUBBERT, B. A., SIU, M., PRABHU, M. K., CHEN, M., AND OLUKOTUN, K. 2000. The Stanford Hydra CMP. *IEEE Micro* 20, 2 (March/April), 71–84.
- JERRAYA, A. AND WOLF, W., Eds. 2005a. *Multiprocessor Systems-on-Chips*. Morgan Kaufman, San Francisco.
- JERRAYA, A. AND WOLF, W. 2005b. The what, why, and how of MPSoCs. In *Multiprocessor Systems-on-Chips*, A. Jerraya and W. Wolf, Eds. Morgan Kaufman, San Francisco, 1–18.
- KAHLE, J. A., DAY, M. N., HOFSTEE, H. P., JOHNS, C. R., MAEURER, T. R., AND SHIPPY, D. 2005. Introduction to the cell multiprocessor. *IBM Journal of Research and Development* 49, 4/5 (July-September), 589–604.
- KGIL, T., D'SOUZA, S., SAIDI, A., BINKERT, N., DRESLINSKI, R., REINHARDT, S., FLAUTNER, K., AND MUDGE, T. 2006. Picoserver: Using 3D stacking technology to enable a compact energy efficient chip multiprocessor. In *Proc. 12th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. San Jose, CA. To appear.
- KONGETIRA, P., AINGARAN, K., AND OLUKOTUN, K. 2005. Niagara: A 32-way multithreaded Sparc processor. *IEEE Micro* 25, 2 (Mar.), 21–29.
- KREWELL, K. 2005a. Cell moves into the limelight. *Microprocessor Report*.
- KREWELL, K. 2005b. A new MIPS powerhouse arrives. *Microprocessor Report*.
- LIEDTKE, J. 1996. Toward real microkernels. *Commun. ACM* 39, 9, 70–77.
- LO, J. L., EMER, J. S., LEVY, H. M., STAMM, R. L., TULLSEN, D. M., AND EGGERS, S. J. 1997. Converting thread-level parallelism to instruction-level parallelism via simultaneous multithreading. *ACM Transactions on Computer Systems (TOCS)* 15, 3, 322–354.
- MACHANICK, P. 2000. Scalability of the RAMPAGE memory hierarchy. *South African Computer Journal* 25 (August), 68–73.
- MACHANICK, P. 2004. Initial Experiences with Dreamy Memory and the RAMPAGE Memory Hierarchy. In *Proc. Ninth Asia-Pacific Computer Systems Architecture Conf.* Beijing, 146–159.
- MACHANICK, P. AND PATEL, Z. 2003. L1 Cache and TLB Enhancements to the RAMPAGE Memory Hierarchy. In *Proc. Eighth Asia-Pacific Computer Systems Architecture Conf.* Aizu-Wakamatsu City, Japan, 305–319.
- MACHANICK, P., SALVERDA, P., AND POMPE, L. 1998. Hardware-software trade-offs in a Direct Rambus implementation of the RAMPAGE memory hierarchy. In *Proc. 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*. San Jose, CA, 105–114.
- MAGNUSSON, P. S., CHRISTENSSON, M., ESKILSON, J., FORSGREN, D., HÅLLBERG, G., HÖGBERG, J., LARSSON, F., MOESTEDT, A., AND WERNER, B. 2002. Simics: A full system simulation platform. *Computer* 35, 2 (February), 50–58.
- MULLER, H. L., STALLARD, P. W. A., AND WARREN, D. H. D. 1996. Multitasking and multithreading on a multiprocessor with virtual shared memory. In *Proc. 2nd Int. Symp. on High-Performance Computer Architecture*. IEEE Computer Society Press, San Jose, California, 212–221.
- OLUKOTUN, K., NAYFEH, B. A., HAMMOND, L., WILSON, K., AND CHANG, K. 1996. The case for a single-chip multiprocessor. In *Proc. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-7)*. Cambridge, MA, 2–11.
- PHAM, D., ASANO, S., BOLLIGER, M., DAY, M. N., HOFSTEE, H. P., JOHNS, C., KAHLE, J., KAMEYAMA, A., KEATY, J., MASUBUCHI, Y., RILEY, M., SHIPPY, D., STASIAK, D., SUZUOKI, M., WANG, M., WARNOCK, J., WEITZEL, S., WENDEL, D., YAMAZAKI, T., AND YAZAWA, K. 2005. The design and implementation of a first-generation CELL processor. In *ISSCC 2005 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*. San Francisco, 184–185,592.
- SAULSBURY, A., PONG, F., AND NOWATZYK, A. 1996. Missing the memory wall: the case for processor/memory integration. In *Proc. 23rd annual Int. Symp. on Computer Architecture*. Philadelphia, Pennsylvania, United States, 90–101.
- WALL, D. 1991. Limits of instruction level parallelism. In *Proc. 4th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-4)*. Santa Clara, CA, 176–188.
- WOLF, W. 2005. Building the software radio. *Computer* 38, 3 (March), 87–89.
- WULF, W. AND MCKEE, S. 1995. Hitting the memory wall: Implications of the obvious. *Computer Architecture News* 23, 1 (March), 20–24.
- YANG, P., MARCHAL, P., WONG, C., HIMPE, S., CATTHOOR, F., DAVID, P., VOUNCKX, J., AND LAUWEREINS, R. 2005. Cost-efficient mapping of dynamic concurrent tasks in embedded real-time multimedia systems. In *Multiprocessor Systems-on-Chips*, A. Jerraya and W. Wolf, Eds. Morgan Kaufman, San Francisco, 313–335.