Power Efficient Processor Architecture and The Cell Processor

H. Peter Hofstee
IBM Server & Technology Group
hofstee@us.ibm.com

Abstract

This paper provides a background and rationale for some of the architecture and design decisions in the Cell processor, a processor optimized for compute-intensive and broadband rich media applications, jointly developed by Sony Group, Toshiba, and IBM.

1. Introduction

The paper is organized as follows. Section 2 discusses some of the challenges microprocessor designers face and provides motivation for performance per transistor as a reasonable first-order metric for design efficiency. Section 3 discusses common microarchitectural enhancements relative to this metric. Section 4 discusses some alternate architectural choices that improve both design efficiency and peak processor performance. Section 5 discusses some of the limitations of the architectural choices introduced in section 3, and proposes a non-homogeneous SMP as a means to overcome these limitations. Section 6 summarizes the organization of the Cell processor.

2. Performance per transistor as a metric

Microprocessor architects and micro-architects have over the last couple of decades been driven by two primary metrics that determine performance: performance per cycle (often approximated by the number of instructions completed per processor cycle), and design frequency (e.g. design cycle time measured in fanout-4 inverter delays). Combined with the capabilities of the technology (e.g. pico-seconds per fo4) and system constraints (e.g. sorting conditions, power supply variation, reference clock jitter, and thermal conditions) these determine the final operating frequency and performance of the end product.

Today, architects and micro-architects, as well as logic and circuit designers, must take power efficiency into account, since virtually all systems, from mobile platforms to PCs and workstations to the largest supercomputers are now power limited. This implies that we must use power efficiency as one of our primary metrics for, and driver of, microprocessor designs.

A number of metrics for efficiency have been proposed, ranging from energy per operation to energy-delay, to energy-delay^2. Each of these metrics balances processor performance to efficiency, and each of these metric can be appropriate [1]. For this paper, however, we examine performance per transistor as a metric. This metric approximates performance per Watt if one assumes a constant per-transistor power penalty. This is reasonable when a high-performance CMOS technology is used and a constant fraction of the power is lost to sub-threshold leakage and gate oxide tunneling currents, and when the intent is to optimize sustained performance when a significant fraction of the chip is being used.

3. Architectural efficiency

Intel's Pat Gelsinger has observed that in recent history we have gained approximately 40% in (design) performance every time the number of available transistors was doubled in a next major CMOS technology node [2]. A corollary is that, on a performance per transistor metric, microprocessors have become less efficient at the same rate. We examine some of the major factors that have driven down performance per transistor.

3.1. Virtual memory and caches

Almost all modern high-performance microprocessors devote the majority of transistors to caches and other structures in support of maintaining the illusion of a large, uniformly accessible, memory. As a rule of thumb cache miss rates are inversely proportional to the square root of their size, thus to first order confirming Gelsinger's law. The illusion of a flat and uniformly accessible memory is however increasingly costly to maintain. Latencies to main memory, in spite of designer's best efforts range in the several hundreds of cycles and approach a thousand cycles in multi-GHz SMP systems. With such a large penalty associated with a cache miss, managing memory locality becomes the main factor determining software performance, and writers of compilers and high-performance software alike spend much of their time reverse-engineering and defeating the sophisticated mechanisms that automatically bring data on to and off the chip. Given the large number of transistors devoted to these mechanisms this is an unsatisfactory situation.

Proceedings of the 11th Int'l Symposium on High-Performance Computer Architecture (HPCA-11 2005)
1530-0897/05 $20.00 © 2005 **IEEE**

IEEE COMPUTER SOCIETY

3.2. Superscalar architectures

The addition of execution units of the same type in order to leverage instruction level parallelism and increase IPC is generally inefficient relative to a performance per transistor metric. As an example, a second load-store port on a cache tends to double its size (a two-ported SRAM cell is often more than twice as big as a single-ported cell), and introduces the need to add logic to maintain the program order between loads and stores. Hence often the cost of the associated resources is more than doubled. In addition, the second load-store issue slot can be used less often than the first, further degrading the efficiency.

3.3. Out-of order instruction processing

Out of order processing of instructions, speculative execution, and register renaming introduce very significant amounts of overhead to keep track of all the instructions and their dependencies and ensure that the computed results are consistent with the in-order semantics. Floorplans of modern microprocessors show that the associated structures tend to far exceed the size of the processor dataflows.

3.4. Hardware branch prediction

Modern branch prediction structures rely on large tables to store branch histories and cache branch target addresses or instructions for incremental increases in performance and thus tend to degrade performance per transistor.

3.5. Hyper-pipelining

Constant penalties associated with unpredictable changes in instruction flow imply that increasing pipeline depth as a means to improve design frequency provides diminishing returns. Recent studies have shown that increasing pipeline depths beyond their current size in today's high-performance microprocessors provides only a small return in terms of performance and a negative return if power is taken into account [3], consistent with the broader thesis of this paper. It should be noted, however, that pipeline depth and optimal processor design frequency is a strong function of both the workload set and the circuit family that is used. Architectures that are more optimized to their applications can usefully achieve higher design frequencies than are shown as optimal by [3]. Also technologies optimized for sustained performance, with substantial transistor leakage currents, favor higher design frequencies more strongly than technologies that are optimized for battery life.

4. Increasing efficiency and performance

Section 3 discussed some of the mechanisms that have led to decreased microprocessor efficiencies. This section discusses architectural and micro-architectural mechanisms that improve design efficiency.

4.1. Multi-core processors

The most obvious way to improve efficiency is to sacrifice per-thread performance (or per-thread performance growth) and instead instantiate multiple cores on a single chip when more transistors become available. The more threads can be accommodated in the application set, the more efficient the processors can become. The approach allows architects to re-introduce simpler processor micro-architectures (e.g. in-order or scalar) in order to re-gain efficiency. This approach forces programmers to multithread their applications, but because server systems have been organized as SMPs for a long time, there is good software support for this model. Many applications scale well on SMP systems and may scale even better for SMPs on a chip. All major microprocessor vendors now offer or plan to offer chip multiprocessors following this model. Even in the face of a slowdown in transistor performance growth, this path allows for continued increases in chip performance at historical rates for at least a decade if accelerated off-chip bandwidth needs are addressed [4].

4.2. Three-level storage

RISC microprocessors bring data from main memory into registers before operating on the data. Register "memory" is managed by the compiler for conventional programming languages. We propose that since the best compilers and the best programmers already manage the caches in software, it is reasonable to postulate that it may be easier for programmers and compilers to deliver power-constrained performance by pre-scheduling transfers of code and data to local storage than to find ways of compensating for the thousand cycle penalties when data or code is not resident in the cache. By allowing the pre-scheduled data and code transfers to occur asynchronously, in parallel with computation, another important limitation of conventional architectures, known as the "memory wall" can be addressed. In order to leverage the bandwidth to a main memory that has a latency of, say, 1K cycles, and transfer granule of, say, 8 cycles, 128 transfers need to be pipelined to fully leverage the available bandwidth. Conventional microprocessors need to use speculative execution, or speculative prefetching to create more than

1530-0897/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

a single memory transfer per thread with strongly diminishing returns. Of course increasing the number of threads on a chip helps, but the more cores are present on the bus, the longer SMP and protocols take, further increasing memory latency.

Performance improvement on structured and compute intensive codes following the 3-level model is immediate. New programming paradigms such as streaming languages are also easily supported. We anticipate that it will take some time before compilers will mature to exceed the performance of single-thread conventional (legacy) programs on conventional architectures, but it may not take very long before a chip-level (multi-thread) power-performance advantage is achieved on a broad set of applications.

4.3. Large unified register files

In order to support high cycle times (low fo4) and relatively deep pipelines efficiently, enough registers need to be available to support the instructions in flight. A large register file is a more efficient means of achieving this objective than using register renaming to extend a smaller architectural register set. An additional advantage of a large set of named registers is that extensive loop unrolling (and interleaving) can be supported without the need for hardware to reorder instructions. Unrolling of loops and interleaving of loop iterations is well within the capabilities of modern compilers.

The cost of a large register file can be further reduced by leveraging a single register file and single name space for all instruction types as well as special purpose registers (condition registers, link, and count registers) rather than dedicating separate files for scalar, floating-point and media/SIMD.

4.4. Software branch prediction

Whereas modern hardware branch prediction structures do not do well on performance per transistor, software branch prediction tends to be an inexpensive means to gain a substantial performance improvement. In order to support high-frequency deeply pipelined designs efficiently, a branch hint instruction, present in the code well before the actual branch instruction allows for "just in time" prefetching of the instructions at the branch target. In combination with loop unrolling and interleaving, compilers can achieve near optimal performance even on deeply pipelined high-frequency processors.

4.5. SIMD dataflow organization

In a SIMD dataflow instruction decode and control overhead is amortized over multiple instructions in parallel. A popular SIMD organization in PC processors is a 128-bit wide dataflow that can be used in a variety of ways, including 4 wide for 32 bit integer or floating-point. Even though some compilers can generate SIMD instructions from generic code, SIMD units are most commonly and effectively used by using SIMD data-types explicitly in the source code and restructure algorithms accordingly.

Another approach that amortizes control overhead is (long) vector instructions. This organization is well known in the science and supercomputing community but is not as well represented in the software base for media and personal computing applications.

5. Non-homogeneous SMPs

The previous section outlined some of the approaches that can be taken to improve performance per transistor in microprocessors while maintaining a high degree of programmability. However, the organization is not well suited for all applications. One shortcoming is the large state that is introduced with local storage and large register files. Hence the organization is not particularly well suited for applications with very frequent context switches, such as operating systems.

A preferred approach therefore is to combine processors optimized for performance per transistor on compute intensive applications, with processors with a more conventional architecture to run the operating system and more control intensive applications. Since operating systems are amongst the most expensive and extensive software, an organization that does not unnecessarily forces creating a new OS has substantial cost and schedule advantages.

6. The Cell processor

The Cell processor incorporates many of the listed ideas to improve microprocessor efficiency. On the Cell processor a Power Architecture™ core is combined with Synergistic Processor Elements and associated memory transfer mechanisms. Memory translation and protection are consistent with the Power Architecture™. In addition facilities are added to enhance the real-time behavior of the processor. The 64-b Power processor supports multiple operating systems and virtualization. In particular real-time operating systems (such as an embedded or game OS) and non-real time OS (such as Linux) can run simultaneously. The Power processor virtualization layer (logical partitioning) governs the allocation of resources, including the SPEs and other resources to the various OS partitions. Unlike Power processors, the SPEs operate only on their local memory (local store or LS). Code and data must be transferred into

1530-0897/05 $20.00 © 2005 IEEE

IEEE COMPUTER SOCIETY

the associated LS for an SPE to execute or operate on. Local Store addresses do have an alias in the Power processor address map and transfers to and from Local Store to memory at large (including other local stores) are coherent in the system. As a result a pointer to a data structure that has been created on the Power processor can be passed to an SPE and the SPE can use this pointer to issue a DMA command to bring the data structure into its local store in order to perform operations on it. If after operating on this data structure the SPE (or Power core) issues a DMA command to place it back in non-LS memory, the transfer is again coherent in the system according to the normal Power memory ordering rules. Equivalents of the Power processor locking instructions, as well as a memory mapped mailbox per SPE, are used for synchronization and mutual exclusion.

Figure 1 shows the organization of the first-generation Cell processor [5]. We summarize the highlights. The processor has an 11fo4 design frequency. The processor is constructed around a high bandwidth on-chip SMP fabric capable of supporting up to 96 bytes per processor cycle total using up to 12 simultaneous transfers. Connected to this SMP fabric are:

(1) A dual threaded (SMT) 64b Power core. This dual-issue core has 32kB L1 instruction and data caches, a 512 kB L2 cache, and supports the VMX SIMD instruction set.

(2) Eight Synergistic Processor Elements, described in more detail in Figure 2.

(3) An on-chip memory controller. This memory controller supports high memory bandwidth using two Rambus XDRAM™ memory banks.

(4) A controller supporting off chip coherent and I/O interfaces. These interfaces support partitioning of an aggregate of 7 Bytes outbound and 5 Bytes inbound Rambus RRAC raw bandwidth in Byte increments over two external interfaces, on of which can be configured as a coherent bus. This flexibility supports multiple system configurations including a glueless two-way SMP.

Each of the listed elements has 8 byte wide (relative to processor frequency) inbound and outbound interfaces to the coherent on-chip bus, except the I/O interface unit which has two 8 byte interfaces.

Figure 2 shows the high-level organization of the SPE. The SPE processor incorporates many of the characteristics that allow for it to achieve high frequency and near optimal CPI in a small area. The SPE uses the three level model of memory we discussed in section 4.2. Details of the SPE can be found in [6], we summarize the highlights. The dual issue SPE consists of an SPU with a 256kB local store memory and an interface unit that supports up to 16 simultaneous transfers to or from the local stores. The SPU has a unified 128 bit by 128 entry register file and a 128-bit wide dataflow that supports integer and floating-point instructions of various SIMD widths.

Conclusion

Power limitations and limitations on main memory access latency force a re-evaluation of microprocessor architecture. This paper lists a number of mechanisms that have significantly degraded efficiency and also discusses a number of mechanisms to increase efficiency that are not in use in today's high performance processors. The Cell processor incorporates many of these, and combines power efficiency with a very high design frequency.

Acknowledgements

The Cell processor is the result of a deep collaboration by engineers from IBM, Sony Computer Entertainment, Toshiba Corporation, and Sony Corporation. Each brought unique requirements to the table that helped shape this microprocessor. Ken Kutaragi should be credited with presenting the challenge and with insisting that we explore something beyond the boundaries of the conventional. Jim Kahle provided the technical leadership for the project together with Masakazu Suzuoki, Haruyuki Tago, and Yoshio Masubuchi. Jim also led the Cell architecture team consisting of Michael Day, Charles Johns, Dave Shippy, Takeshi Yamazaki, Shigehiro Asano, Andy Wottreng and myself. Sang Dhong provided the technical leadership for the SPE processor and drove its efficiency to reality. The development of the micro-architecture of the SPE was led by Brian Flachs. Other key contributors to the SPE are listed on Brian's paper. Marty Hopkins ("no great architectural idea has ever gone unpunished") did his best to ensure the SPE would be a good target for a compiler. Michael Gschwind and Peter Capek also made contributions in this area. Sumedh Sathaye, J.D. Wellman, and Ravi Nair contributed ideas that eventually led to the organization of this processor as a non-homogeneous SMP. Several people from all three companies provided direction from an application perspective, of these Nobuo Sasaki, Yukio Watanabe, Brad Michael and Barry Minor had the greatest impact on the organization of the SPE. Dac Pham was the chief engineer driving the design of the Cell processor, and Mary Many coordinated the design activities. Chekib Akrout and senior management in the three companies provided management oversight and created the right business conditions for this project. With more than 400 engineers and managers making contributions to this processor it is impossible to acknowledge everyone, but it should be noted that the level of effort that everyone has put into the project is what turns a reasonable set of ideas into a great processor.

1530-0897/05 $20.00 © 2005 **IEEE**

IEEE COMPUTER SOCIETY

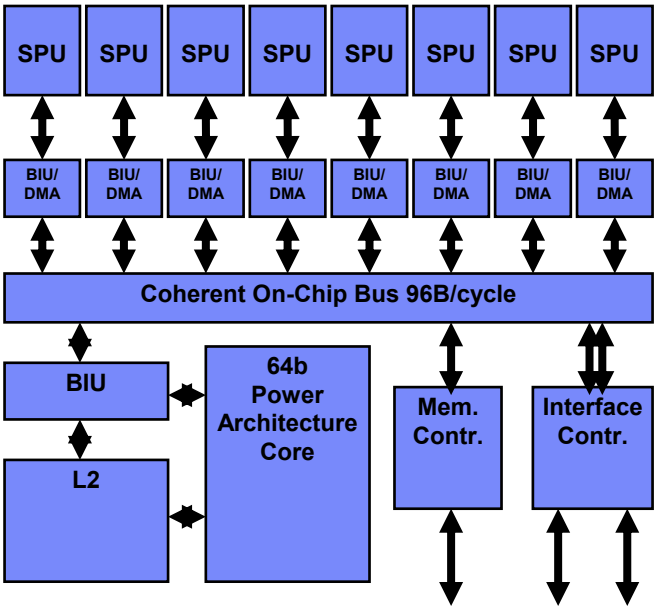

Figure 1. 1st Generation Cell components.

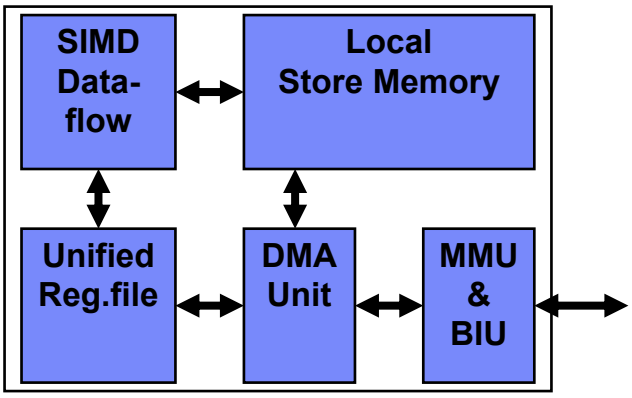

Figure 2. High-level SPE organization.

References

[1] H. P. Hofstee, "Power-Constrained Microprocessor Design", *2002 IEEE International Conference on Computer Design*, Sep. 2002, pp. 14-16.

[2] F. Pollack, "Microarchitecture challenges in the coming generations of CMOS process technologies," *32nd Annual International Symposium on Microarchitecture*, (MICRO-32) Nov. 1999, pp. 2-.

[3] H. P. Hofstee, "Future Microprocessors and Off-Chip SOP Interconnect," *IEEE Transactions on Advanced Packaging*, Vol. 27, No. 2, May 2004, pp.301-303.

[4] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V.Zyuban, P.N. Strenski, and P.G. Emma "Optimizing pipelines for power and performance," in *Conf. Proc. 35th Annual IEEE/ACM International Symposium on Microarchitecture 2002*. pp. 333-344.

[5] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, K. Yazawa, "The Design and Implementation of a First-Generation CELL Processor," *to appear: IEEE International Solid-State Circuits Symposium,* Feb. 2005.

[6] B. Flachs, S. Asano, S. H. Dhong, H. P. Hofstee. G. Gervais, R. Kim, T. Le, P. Liu, J. Leenstra, J. Liberty, B. Michael, H-J. Oh, S. M. Mueller, O. Takahashi, A. Hatakeyama, Y. Watanabe, N. Yano. "The Microarchitecture of the Streaming Processor for a CELL Processor," *to appear: IEEE International Solid-State Circuits Symposium*, Feb. 2005.

1530-0897/05 $20.00 © 2005 **IEEE**

IEEE COMPUTER SOCIETY