The Challenges of Multi-Core Processor

Yaser Ahangari Nanehkaran[1], Sajjad Bagheri Baba Ahmadi[2]

[1]Department of Information Technology, Çankaya University, Ankara, Turkey, PH: +905545058732, Email: y.ahangari@yahoo.com
2Department of Computer Engineering, Çankaya University, Ankara, Turkey, PH: +905436977943, Email: sajad_bagheri67@yahoo.com

ABSTRACT

One of the most important trend of increasing the speed of processor to get a boost in performance is Multi-core. Multi-core processors are the new direction manufacturers are focusing on. A multi-core processor has many advantages especially for those looking to boost their multitasking computing power of system. These kinds of processors provide few complete execution cores instead of one, each with an independent interface to the front side bus. Since each core has its own cache, so the only one operating system which has sufficient resources and provides a noticeable improvement to multitasking can handle intensive tasks in parallel. But in fact there are some pluses and minuses when we add new cores. In this article, our main goal is describe some of the important challenges of multi-core. In addition, the paper describes its basic concept, advantages, and a sample of Dual-core Processors in Intel and AMD.

Keywords: Chip multiprocessor, Hyper Transport, printed circuit board, front side bus, multithread, DRAM memory, and cache.

1 INTRODUCTION

A multi-core processor is a single computing component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions. The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing. Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package.

Processors were originally developed with only one core. A dual-core processor has two cores (e.g. AMD Phenom II X2, Intel Core Duo), a quad-core processor contains four cores, a hexa-core processor contains six cores (e.g. AMD Phenom II X6, Intel Core i7 Extreme Edition 980X), an octo-core processor or octa-core processor contains eight cores (e.g. Intel Xeon E7-2820, AMD FX-8350), a deca-core processor contains ten cores (e.g. Intel Xeon E7-2850). A multi-core processor implements multiprocessing in a single physical package. Designers may couple cores in a multi-core device tightly or loosely. For example, cores may or may not share caches, and they may implement message passing or shared memory inter-core communication methods. Common network topologies to interconnect cores include bus, ring, two-dimensional mesh, and crossbar. Homogeneous multi-core systems include only identical cores, heterogeneous multi-core systems have cores that are not identical. Just as with single-processor systems, cores in multi-core systems may implement architectures such as superscalar, VLIW, vector processing, SIMD, or multithreading.

Multi-core processors are widely used across many application domains including general-purpose, embedded, network, digital signal processing (DSP), and graphics.

The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be run in parallel simultaneously on multiple cores; this effect is described by Amdahl's law. In the best case, so-called embarrassingly parallel problems may realize speedup factors near the number of cores, or even more if the problem is split up enough to fit within each core's cache(s), avoiding use of much slower main system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in re-factoring the whole problem. The parallelization of software is a significant ongoing topic of research.

2. Intel and AMD Dual-core Processors

Intel and AMD are the mainstream manufacturers of microprocessors. Intel produces many different flavors of Multi-core processors: the Pentium D is used in desktops, Core 2 Duo is used in both laptop and desktop, Core 2 Duo is used in both laptop and desktop environments, and the Xeon processor is used in servers. AMD has the Althon lineup for desktops, Turion for laptops, and Opteron for servers/workstations. Although the Core 2 Duo and Athlon 64 X2 run on the same platforms their architectures differ greatly.

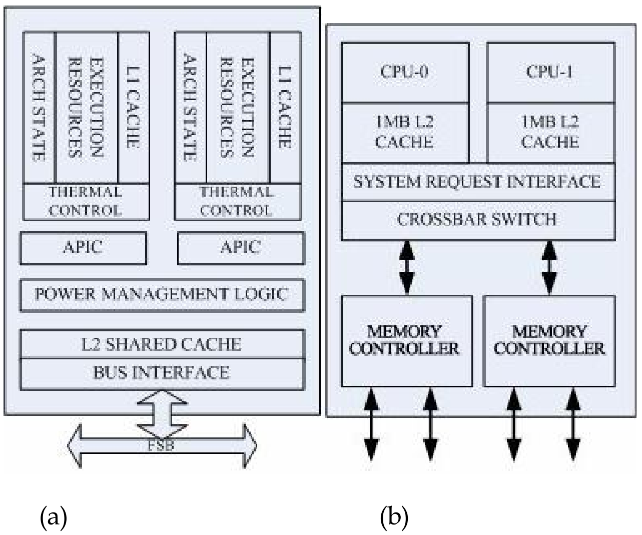

Figure 1 : (a) Intel Core 2 Duo, (b) AMD Athlon 64 X2.

Figure 1 shows block diagrams for the Core 2 Duo and Athlon 64 X2, respectively. Both architectures are homogenous dual-core processors. The Core 2 Duo adheres to a shared memory model with private L1 caches and a shared L2 cache which "provides a peak transfer rate of 96 GB/sec." If a L1 cache miss occurs both the L2 cache and the second cores L1 cache are traversed in parallel before sending a request to main memory. In contrast, the Athlon follows a distributed memory model with discrete L2 caches. These L2 caches share a system request interface, eliminating the need for a bus.

The system request interface also connects the cores with an on-chip memory controller and an interconnect called HyperTransport. HyperTransport effectively reduces the number of buses required in a system, reducing bottlenecks and increasing bandwidth. The Core 2 Duo instead uses a bus interface. The Core 2 Duo also has explicit thermal and power control units on-chip. There is no definitive performance advantage of a bus vs. an interconnect, and the Core 2 Duo and Athlon 64 X2 achieve similar performance measures, each using a different communication protocol.

3. Advantages

The proximity of multiple CPU cores on the same die allows the cache coherency circuitry to operate at a much higher clock-rate than is possible if the signals have to travel off-chip. Combining equivalent CPUs on a single die significantly improves the performance of cache snoop (alternative: Bus snooping) operations. Put simply, this means that signals between different CPUs travel shorter distances, and therefore those signals degrade less. These higher-quality signals allow more data to be sent in a given time period, since individual signals can be shorter and do not need to be repeated as often.

Assuming that the die can fit into the package, physically, the multi-core CPU designs require much less printed circuit board (PCB) space than do multi-chip SMP designs. Also, a dual-core processor uses slightly less power than two coupled single-core processors, principally because of the decreased power required to drive signals external to the chip. Furthermore, the cores share some circuitry, like the L2 cache and the interface to the front side bus (FSB). In terms of competing technologies for the available silicon die area, multi-core design can make use of proven CPU core library designs and produce a product with lower risk of design error than devising a new wider core-design. Also, adding more cache suffers from diminishing returns.

Multi-core chips also allow higher performance at lower energy. This can be a big factor in mobile devices that operate on batteries. Since each core in multi-core is generally more energy-efficient, the chip becomes more efficient than having a single large monolithic core. This allows higher performance with less energy. The challenge of writing parallel code clearly offsets this benefit.

4. Multi-core Challenges

Having multiple cores on a single chip gives rise to some problems and challenges. Power and temperature management are two concerns that can increase exponentially with the addition of multiple cores. Memory/cache coherence is another challenge, since all designs discussed above have distributed L1 and in some cases L2 caches which must be coordinated. And finally, using a multi-core processor to its full potential is another issue. If programmers do not write applications that take advantage of multiple cores there is no gain, and in some cases there is a loss of performance. Application need to be written so that different parts can be run concurrently (without any ties to another part of the application that is being run simultaneously).

4.1. Power and temperature

If two cores were placed on a single chip without any modification, the chip would, in theory, consume twice as much power and generate a large amount of heat. In the extreme case, if a processor overheats your computer may even combust. To account for this each design above runs the multiple cores at a lower frequency to reduce power consumption. To combat unnecessary power consumption many designs also incorporate a power control unit that has the authority to shut down unused cores or limit the amount of power. By powering off unused cores and using clock gating the amount of leakage in the chip is reduced.

To lessen the heat generated by multiple cores on a single chip, the chip is architected so that the number of hot spots does not grow too large and the heat is spread out across the chip. As seen in Figure 2, the majority of the heat in the CELL processor is dissipated in the Power Processing Element and the rest is spread across the Synergistic Processing Elements. The CELL processor follows a common trend to build temperature monitoring into the system, with its one linear sensor and ten internal digital sensors.

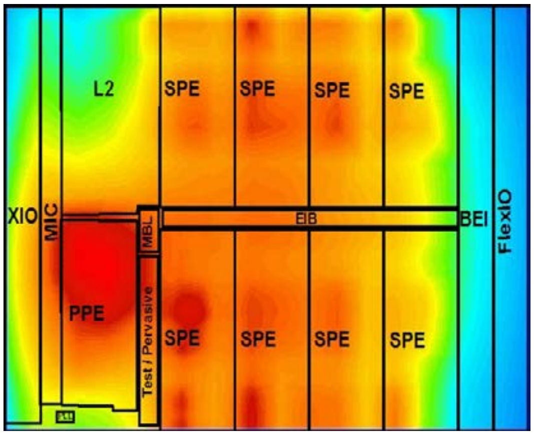

Figure 2: CELL Thermal Diagram

4.2 Cache coherence

Cache coherence is a concern in a multi-core environment because of distributed L1 and L2 cache. Since each core has its own cache, the copy of the data in that cache may not always be the most up-to-date version. For example, imagine a dual-core processor where each core brought a block of memory into its private cache. One core writes a value to a specific location; when the second core attempts to read that value from its cache it will not have the updated copy unless its cache entry is invalidated and a cache miss occurs. This cache miss forces the second cores cache entry to be updated. If this coherence policy was not in place garbage data would be read and invalid results would be produced, possibly crashing the program or the entire computer.

4.3 Multithreading

The last, and most important, issue is using multithreading or other parallel processing techniques to get the most performance out of the multi-core processor. “With the possible exception of Java, there are no widely used commercial development languages with [multithreaded] extensions.”Rebuilding applications to be multithreaded means a complete rework by programmers in most cases. Programmers have to write applications with subroutines able to be run in different cores, meaning that data dependencies will have to be resolved or accounted for (e.g. latency in communication or using a shared cache). Applications should be balanced. If one core is being used much more than another, the programmer is not taking full advantage of the multi- core system. Some companies have heard the call and designed new products with multi-core capabilities; Microsoft and Apples newest operating systems can run on up to 4 cores.

4.3.1 Starvation

If a program is not developed correctly for use in a multi-core processor one or more of the cores may starve for data. This would be seen if a single-threaded application is run in a multi-core system. The thread would simply run in one of the cores while the other cores sat idle. This is an extreme case, but illustrates the problem.

4.3.2 Denial of memory service

All major high-performance processor manufacturers have integrated at least two cores (processors) on the same chipand it is predicted that chips with many more cores will become widespread in the near future. As cores on the same chip share the DRAM memory system, multiple programs executing on different cores can interfere with each others’ memory access requests, thereby adversely affecting one another’s performance.

4.3.3 Interconnect

Another important feature with impact on performance of multicore chips is the communication among different on-chip components: cores, caches, and—if integrated—memory controllers and network controllers. Initial designs used a bus as in traditional multi-CPU systems. The trend now is to use a crossbar or other advanced mechanisms to reduce latency and contention .For instance, AMD CPUs employ a crossbar, and the Tilera TILE64 implements a fast non-blocking multi-link mesh. However, the interconnect can become expensive: an 8 x 8 crossbar on-chip can consume as much area as five cores and as much power as two cores.

With only private caches on-chip, data exchange between threads running on different cores historically necessitated using the off-chip interconnect. Shared on-chip caches naturally support data exchange among threads running on different cores. Thus, introducing a level of shared cache on chip—commonly L2 or, as the more recent trend, L3—or supporting data-exchange short-cuts such as cache-to-cache transfer helped reduce the off-chip traffic. However, more on-chip cache levels put additional requirements on the on-chip interconnect in terms of complexity and bandwidth.

As data processing increases with more thread-level parallelism, demands also typically increase on the off-chip communication fabric for memory accesses, I/O, or CPU-to-CPU communication. To address these requirements, the new trend in off-chip communication is from bus-based towards packet-based, point-to-point interconnects. This concept was first implemented in HyperTransport for AMD CPUs and later in Intel's QuickPath Interconnect. The off-chip interconnect and data-coherency support also impact scalability of multiple-CPU servers.

5. Summary and Conclusion

We report on the basic concept of multi-core processors, a sample of Dual-core Processors in Intel and AMD, and its advantages. In this article also we survey the most important aspects of challenge in this method. However, Before multi-core processors the performance increase from generation to generation was easy to see, an increase in frequency. This model broke when the high frequencies caused processors to run at speeds that caused increased power consumption and heat dissipation at detrimental levels. Adding multiple cores within a processor gave the solution of running at lower frequencies, but added interesting new problems.

Multi-core processors are architected to adhere to reasonable power consumption, heat dissipation, and cache coherence protocols. However, many issues remain unsolved. In order to use a multi-core processor at full capacity the applications run on the system must be multithreaded. There are relatively few applications (and more importantly few programmers with the know-how) written with any level of parallelism. And finally the memory systems and interconnection networks also need improvement.

Acknowledgements

We are obliged to Asst. Prof. Dr Kasım Öztoprak, for the valuable information provided by him in his respective fields. We are grateful for his cooperation.

References

[1] Margaret Rouse (March 27, 2007). "Definition: multi-core processor". TechTarget. Retrieved March 6, 2013.

[2] Aater Suleman (May 19, 2011). "Q & A: Do multicores save energy? Not really.". Retrieved March 6, 2013.

[3] Aater Suleman (May 20, 2011). "What makes parallel programming hard?". FutureChips. Retrieved March 6, 2013.

[4] Bryan Schauer (September, 2008). " Multicore Processors – A Necessity ". ProQuest Discovery Guides.

[5] A.C. Sodan, Jacob Machina, Arash Deshmeh, Kevin Macnaughton, Bryan Esbaugh." Parallelism via Multithreaded and Multicore CPUs". Universidad Rosario, Computer Engineering, 0018-9162$26.00, 2009.

[6] Ryan Shrout (December 2, 2009). "Intel Shows 48-core x86 Processor as Single-chip Cloud Computer". Retrieved March 6, 2013.

[7] Ni, Jun. "Multi-core Programming for Medical Imaging". Retrieved 17 February 2013.

[8] Rick Merritt (February 6, 2008). "CPU designers debate multi-core future". EE Times. Retrieved March 6, 2013.

[9] John Darlinton, Moustafa Ghanem, Yike Guo, Hing Wing To (1996), "Guided Resource Organisation in Heterogeneous Parallel Computing", Journal of High Performance Computing 4 (1).