Multicore Processors – A Necessity

By Bryan Schauer

Abstract

As personal computers have become more prevalent and more applications have been designed for them, the end-user has seen the need for a faster, more capable system to keep up. Speedup has been achieved by increasing clock speeds and, more recently, adding multiple processing cores to the same chip. Although chip speed has increased exponentially over the years, that time is ending and manufacturers have shifted toward multicore processing. However, by increasing the number of cores on a single chip challenges arise with memory and cache coherence as well as communication between the cores. Coherence protocols and interconnection networks have resolved some issues, but until programmers learn to write parallel applications, the full benefit and efficiency of multicore processors will not be attained.

Background

New Scientist Blogs
http://www.newscientist.com/blog/technology/2008_01_01_archive.html

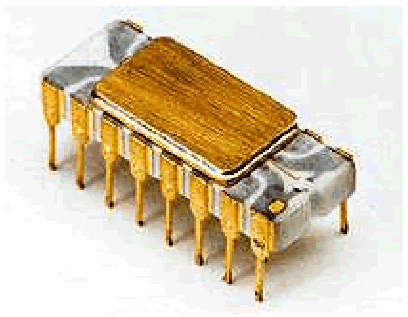
The trend of increasing a processor's speed to get a boost in performance is a way of the past. Multicore processors are the new direction manufacturers are focusing on. Using multiple cores on a single chip is advantageous in raw processing power, but nothing comes for free.

With additional cores, power consumption and heat dissipation become a concern and must be simulated before layout to determine the best floorplan which distributes heat across the chip, while being careful not to form any hot spots. Distributed and shared caches on the chip must adhere to coherence protocols to make sure that when a core reads from memory it is reading the current piece of data and not a value that has been updated by a different core.

With multicore processors come issues that were previously unforeseen. How will multiple cores communicate? Should all cores be homogenous, or are highly specialized cores more efficient? And most importantly, will programmers be able to write multithreaded code that can run across multiple cores?

1.1 A Brief History of Microprocessors

Intel manufactured the first microprocessor, the 4-bit 4004, in the early 1970s which was basically just a number-crunching machine. Shortly afterwards they developed the 8008 and 8080, both 8-bit, and Motorola followed suit with their 6800 which was equivalent to Intel's 8080. The companies then fabricated 16-bit microprocessors, Motorola had their 68000 and Intel the 8086 and 8088; the former would be the basis for Intel's 80386 32-bit and later their popular Pentium lineup which were in the first consumer-based PCs. [18, 19] Each generation of processors grew smaller, faster, dissipated more heat, and consumed more power.



The world's first single-chip processor. Netrino Institute http://www.netrino.com/node/91

1.2 Moore's Law

One of the guiding principles of computer architecture is known as Moore's Law. In 1965 Gordon Moore stated that the number of transistors on a chip will roughly double each year (he later refined this, in 1975, to every two years). What is often quoted as Moore's Law is Dave House's revision that computer performance will double every 18 months. [20] The graph in Figure 1 plots many of the early microprocessors briefly discussed in Section 1.1 against the number of transistors per chip.

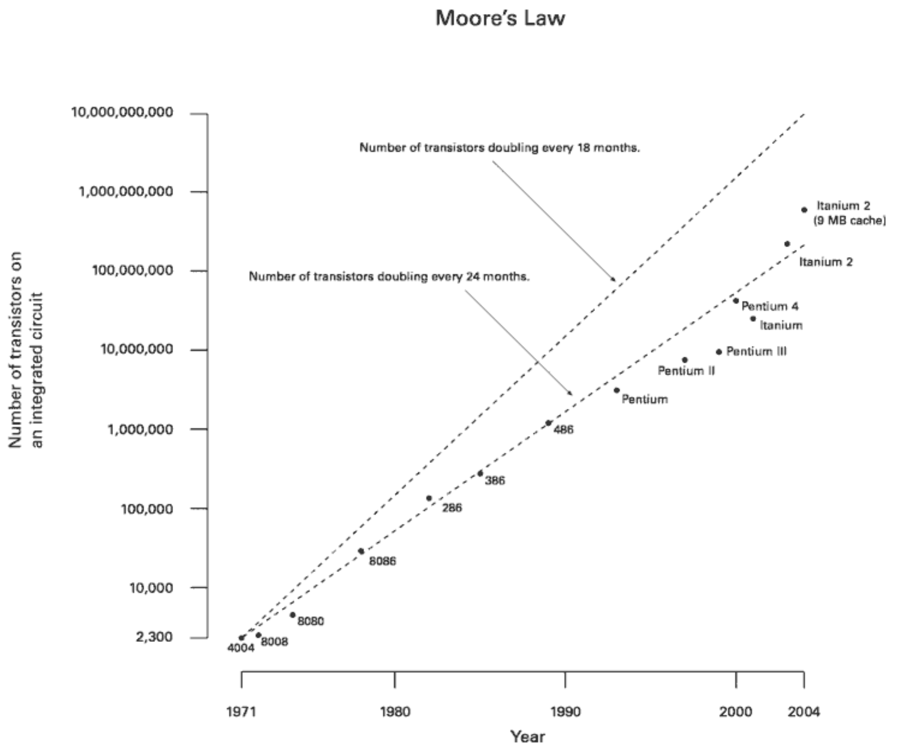


Figure 1: Depiction of Moore's Law [21]

As shown in Figure 1, the number of transistors has roughly doubled every 2 years. Moore's law continues to reign; for example, Intel is set to produce the 'world's first 2 billion transistor microprocessor' "Tukwila" later in 2008. [22] House's prediction, however, needs another correction. Throughout the 1990's and the earlier part of this decade microprocessor frequency was synonymous with performance; higher frequency meant a faster, more capable computer. Since processor frequency has reached a plateau, we must now consider other aspects of the overall performance of a system: power consumption, temperature dissipation, frequency, and number of cores. Multicore processors are often run at slower frequencies, but have much better performance than a single-core processor because 'two heads are better than one'.

1.3 Past Efforts to Increase Efficiency

Apple II, an early personal computer
Solar Navigator
http://www.solarnavigator.net/computers.htm

As touched upon above, from the introduction of Intel's 8086 through the Pentium 4 an increase in performance, from one generation to the next, was seen as an increase in processor frequency. For example, the Pentium 4 ranged in speed (frequency) from 1.3 to 3.8 GHz over its 8 year lifetime. The physical size of chips decreased while the number of transistors per chip increased; clock speeds increased which boosted the heat dissipation across the chip to a dangerous level. [1]

To gain performance within a single core many techniques are used. Superscalar processors with the ability to issue multiple instructions concurrently are the standard. In these pipelines, instructions are pre-fetched, split into sub-components and executed out-of-order. A major focus of computer architects is the branch instruction. Branch instructions are the equivalent of a fork in the road; the processor has to gather all necessary information before making a decision. In order to speed up this process, the processor predicts which path will be taken; if the wrong path is chosen the processor must throw out any data computed while taking the wrong path and backtrack to take the correct path. Often even when an incorrect branch is taken the effect is equivalent to having waited to take the correct path. Branches are also removed using loop unrolling and sophisticated neural network-based predictors are used to minimize the misprediction rate. Other techniques used for performance enhancement include register renaming, trace caches, reorder buffers, dynamic/software scheduling, and data value prediction.

There have also been advances in power- and temperature-aware architectures. There are two flavors of power-sensitive architectures: low-power and power-aware designs. Low-power architectures minimize power consumption while satisfying performance constraints, e.g. embedded systems where low-power and real-time performance are vital. Power-aware architectures maximize performance parameters while satisfying power constraints. Temperature-aware design uses simulation to determine where hot spots lie on the chips and revises the architecture to decrease the number and effect of hot spots.

1.4 The Need for Multicore

Due to advances in circuit technology and performance limitation in wide-issue, super-speculative processors, Chip-Multiprocessors (CMP) or multi-core technology has become the mainstream in CPU designs. [5]

Speeding up processor frequency had run its course in the earlier part of this decade; computer architects needed a new approach to improve performance. Adding an additional processing core to the same chip would, in theory, result in twice the performance and dissipate less heat, though in practice the actual speed of each core is slower than the fastest single core processor. In September 2005 the IEE Review noted that “power consumption increases by 60% with every 400MHz rise in clock speed…But the dual-core approach means you can get a significant boost in performance without the need to run at ruinous clock rates.” [1]

Multicore is not a new concept, as the idea has been used in embedded systems and for specialized applications for some time, but recently the technology has become mainstream with Intel and Advanced Micro Devices (AMD) introducing many commercially available multicore chips. In contrast to commercially available two and four core machines in 2008, some experts believe that “by 2017 embedded processors could sport 4,096 cores, server CPUs might have 512 cores and desktop chips could use 128 cores.” [2] This rate of growth is astounding considering that current desktop chips are on the cusp of using four cores and a single core has been used for the past 30 years.

2. Multicore Basics

The following isn’t specific to any one multicore design, but rather is a basic overview of multicore architecture. Although manufacturer designs differ from one another, multicore architectures need to adhere to certain aspects. The basic configuration of a microprocessor is seen in Figure 2.

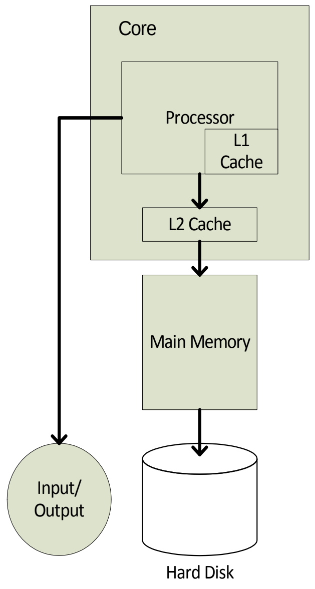
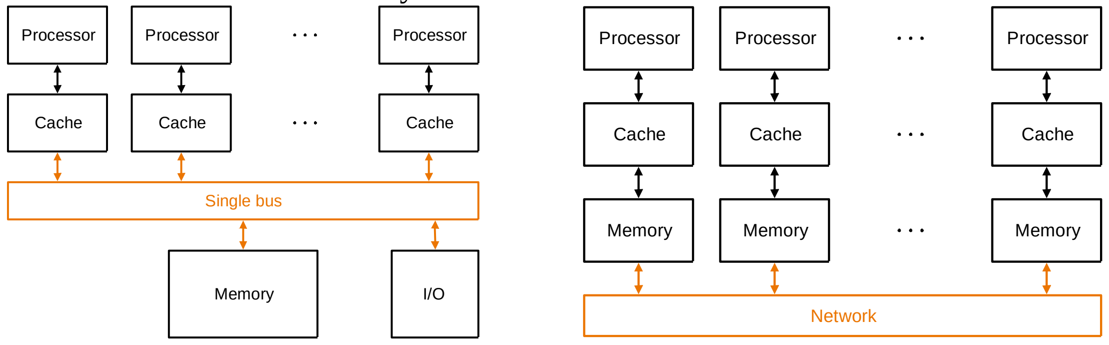


Figure 2: Generic Modern Processor Configuration

Closest to the processor is Level 1 (L1) cache; this is very fast memory used to store data frequently used by the processor. Level 2 (L2) cache is just off-chip, slower than L1 cache, but still much faster than main memory; L2 cache is larger than L1 cache and used for the same purpose. Main memory is very large and slower than cache and is used, for example, to store a file currently being edited in Microsoft Word. Most systems have between 1GB to 4GB of main memory compared to approximately 32KB of L1 and 2MB of L2 cache. Finally, when data isn’t located in cache or main memory the system must retrieve it from the hard disk, which takes exponentially more time than reading from the memory system.

If we set two cores side-by-side, one can see that a method of communication between the cores, and to main memory, is necessary. This is usually accomplished either using a single communication bus or an interconnection network. The bus approach is used with a shared memory model, whereas the interconnection network approach is used with a distributed memory model. After approximately 32 cores the bus is overloaded with the amount of processing, communication, and competition, which leads to diminished performance; therefore, a communication bus has a limited scalability.



(a) (b)

Figure 3: (a) Shared Memory Model, (b) Distributed Memory Model [23]

Multicore processors seem to answer the deficiencies of single core processors, by increasing bandwidth while decreasing power consumption. Table 1, below, shows a comparison of a single and multicore (8 cores in this case) processor used by the Packaging Research Center at Georgia Tech. With the same source voltage and multiple cores run at a lower frequency we see an almost tenfold increase in bandwidth while the total power consumption is reduced by a factor of four.

	Single core processor (45nm)	Multi-core processor (45nm)
Vdd	1.0V	1.0V
I/O pins(total)	1280 (ITRS)	3000 (Estimated)
Operating frequency	7.8GHz	4GHz
Chip-package data rate	7.8 Gb/s	4Gb/s
Bandwidth	125GByte/s	1 TeraByte/s
Power	429.78W	107.39W
Total number of pins on chip	3840	9000(Estimated)
Number of pins on the package	2480	4500(Estimated)

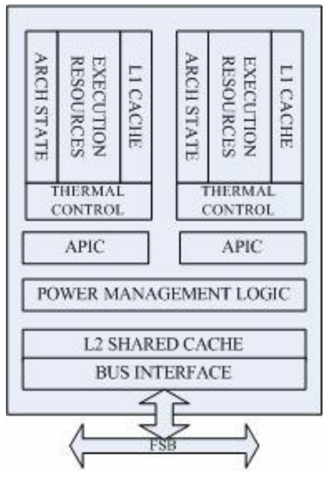
Table 1: Single Core vs. Multicore [16]

3. Multicore Implementations

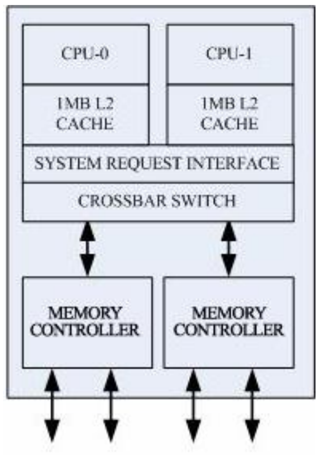
As with any technology, multicore architectures from different manufacturers vary greatly. Along with differences in communication and memory configuration another variance comes in the form of how many cores the microprocessor has. And in some multicore architectures different cores have different functions, hence they are heterogeneous. Differences in architectures are discussed below for Intel's Core 2 Duo, Advanced Micro Devices' Athlon 64 X2, Sony-Toshiba-IBM's CELL Processor, and finally Tilera's TILE64.

3.1 Intel and AMD Dual-Core Processors

Intel and AMD are the mainstream manufacturers of microprocessors. Intel produces many different flavors of multicore processors: the Pentium D is used in desktops, 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.



(a)



(b)

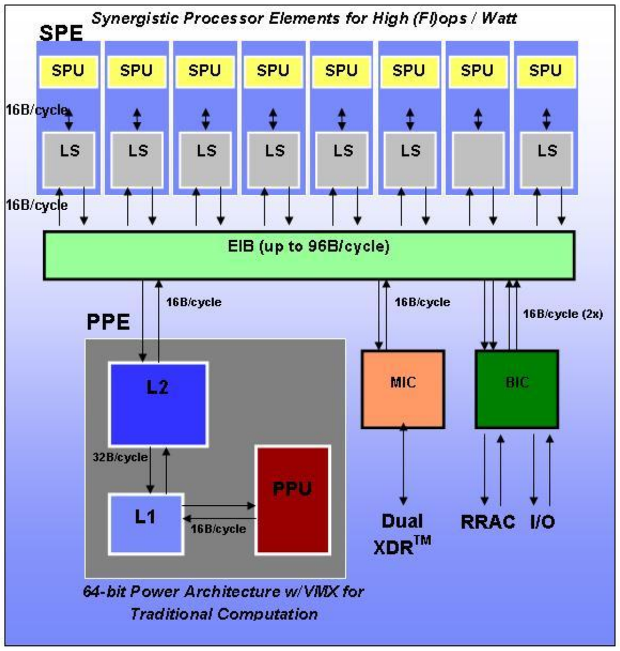
Figure 4: (a) Intel Core 2 Duo, (b) AMD Athlon 64 X2 [5]

Figure 4 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." [25] If a L1 cache miss occurs both the L2 cache and the second core's 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.

3.2 CELL Processor

A Sony-Toshiba-IBM partnership (STI) built the CELL processor for use in Sony's PlayStation 3, therefore, CELL is highly customized for gaming/graphics rendering which means superior processing power for gaming applications. The CELL is a heterogeneous multicore processor consisting of nine cores, one Power Processing Element (PPE) and eight Synergistic Processing Elements (SPEs), as can be seen in Figure 5. With CELL's real-time broadband architecture, 128 concurrent transactions to memory per processor are possible.



Error! Reference source not found.

Figure 5: CELL Processor [6]

The PPE is an extension of the 64-bit PowerPC architecture and manages the operating system and control functions. Each SPE has simplified instruction sets which use 128-bit SIMD instructions and have 256KB of local storage. Direct Memory Access is used to transfer data between local storage and main memory which allows for the high number of concurrent memory transactions. The PPE and SPEs are connected via the Element Interconnect Bus providing internal communication.

Other interesting features of the CELL are the Power Management Unit and Thermal Management Unit. Power and temperature are fundamental concerns in microprocessor design. The PMU allows for power reduction in the form of slowing, pausing, or completely stopping a unit. The TMU consists of one linear sensor and ten digital thermal sensors used to monitor temperature throughout the chip and provide an early warning if temperatures are rising in a certain area of the chip. The ability to measure and account for power and temperature changes has a great advantage in that the processor should never overheat or draw too much power.

3.3 Tilera TILE64

Tilera has developed a multicore chip with 64 homogeneous cores set up in a grid, shown in Figure 6. An application that is written to take advantage of these additional cores will run far faster than if it were run on a single core. Imagine having a project to finish, but instead of having to work on it alone you have 64 people to work for you. Each processor has its own L1 and L2 cache for a total of 5MB on-chip and a switch that connects the core into the mesh network rather than a bus or interconnect. The TILE64 also includes on-chip memory and I/O controllers. Like the CELL processor, unused tiles (cores) can be put into a sleep mode to further decrease power consumption. The TILE64 uses a 3-way VLIW (very long instruction word) pipeline to deliver 12 times the instructions as a single-issue, single-core processor. When VLIW is combined with the MIMD (multiple instruction, multiple data) processors, multiple operating systems can be run simultaneously and advanced multimedia applications such as video conferencing and video-on-demand can be run efficiently. [26]

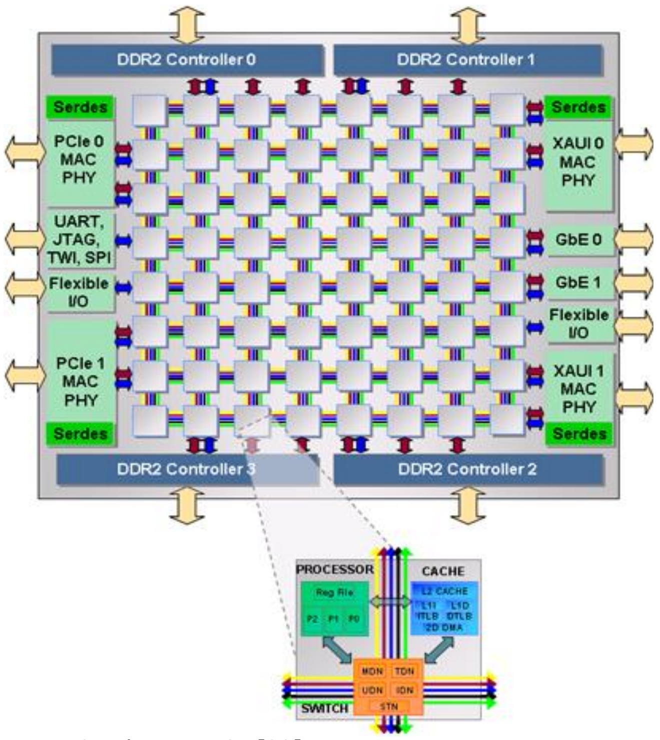


Figure 6: Tilera TILE64 [28]

4. Multicore 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 multicore processor to its full potential is another issue. If programmers don’t 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.

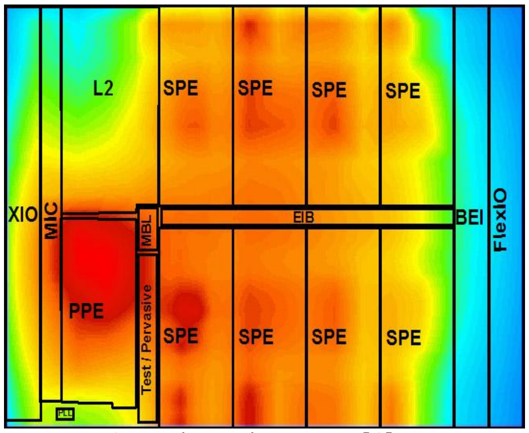


Figure 7: CELL Thermal Diagram [6]

To lessen the heat generated by multiple cores on a single chip, the chip is architected so that the number of hot spots doesn't grow too large and the heat is spread out across the chip. As seen in Figure 7, 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.

4.2 Cache Coherence

Cache coherence is a concern in a multicore 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 won't have the updated copy unless its cache entry is invalidated and a cache miss occurs. This cache miss forces the second core's cache entry to be updated. If this coherence policy wasn't in place garbage data would be read and invalid results would be produced, possibly crashing the program or the entire computer.

In general there are two schemes for cache coherence, a snooping protocol and a directory-based protocol. The snooping protocol only works with a bus-based system, and uses a number of states to determine whether or not it needs to update cache entries and if it has control over writing to the block. The directory-based protocol can be used on an arbitrary network and is, therefore, scalable to many processors or cores, in contrast to snooping which isn't scalable. In this scheme a directory is used that holds information about which memory locations are being shared in multiple caches and which are used exclusively by one core's cache. The directory knows when a block needs to be updated or invalidated. [23]

Intel's Core 2 Duo tries to speed up cache coherence by being able to query the second core's L1 cache and the shared L2 cache simultaneously. Having a shared L2 cache also has an added benefit since a coherence protocol doesn't need to be set for this level. AMD's Athlon 64 X2, however, has to monitor cache coherence in both L1 and L2 caches. This is sped up using the HyperTransport connection, but still has more overhead than Intel's model.

4.3 Multithreading

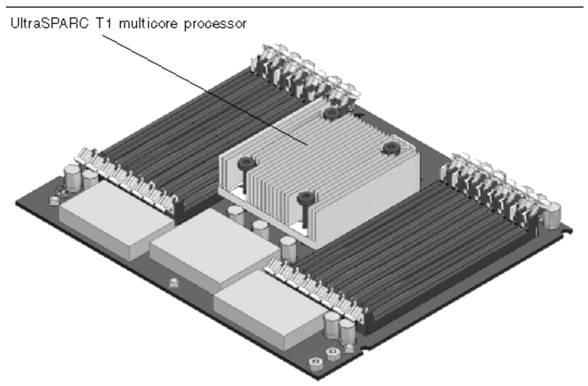
The last, and most important, issue is using multithreading or other parallel processing techniques to get the most performance out of the multicore processor. "With the possible exception of Java, there are no widely used commercial development languages with [multithreaded] extensions." [12] Rebuilding 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 multicore system. Some companies have heard the call and designed new products with multicore capabilities; Microsoft and Apple's newest operating systems can run on up to 4 cores, for example. [12, 11]

5. Open Issues

5.1 Improved Memory System

With numerous cores on a single chip there is an enormous need for increased memory. 32-bit processors, such as the Pentium 4, can address up to 4GB of main memory. With cores now using 64-bit addresses the amount of addressable memory is almost infinite. An improved memory system is a necessity; more main memory and larger caches are needed for multithreaded multiprocessors.

5.2 System Bus and Interconnection Networks



Sun Microsystems
http://docs.sun.com/source/819-7989-10/figures/2_Overview-5.gif

Extra memory will be useless if the amount of time required for memory requests doesn't improve as well. Redesigning the interconnection network between cores is a major focus of chip manufacturers. A faster network means a lower latency in inter-core communication and memory transactions. Intel is developing their Quickpath interconnect, which is a 20-bit wide bus running between 4.8 and 6.4 GHz; AMD's new HyperTransport 3.0 is a 32-bit wide bus and runs at 5.2 GHz [14]. A different kind of interconnect is seen in the TILE64's iMesh, which consists of five networks used to fulfill I/O and off-chip memory communication. Using five mesh networks gives the Tile architecture a per tile (or core) bandwidth of up to 1.28 Tbps (terabits per second). [27]

The question remains though, which type of interconnect is best suited for multicore processors? Is a bus-based approach better than an interconnection network? Or is there a hybrid like the mesh network that would work best?

5.3 Parallel Programming

> *To use multicore, you really have to use multiple threads. If you know how to do it, it's not bad. But the first time you do it there are lots of ways to shoot yourself in the foot. The bugs you introduce with multithreading are so much harder to find. [36]*

In May 2007, Intel fellow Shekhar Borkar stated that "The software has to also start following Moore's Law, software has to double the amount of parallelism that it can support every two years." [35] Since the number of cores in a processor is set to double every 18 months, it only makes sense that the software running on these cores takes this into account. Ultimately, programmers need to learn how to write parallel programs that can be split up and run concurrently on multiple cores instead of trying to exploit single-core hardware to increase parallelism of sequential programs.

Developing software for multicore processors brings up some latent concerns. How does a programmer ensure that a high-priority task gets priority across the processor, not just a core? In theory even if a thread had the highest priority within the core on which it is running it might not have a high priority in the system as a whole. Another necessary tool for developers is debugging. However, how do we guarantee that the entire system stops and not just the core on which an application is running?

These issues need to be addressed along with teaching good parallel programming practices for developers. Once programmers have a basic grasp on how to multithread and program in parallel, instead of sequentially, ramping up to follow Moore's law will be easier.

5.4 Starvation

If a program isn't developed correctly for use in a multicore processor one or more of the cores may starve for data. This would be seen if a single-threaded application is run in a multicore 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.

With a shared cache, for example Intel Core 2 Duo's shared L2 cache, if a proper replacement policy isn't in place one core may starve for cache usage and continually make costly calls out to main memory. The replacement policy should include stipulations for evicting cache entries that other cores have recently loaded. This becomes more difficult with an increased number of cores effectively reducing the amount of evictable cache space without increasing cache misses.

5.5 Homogeneous vs. Heterogeneous Cores

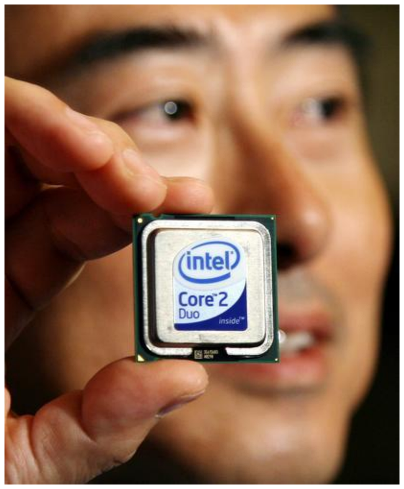
Architects have debated whether the cores in a multicore environment should be homogeneous or heterogeneous, and there is no definitive answer…yet. Homogenous cores are all exactly the same: equivalent frequencies, cache sizes, functions, etc. However, each core in a heterogeneous system may have a different function, frequency, memory model, etc. There is an apparent trade-off between processor complexity and customization. All of the designs discussed above have used homogeneous cores except for the CELL processor, which has one Power Processing Element and eight Synergistic Processing Elements.

Homogeneous cores are easier to produce since the same instruction set is used across all cores and each core contains the same hardware. But are they the most efficient use of multicore technology?

Each core in a heterogeneous environment could have a specific function and run its own specialized instruction set. Building on the CELL example, a heterogeneous model could have a large centralized core built for generic processing and running an OS, a core for graphics, a communications core, an enhanced mathematics core, an audio core, a cryptographic core, and the list goes on. [33] This model is more complex, but may have efficiency, power, and thermal benefits that outweigh its complexity. With major manufacturers on both sides of this issue, this debate will stretch on for years to come; it will be interesting to see which side comes out on top.

6. Conclusion

Before multicore 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.



Intel's Japanese subsidiary President Kazumasa Yoshida unveils the new processor "Core 2 Duo"
Getty Images, Agence France Presse 07-27-2006

Multicore 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 multicore 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. The memory systems and interconnection networks also need improvement. And finally, it is still unclear whether homogeneous or heterogeneous cores are more efficient.

With so many different designs (and potential for even more) it is nearly impossible to set any standard for cache coherence, interconnections, and layout. The greatest difficulty remains in teaching parallel programming techniques (since most programmers are so versed in sequential programming) and in redesigning current applications to run optimally on a multicore system.

Multicore processors are an important innovation in the microprocessor timeline. With skilled programmers capable of writing parallelized applications multicore efficiency could be increased dramatically. In years to come we will see much in the way of improvements to these systems. These improvements will provide faster programs and a better computing experience.

References

[1] W. Knight, “Two Heads Are Better Than One”, IEEE Review, September 2005
[2] R. Merritt, “CPU Designers Debate Multi-core Future”, EETimes Online, February 2008, http://www.eetimes.com/showArticle.jhtml?articleID=206105179
[3] P. Frost Gorder, “Multicore Processors for Science and Engineering”, IEEE CS, March/April 2007
[4] D. Geer, “Chip Makers Turn to Multicore Processors”, Computer, IEEE Computer Society, May 2005
[5] L. Peng et al, “Memory Performance and Scalability of Intel’s and AMD’s Dual-Core Processors: A Case Study”, IEEE, 2007
[6] D. Pham et al, “The Design and Implementation of a First-Generation CELL Processor”, ISSCC
[7] P. Hofstee and M. Day, “Hardware and Software Architecture for the CELL Processor”, CODES+ISSS ’05, September 2005
[8] J. Kahle, “The Cell Processor Architecture”, MICRO-38 Keynote, 2005
[9] D. Stasiak et al, “Cell Processor Low-Power Design Methodology”, IEEE MICRO, 2005
[10] D. Pham et al, “Overview of the Architecture, Circuit Design, and Physical Implementation of a First-Generation Cell Processor”, IEEE Journal of Solid-State Circuits, Vol. 41, No. 1, January 2006
[11] D. Geer, “For Programmers, Multicore Chips Mean Multiple Challenges”, Computer, September 2007
[12] M. Creeger, “Multicore CPUs for the Masses”, QUEUE, September 2005
[13] R. Merritt, “Multicore Puts Screws to Parallel-Programming Models”, EETimes Online, February 2008, http://www.eetimes.com/news/latest/showArticle.jtml?articleID=206504466
[14] R. Merritt, “X86 Cuts to the Cores”, EETimes Online, September 2007, http://www.eetimes.com/showArticle.jtml?articleID=202100022
[15] R. Merritt, “Multicore Goals Mesh at Hot Chips”, EETimes Online, August 2007, http://www.eetimes.com/showArticle.jtml?articleID=201800925
[16] P. Muthana et al, “Packaging of Multi-Core Microprocessors: Tradeoffs and Potential Solutions”, 2005 Electronic Components and Technology Conference, 2005

[17] S. Balakrishnan et al, "The Impact of Performance Asymmetry in Emergng Multicore Architectures", Proceedings of the 32nd International Symposium on Computer Architecture, 2005

[18] "A Brief History of Microprocessors", Microelectronics Industrial Centre, Northumbria University, 2002, http://mic.unn.ac.uk/miclearning/modules/micros/ch1/micro01hist.html

[19] B. Brey, "The Intel Microprocessors", Sixth Edition, Prentice Hall, 2003

[20] Video Transcript, "Excerpts from a Conversation with Gordon Moore: Moore's Law", Intel Corporation, 2005

[21] Wikipedia, "Moore's Law", http://upload.wikimedia.org/wikipedia/commons/0/06/Moore_Law_diagram_(2004).png

[22] Intel, "World's First 2-Billion Transistor Microprocessor", http://www.intel.com/technology/architecture-silicon/2billion.htm?id=tech_mooreslaw+rhc_2b

[23] M. Franklin, "Notes from ENEE759M: Microarchitecture", Spring 2008

[24] U. Nawathe et al, "An 8-core, 64-thread, 64-bit, power efficient SPARC SoC (Niagara 2)", ISSCC, http://www.opensparc.net/pubs/preszo/07/n2isscc.pdf

[25] J. Dowdeck, "Inside Intel Core Microarchitecture and Smart Memory Access", Intel, 2006, http://download.intel.com/technology/architecture/sma.pdf

[26] Tilera, "Tile 64 Product Brief", Tilera, 2008, http://www.tilera.com/pdf/ProductBrief_Tile64_Web_v3.pdf

[27] D. Wentzlaff et al, "On-Chip Interconnection Architecture of the Tile Processor", IEEE Micro, 2007

[28] Tilera, "TILE64 Processor Family", http://www.tilera.com/products/processors.php

[29] D. Olson, "Intel Announces Plan for up to 8-core Processor", Slippery Brick, March 2008, http://www.slipperybrick.com/2008/03/intel-dunnington-nehalem-processor-chips/

[30] K. Shi and D. Howard, "Sleep Transistor Design and Implementation – Simple Concepts Yet Challenges To Be Optimum", Synopsys, http://www.synopsys.com/sps/pdf/optimum_sleep_transistor_vlsi_dat06.pdf

[31] W. Huang et al, "An Improved Block-Based Thermal Model in HotSpot 4.0 with Granularity
Considerations", University of Virginia, April 2007

[32] S. Mukherjee and M. Hill, "Using Prediction to Accelerate Coherence Protocols", Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA), 1998

[33] R. Alderman, "Multicore Disparities", VME Now, December 2007, http://vmenow.com/c/index.php?option=com_content&task=view&id=105&Itemid=46

[34] R. Kumar et al, "Single-ISA Heterogeneous Multi-core Architectures with Multithreaded Workload Performance", Proceedings of the 31st Annual International Symposium on Computer Architecture, June 2004

[35] T. Holwerda, "Intel: Software Needs to Heed Moore's Law", http://www.osnews.com/story/17983/Intel-Software-Needs-to-Heed-Moores-Law/

[36] R. Goering, "Panel Confronts Multicore Pros and Cons", http://www.eetimes.com/news/design/showArticle.jhtml?articleID=183702416