



# E - NEWS LETTER



Tuticorin Branch of Southern India Regional Council of The Institute of Chartered Accountants of India  
(Set up by an Act of Parliament)

*April 2010*

## Chairman Writes



Dear Colleagues,

Greetings,

After the heated discussions and various post-budget analyses, we are entering into the bank audits' season with a hectic month ahead. With a lot of work to be done and acute deadlines given by banks and with no compromises on the integrity of our profession, bank audits require optimal planning and execution.

Recent investigations have brought to light, an astounding multi-crore nexus between authorized contractors dealing in medicine, local brokers and high profile pharmaceutical owners. It is just the tip of the iceberg. There could be crores and crores worth of medicines circulating in almost any hospital and it is almost next to impossible to trace them. Innocent lives become insignificant before money. Ultimately money becomes the basis for all crimes.

Striking a blow against corruption, our Prime Minister Dr.Manmohan Singh has done away with the requirement to disclose one's name and address while giving a complaint against a corrupt official. A properly framed complaint addressed to the cabinet secretary either anonymously or using a pseudonym is enough. Earlier such complaints were not entertained. This change has been made keeping in mind, the fear people had earlier while making complaints. This is a very welcome move.

Every service oriented field has become corrupt. Our Chartered Accountancy profession being branded as 'Partners in Nation building', we are called to be socially responsible. Propagating alone will not help, we should act accordingly. Besides constantly updating ourselves in our profession and keeping up with demanding schedules, there is always room for social contribution. Instead of waiting for an opportunity, we can and we should find one.

As Emerson says,

"Do not follow where the path may lead. Go instead where there is no path and leave a trail."

Yours ever loving,

**CA.H.Raman**

# Core Technology

A **multi-core processor** is a processing system composed of two or more independent cores. It can be described as an integrated circuit to which two or more individual processors (called *cores* in this sense) have been attached. The cores are typically integrated onto a single integrated circuit die (known as a chip multiprocessor or CMP), or they may be integrated onto multiple dies in a single chip package. A **many-core** processor is one in which the number of cores is large enough that traditional multiprocessor techniques are no longer efficient — this threshold is somewhere in the range of several tens of cores — and probably requires a network on chip.

A **dual-core processor** contains two cores, and a **quad-core processor** contains four cores. A multi-core processor implements multiprocessing in a single physical package. Cores in a multi-core device may be coupled together 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, 2-dimensional mesh, and crossbar. All cores are identical in *homogeneous* multi-core systems and they are not identical in heterogeneous multi-core systems. Just as with single-processor systems, cores in multi-core systems may implement architectures like 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 amount of performance gained by the use of a multi-core processor is strongly dependent on the software algorithms and implementation. In particular, the possible gains are limited by the fraction of the software that can be parallelized to run on multiple cores simultaneously; 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. Many typical applications, however, do not realize such large speedup factors, and thus the parallelization of software is a significant on-going topic of research.

## Terminology

There is some discrepancy in the semantics by which the terms *multi-core* and *dual-core* are defined. Most commonly they are used to refer to some sort of central processing unit (CPU), but are sometimes also applied to digital signal processors (DSP) and system-on-a-chip (SoC). Additionally, some use these terms to refer only to multi-core microprocessors that are manufactured on the *same* integrated circuit die. These people generally refer to separate microprocessor dies in the same package by another name, such as *multi-chip module*. This article uses both the terms "multi-core" and "dual-core" to reference microelectronic CPUs manufactured on the *same* integrated circuit, unless otherwise noted.

In contrast to multi-core systems, the term *multi-CPU* refers to multiple physically separate processing units (which often contain special circuitry to facilitate communication between each other).

The terms many-core and *massively multi-core* are sometimes used to describe multi-core architectures with an especially high number of cores (tens or hundreds).

Some systems use many soft microprocessor cores placed on a single FPGA. Each of "cores" can be considered a "semiconductor intellectual property core" as well as a CPU core.

## Development

While manufacturing technology continues to improve, reducing the size of single gates, physical limits of semiconductor-based microelectronics have become a major design concern. Some effects of these physical limitations can cause significant **heat dissipation** and **data synchronization** problems. The demand for more capable microprocessors causes CPU designers to use various methods of increasing performance. Some *instruction-level parallelism* (ILP) methods like superscalar pipelining are suitable for many applications, but are inefficient for others that tend to contain difficult-to-predict code. Many applications are better suited to *thread level parallelism* (TLP) methods, and multiple independent CPUs is one common method used to increase a system's overall TLP. A combination of increased available space due to refined manufacturing processes and the demand for increased TLP is the logic behind the creation of multi-core CPUs.

### Commercial incentives

Several business motives drive the development of dual-core architectures. Since symmetric multiprocessing (SMP) designs have long been implemented using discrete CPUs, the issues regarding implementing the architecture and supporting it in software are well known.

Additionally,

- Utilizing a proven processing core design without architectural changes reduces design risk significantly.
- For general-purpose processors, much of the motivation for multi-core processors comes from greatly diminished gains in processor performance from increasing the operating frequency. This is due to three primary factors:
  1. The *memory wall*; the increasing gap between processor and memory speeds. this effect pushes cache sizes larger in order to mask the latency of memory. This helps only to the extent that memory bandwidth is not the bottleneck in performance.
  2. The *ILP wall*; the increasing difficulty of finding enough parallelism in a single instructions stream to keep a high performance single-core processor busy.
  3. The *power wall*; the trend of consuming exponentially increasing power with each factorial increase of operating frequency. This increase can be mitigated by "shrinking" the processor by using smaller traces for the same logic. The *power wall* poses manufacturing, system design and deployment problems that have not been justified in the face of the diminished gains in performance due to the *memory wall* and *ILP wall*.
- The terminology "dual-core" (and other multiples) lends itself to marketing efforts.

In order to continue delivering regular performance improvements for general-purpose processors, manufacturers such as Intel and AMD have turned to multi-core designs, sacrificing lower manufacturing costs for higher performance in some applications and systems. Multi-core architectures are being developed, but so are the alternatives. An especially strong contender for established markets is the further integration of peripheral functions into the chip.

## 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.

The largest boost in performance will likely be noticed in improved response time while running CPU-intensive processes, like antivirus scans, ripping/burning media (requiring file conversion), or searching for folders. For example, if the automatic virus scan initiates while a movie is being watched, the application running the movie is far less likely to be starved of processor power, as the antivirus program will be assigned to a different processor core than the one running the movie playback.

Assuming that the die can fit into the package, physically, the multi-core CPU designs require much less Printed Circuit Board (PCB) space than 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.

## Disadvantages

In addition to operating system (OS) support, adjustments to existing software are required to maximize utilization of the computing resources provided by multi-core processors. Also, the ability of multi-core processors to increase application performance depends on the use of multiple threads within applications. The situation is improving: for example the Valve Corporation's Source engine, offers multi-core support and Crytek has developed similar technologies for CryEngine 2, which powers their game, *Crysis*. Emergent Game Technologies' Gamebryo engine includes their Floodgate technology which simplifies multicore development across game platforms.

Integration of a multi-core chip drives production yields down and they are more difficult to manage thermally than lower-density single-chip designs. Intel has partially countered this first problem by creating its quad-core designs by combining two dual-core on a single die with a unified cache, hence any two working dual-core dies can be used, as opposed to producing four cores on a single die and requiring all four to work to produce a quad-core. From an architectural point of view, ultimately, single CPU designs may make better use of the silicon surface area than multiprocessing cores, so a development commitment to this architecture may carry the risk of obsolescence. Finally, raw processing power is not the only constraint on system performance. Two processing cores sharing the same system bus and memory bandwidth limits the real-world performance advantage. If a single core is close to being memory bandwidth limited, going to dual-core might only give 30% to 70% improvement. If memory bandwidth is not a problem, a 90% improvement can be expected. It would be possible for an application that used two

CPUs to end up running faster on one dual-core if communication between the CPUs was the limiting factor, which would count as more than 100% improvement.

## Hardware

### Trends

The general trend in processor development has been from multi-core to many-core: from dual-, tri-, quad-, hexa-, octo-core chips to ones with tens or even hundreds of cores. In addition, multi-core chips mixed with simultaneous multithreading, memory-on-chip, and special-purpose "heterogeneous" cores promise further performance and efficiency gains, especially in processing multimedia, recognition and networking applications. There is also a trend of improving energy efficiency by focusing on performance-per-watt with advanced fine-grain or ultra fine-grain power management and dynamic voltage and frequency scaling (i.e. laptop computers and portable media players).

### Architecture

One of the biggest areas for variety in multi-core architecture is the composition and balance of the cores themselves. Some architectures use one core design which is repeated consistently ("homogeneous"), while others use a mixture of different cores, each optimized for a different, "heterogeneous", role .

As an example of this discussion, the article *CPU designers debate multi-core future* by Rick Merritt, EE Times 2008, includes comments:

*"Chuck Moore [...] suggested computers should be more like cellphones, using a variety of specialty cores to run modular software scheduled by a high-level applications programming interface.*

*[...] Atsushi Hasegawa, a senior chief engineer at Renesas, generally agreed. He suggested the cellphone's use of many specialty cores working in concert is a good model for future multi-core designs.*

*[...] Anant Agarwal, founder and chief executive of startup Tilera, took the opposing view. He said multi-core chips need to be homogeneous collections of general-purpose cores to keep the software model simple. "*

## Software impact

An anti-virus application may create a new thread for a scan process, while its GUI thread waits for commands from the user (e.g. cancel the scan). In such cases, a multicore architecture is of little benefit for the application itself due to the single thread doing all heavy lifting and the inability to balance the work evenly across multiple cores. Programming truly multithreaded code often requires complex co-ordination of threads and can easily introduce subtle and difficult-to-find bugs due to the interleaving of processing on data shared between threads (thread-safety). Consequently, such code is much more difficult to debug than single-threaded code when it breaks. There has been a perceived lack of motivation for writing consumer-level threaded applications because of the relative rarity of consumer-level multiprocessor hardware. Although threaded applications incur little additional performance penalty on single-processor machines, the extra overhead of development has been difficult to justify due to the preponderance of single-processor machines.

Given the increasing emphasis on multicore chip design, stemming from the grave thermal and power consumption problems posed by any further significant increase in processor clock speeds, the extent to which software can be multithreaded to take advantage of these new chips is likely to be the single greatest constraint on computer performance in the future. If developers are unable to design software to fully exploit the resources provided by multiple cores, then they will ultimately reach an insurmountable performance ceiling.

The telecommunications market had been one of the first that needed a new design of parallel datapath packet processing because there was a very quick adoption of these multiple core processors for the datapath and the control plane. These MPUs are going to replace the traditional Network Processors that were based on proprietary micro- or pico-code.

Parallel programming techniques can benefit from multiple cores directly. Some existing parallel programming models such as Cilk++, OpenMP, Skandium, MPI can be used on multi-core platforms. Intel introduced a new abstraction for C++ parallelism called TBB. Other research efforts include the Codeplay Sieve System, Cray's Chapel, Sun's Fortress, and IBM's X10.

Multi-core processing has also affected the ability of modern day computational software development. Developers programming in newer languages might find that their modern languages do not support multi-core functionality. This then requires the use of numerical libraries to access code written in languages like C and Fortran, which perform math computations faster than newer languages like C#. Intel's MKL and AMD's ACML are written in these native languages and take advantage of multi-core processing.

Managing concurrency acquires a central role in developing parallel applications. The basic steps in designing parallel applications are:

#### Partitioning

The partitioning stage of a design is intended to expose opportunities for parallel execution. Hence, the focus is on defining a large number of small tasks in order to yield what is termed a fine-grained decomposition of a problem.

#### Communication

The tasks generated by a partition are intended to execute concurrently but cannot, in general, execute independently. The computation to be performed in one task will typically require data associated with another task. Data must then be transferred between tasks so as to allow computation to proceed. This information flow is specified in the communication phase of a design.

#### Agglomeration

In the third stage, development moves from the abstract toward the concrete. Developers revisit decisions made in the partitioning and communication phases with a view to obtaining an algorithm that will execute efficiently on some class of parallel computer. In particular, developers consider whether it is useful to combine, or agglomerate, tasks identified by the partitioning phase, so as to provide a smaller number of tasks, each of greater size. They also determine whether it is worthwhile to replicate data and/or computation.

#### Mapping

In the fourth and final stage of the parallel algorithm design process, the developers specify where each task is to execute. This mapping problem does not arise on uniprocessors or on shared-memory computers that provide automatic task scheduling.

On the other hand, on the server side, multicore processors are ideal because they allow many users to connect to a site simultaneously and have independent threads of execution. This allows for Web servers and application servers that have much better throughput.

## **Licensing**

Typically, proprietary enterprise server software is licensed "per processor". In the past a CPU was a processor and most computers had only one CPU, so there was no ambiguity.

Now there is the possibility of counting cores as processors and charging a customer for multiple licenses for a multi-core CPU. However, the trend seems to be counting dual-core chips as processor as Microsoft, Intel, and AMD support this view. Microsoft have said they would treat a socket as a single processor.

Oracle counts an AMD X2 or Intel dual-core CPU as a single processor but has other numbers for other types, especially for processors with more than two cores. IBM and HP count a multi-chip module as multiple processors. If multi-chip modules count as one processor, CPU makers have an incentive to make large expensive multi-chip modules so their customers save on software licensing. It seems that the industry is slowly heading towards counting each die (see Integrated circuit) as a processor, no matter how many cores each die has.

## **Embedded applications**

An area of processor technology distinct from "mainstream" PCs is that of embedded computing. The same technological drivers towards multicore apply here too. Indeed, in many cases the application is a "natural" fit for multicore technologies, if the task can easily be partitioned between the different processors.

In addition, embedded software is typically developed for a specific hardware release, making issues of software portability, legacy code or supporting independent developers less critical than is the case for PC or enterprise computing. As a result, it is easier for developers to adopt new technologies and as a result there is a greater variety of multicore processing architectures and suppliers.

In network processing, it is now mainstream for devices to be multi-core, with companies such as Freescale Semiconductor, Cavium Networks, Wintegra and Broadcom all manufacturing products with eight processors.

In digital signal processing the same trend applies: Texas Instruments has the three-core TMS320C6488 and four-core TMS320C5441, Freescale the four-core MSC8144 and six-core MSC8156 (and both have stated they are working on eight-core successors). Newer entries include the Storm-1 family from Stream Processors, Inc with 40 and 80 general purpose ALUs per chip, all programmable in C as a SIMD engine and Picochip with three-hundred processors on a single die, focused on communication applications.