Microprocessors Face a Multicore Future

by **Alan S. Brown**

*h*ow will engineers master the billions of transistors on tomorrow's microprocessors? It's going to take a combination of divide-and-conquer and urban renewal to put them all to work.

In 1965, Gordon Moore, who founded Intel Corp. three years later, noted that the number of transistors on commercial microprocessors had doubled every year for the past seven years. He estimated that the trend might continue for another ten years and that by 1975 a single processor might contain 65,000 transistors.

By 1975, Intel was producing chips with 6,500 transistors. Three years later, Intel's 8088 chip (which powered the original IBM PC) was up to 29,000 transistors. That year, Moore revised his original prediction. He now forecast that transistors would double every two years, and processor performance every 18 months.

So much for the history lesson. Fast-forward 35 years, and processors now have hundreds of millions of transistors. Intel's Core i7 processors, introduced in 2008, sports 731 million transistors. Doing the math that is now enshrined as Moore's Law, we find that commercial processors are likely to contain nearly 12 billion transistors within five years.

That is a lot of real estate. To appreciate what it means, consider how a city like New York, with 3.2 million housing units, functions. It is organized into five boroughs, each of which is divided into many neighborhoods. Some neighborhoods have specialized functions, such as residences, offices, stores, manufacturing, distribution, and recreation. Others mix everything together. A vast network of streets, highways, bus routes, and subways keeps people flowing from one destination to the next.

In the past, microprocessors have acted like New York's mixed-use neighborhoods. They did general-purpose calculations. If you needed, say, faster graphics for games or engineering simulations, you would buy a video card with a graphics processor optimized for the complex mathematical and geo-
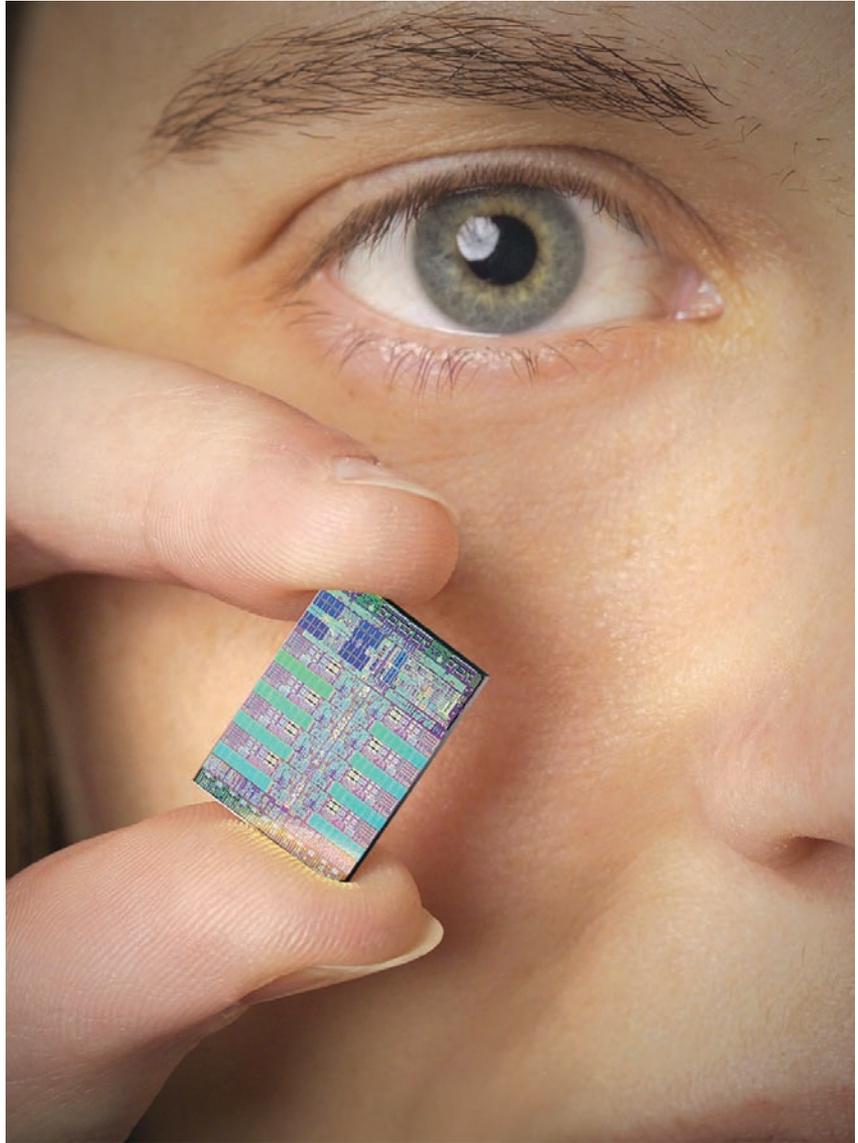
An IBM Cell combines processor, graphics processor, and math acclerators onto a single chip, held here by an engineer.

metrical calculations needed to render high-speed images.

That worked fine for many applications. Yet it involved long time lags (at least when measured against the gigahertz speed of modern chips) as data moved along a circuit board to an external processor. With so much real estate available on today's chips, hardware developers reasoned, why not use some of the processor's hundreds of millions of transistors to build a powerful graphic processor into the chip itself?
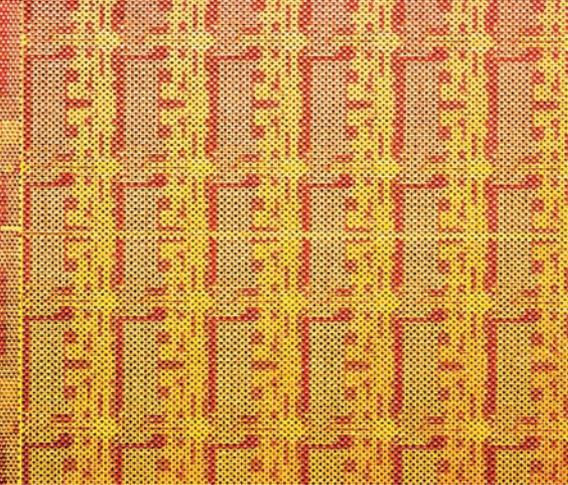
In fact, why not build in more high-speed memory, signal processing, accelerators for non-graphic processing, and chipsets to link the processor with outside devices? For processors used in mobile computing, why not include GPS systems, video processors, and sensors that enable us to use our phones as game consoles?

The Law

Those advantages are compelling, but not overriding. The real reason processor designers abandoned ever larger and more complex single processors had to do with the part of Moore's Law that involves doubling performance rather than transistors.

There are many ways to goose processor performance, but none is more important than clock speed. Every operation on a chip is synchronized with a clock. In 1975, 10 years after Moore first enunciated his law, Intel's 8080 processor had a clock frequency of 2 MHz and could perform 2 million operations per second. Today's chips run 1,000 times faster, in the 2-4 GHz range.

Faster clock speeds require higher voltages, and that generates more heat. This became painfully obvious in 2005, when Intel cancelled its single-core successor to the Pentium 4, codenamed Tejas, because it generated such high temperatures.

Instead, Intel released its first Core 2 dual-core processor. By parceling out the computing load between two cores, Intel could throttle back its clock speeds, reduce power and heat, and still improve processor performance.

In fact, hardware engineers had been working on multicore processors for many years. Yet there was no real consensus. Should they build neighborhoods where different groups of transistors have specialized functions? Or should they clone their processor into multiple mixed-use neighborhoods that can perform any task as needed? What kind of a transportation system would they need to keep data shuttling at high speeds between different cores without getting lost?

The semiconductor industry has found workable solutions to those questions as it went from two- to four- and now six- and eight-core processors.

Dividing processing-intensive tasks among several processors has yielded some impressive results. Scanning for viruses, compressing and uncompressing media streams, and juggling internet applications that would ordinarily take several dedicated servers all run significantly faster on multicore processors.

LabView software for building virtual instrumentation and controls has long embraced multicore processing. Its developer, National Instruments Coorporation of Austin, TX, routinely stuns audiences with examples of multicore PC-based instruments that run nearly as fast as the best dedicated hardware developed for the same purpose.

Such multicore designs look revolutionary, although they often build on established design principles. This will not be enough in the future when processors contain scores of cores.

Intel, for example, has built experimental chips with 48 and 100 cores. Tilera Corporation, a fabless semiconductor company founded in San Jose, CA, in 2004, launched a 64-core processor in 2007.

Such core proliferation promises complete computers and even server farms on a chip, as well as smart phones with desktop capabilities. Yet getting there will not prove easy.

Power to the Groups

Tilera plans to ship this new 100-core Gx chip in 2011. While each core itself is relatively simple, the chip can partition the cores into groups to apply computing power as needed. A two-dimensional interconnect mesh moves data quickly between cores and the processor's memory controllers and I/O's.
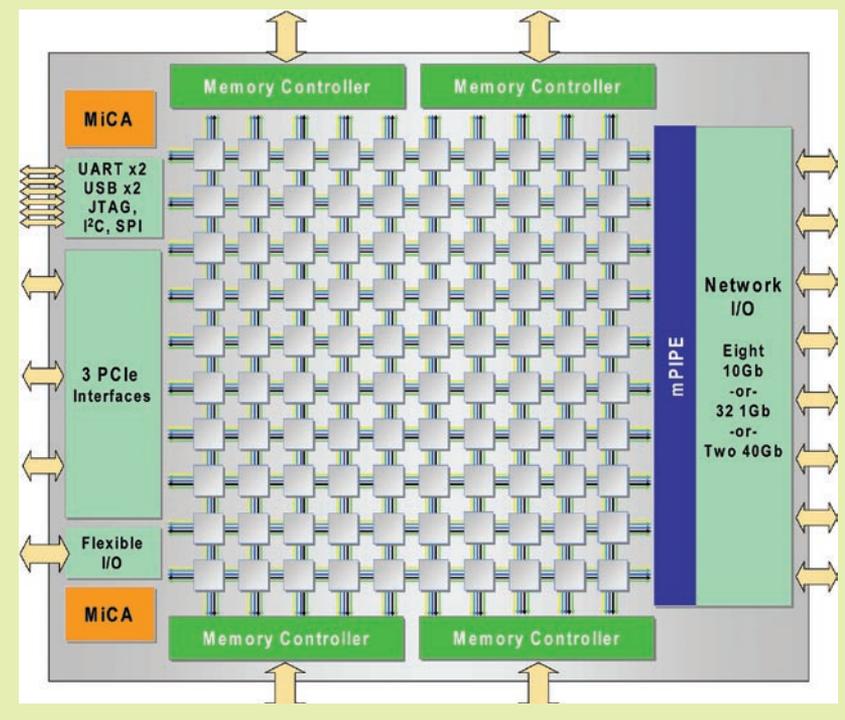
Tiles

Tilera is one company that has taken the multicore plunge. The firm was co-founded by Anant Agarwal, an MIT professor who has studied multicore processing since the 1980s. Tilera's philosophy is simple: Instead of mixing general-purpose and specialized cores, its commercial TilePro64 chip has 64 identical cores. It plans to ship a 100-core chip, the Tile-Gx™, later in 2011.

Tilera's cores are lightweight—that is, they lack specialized hardware to run such calculations as floating-point and vector operations efficiently. They also use reduced instruction set computing (RISC), opting for simplified instructions that run much faster (but often with more iterations) than more complex instruction sets.

On the other hand, the processor is managed by a proprietary compiler. It breaks software into instructions that run in parallel on the chip's cores. The compiler also monitors the cores and can partition them into workgroups in ways that best use the processor's capacity. Since the compiler handles scheduling, the cores do not need additional hardware to manage this task.

The key to Tilera's architecture is its mesh, the high-speed interconnections that tie the cores together. The company's marketing director, Bob Doud, likens the mesh to the grout that runs between tiles (which are the cores) on a bathroom floor. It forms a tiny network that enables the cores to talk with one another and share such on-chip resources as memory, Ethernet, PC Express, and other input/output.

To appreciate how it works, consider how other multicore chips communicate. The first dual-core chips used a bus, a data highway between cores and shared resources. It could send only one message at a time and needed circuitry to decide which data segment went next.

Simple buses become inefficient as the number of cores grow. Most four-core processors switched to ring interconnects. Rings pass data two ways, clockwise and counterclockwise, but still only one message at a time. They also need hardware to register the data at each step along its journey. Each *latch* along the ring accepts the input, locks it, then moves it to output and transfers it.

Each operation takes a tick of the clock. Doud likens rings to old-fashioned bucket brigades, passing full buckets in one direction and empties in the other. Yet the need to register each message as it moves from position to position poses a problem.

"You need a large pipeline because you're filling it with information you don't need," Doud notes. "It works for four cores and it's okay for six, but what happens when you have 16 cores communicating at a time? The more cores, the more power you need to run and to move the data faster so you're not starving the cores for information."

Both rings and buses are one-dimensional pathways; that is, data can move in only two directions, right or left. Tilera's mesh, on the other hand, is a two-dimensional grid where data can travel in two dimensions, north, south, east, and west.

"The messages are still moving one step along the way for every clock cycle, so in that way it's not any different from a bus or a ring," Doud explains. "But now imagine 100 cores lined up in a row with a bus or ring across them. It could take as many as 100 stops to go from end to end. But on a 100-core grid, they would move sideways 10 cores and down nine, so 19 stops is the farthest you would need to go. That's why nobody is going to build a bus or ring for 32 or 64 cores."

Future is in the Clouds

This futuristic Intel single-chip cloud computer has 48 Intel cores and runs at as low as 25 watts.

Tilera's new 100-core processor will actually have separate grids for memory, input/output, cache coherency, and two that programmers can define. This will give users up to 200 terabits of mesh bandwidth, and bandwidth has been a key barrier to scaling up multicore processors. "This is more than twice the bandwidth you would need. It's like a freeway system in a busy metropolitan area," Doud states. Equally important, it achieves these high transfer rates running at only 1.5 GHz, a clock speed most processors surpassed years ago.

Cache coherency is another critical Tilera technology. It has to do with memory, which is always an issue. Today's computers have gigabytes of system memory on their motherboards and megabytes of cache memory in their processors. Single core chips check data out of memory, move it into the cache where they can access it very quickly, and then write it back to system memory.

Two cores can negotiate who gets what on a first-come, first-served basis. It gets more complicated for four cores and more. The result is a slow and complicated scheme that involves checking data in and out of centralized caches. Tilera opts for distributing and managing memory among the tiles and enabling the tiles to look inside each other's caches.

Tilera's processors cannot do everything. They will not replace graphics chips or run high-speed, vector-based database searches. Nor are they backward-compatible with software built for Intel chips. On the other hand, putting 100 cores on a single chip and dividing them based upon workload seems a good fit for the Internet and the cloud (which provides the heavy-duty processing behind smartphone applications).

The company claims to have 100 customers, and it recently teamed with Optera Solutions to introduce a cloud server that packs 512 cores (eight TilePro64 processors) into two rack units. Each of those processors consumes only 23 watts. As a result, the server needs just a fraction of the power and cooling required for competitive units.

Mixed Use

Tilera's homogenous processors, which combine many of the same exact cores, are one way to go. Another is to mix components. If there is a poster child for that type of heterogeneous process, it could be the Sony-Toshiba-IBM Cell (for cell broadband engine architecture) chip that powers Sony's PlayStation 3 game console. Starting in 2001, the three companies spent four years and reportedly $400 million to develop the processor.
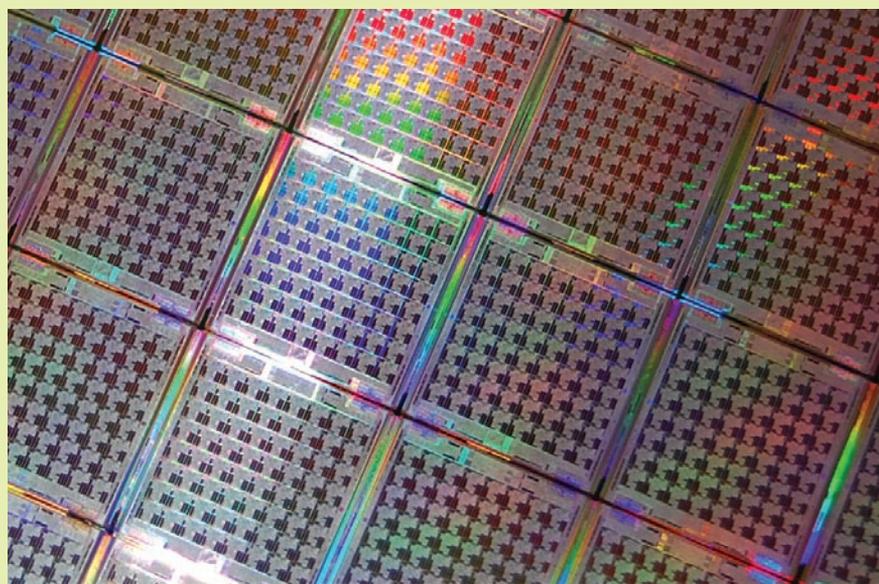
Not surprisingly, the engineers optimized the chip for graphics performance. On one hand, the chip had one mid-range, general-purpose processor base that used IBM's RISC technology. But it also included eight separate co-processing cores designed for specialized graphics tasks. A ring data bus linked together all nine processors. The chip also monitors its own temperature and can turn off unused cores to cool itself.

The eight co-processors are designed for heavy-duty mathematics, like Fourier analysis of data and floating-point vectorized

Chips on a silicon wafer developed and manufactured by Tilera.

processing of arrays of data. These specialized units, often called accelerators, run calculations hundreds of times faster than general-purpose cores because they contain everything needed to process information. Instead of reading and writing processing instructions from the hard drive and memory, the processors pull data, such as vectors, from memory and operate on them directly. This eliminates hundreds of redundant memory calls and thousands of valuable clock ticks.

The experimental processors introduced by Intel, the world's largest processor manufacturer, are not heterogeneous like IBM's Cell but offer more flexibility than Tilera's TilePro64. Intel introduced a simple 80-core chip in 2007 and a more serious 48-core chip, which it calls a single-chip cloud computer, at the end of 2009.

The 48-core processor consists of 27 dual-core tiles on a network mesh (which Intel calls a fabric). Each tile, which contains two separate processors, connects to the network with a router. The entire chip contains about 1.3 billion transistors, or roughly 27 million chips per core (including networking devices), yet fully supports Intel's traditional processor operations.

Like Tilera, Intel has rethought how to take advantage of all that real estate on today's processors. That means breaking down the historic barrier between hardware used to process data (processors) and hardware used to control memory, peripherals, and I/O (chipsets). More of those functions are moving onto its chips.

While the cores are all clones, each tile has its own clock. This enables Intel to manage power (and heat) by running processors at one-quarter, one-half, or full speed. As a result, Intel can scale the processor's power consumption from 25W to 125W, depending upon load.

To enable cores with different clock speeds to balance workload requires a sophisticated way to share data, and Intel has put this in place on its chip. This could enable it to add other accelerators, such as graphics processors, in the future.

Intel also has its own take on its fabric network. Tilera opted for a routed packet network, which breaks data into chunks called packets and then sends them on their often-circuitous way. This is how the Internet and corporate networks work. Intel opted for a design that hybridizes routed packs with more direct switches.

"We're trying to explore the strengths and weaknesses of these designs," explains Jim Held, an Intel fellow and the company's director of terascale computing research. "One cost of packet-routed networks is that you need control logic at each node to handle the packets as they fly between the cores. You need storage while you decide where to send it next, and this all burns energy and that means more power and heat. So with our hybrid, we redesigned the router elements and revisited the interconnect itself. We tried to create a lightweight network that set up direct-switched communications between dies with the routing done ahead of time."

Software

Held's other concern is software. Intel, he noted, has long used parallel processing. This began decades ago, when processor speeds began to increase. At that point, it became faster to break instructions into threads, or small executable tasks, and then jump from executing one thread to executing another. When the processor completed one task, instead of waiting for data to arrive, it jumped to another. It eventually returned to the original thread, which now had new data ready and waiting.

"We were presenting the core to the software as if it were several processors and sharing the processing elements," says Held. He admits that managing this trick with 48 cores presents a much greater challenge. "When you have so many cores, especially if you're using a model where the cores operate on the same memory, it's complicated to keep them from tripping over each other."

Intel has responded by designing software algorithms that identify work that can be done in parallel, such as performing the same operation on all records in a database or lightening every pixel in a digital photograph. "As we go through an increasing number of cores, it becomes harder to get the full benefit of them without specialized software tools. We can run multiple tasks on three or four cores, but it's much harder to use all four cores for the same task and make the task run faster," Held states.

This is especially true for heterogeneous processors like the Cell, which is notoriously difficult to program. "One of the barriers to heterogeneity is whether a programmer can use it. Programmers love it when everything looks the same and they just have more of it," Held continued. "That way they can apply the same code to a technology they already know, instead of trying to map their problem onto a combination of things."

Intel has tried to make programming multiple cores easier by developing an extension for the popular C++ programming language. This enables programmers to write code using a language that they have already mastered. At runtime, a compiler looks at how they defined their operations and data and divides the work among many cores.

Today, multicore processors are moving toward homogenous cores tied together with mesh-type networks and linked to input/output ports. While the software that can take full advantage of multiple cores lags far behind the hardware, it may be good enough for network and cloud servers. As the software gets better, though, it is likely that even more functions will migrate onto processors.

Think of it as a city that has begun to sprout neighborhoods. Those neighborhoods may look alike now, like developments seen through an airplane window, but they are likely to grow more diverse in the future. Because adding diversity—more I/O, specialized processors, and accelerators—is the only way that tomorrow's processors will be able to keep up with Moore's Law.

Alan S. Brown has been an editor and freelance writer for more than 25 years and lives in Dayton, NJ (*insight01@verizon.net*). A member of the National Association of Science Writers and former co-chair of the Science Writers in New York, he graduated magna cum laude from New College at Hofstra University in 1974. He is an associate editor of *Mechanical Engineering* and contributes to a wide range of engineering and scientific publications.