by Alex St. John

Better Ideas In Computing

When I was studying electrical engineering in college in the 1980s, I wanted to pursue a career in parallel super computer design. The field fascinated me because there are domains of computer science problems that are not simply faster to compute with parallelism but in some sense are also only solvable with extreme parallelism. In that era, parallelism in computing was thought of as an efficiency tool for managing a series of fundamentally serial computing problems, and that mindset persists to this day. Searching a large database for a specific entry and computing a large spreadsheet are very common serial computing problems. Yes, you can improve the performance of these kinds of computations by subdividing the serial processing work between multiple processing units, but solving serial problems with greater parallelism isn't breaking any interesting ground in computer science.

3D graphics is one of an infinite set of computing problems that involves computation in which the outcome of many simultaneously interacting forces must be computed to generate a "realistic" result. These problems are all highly recursive in nature, meaning you can't calculate a result for one pixel on the screen without calculating a result for all of them. If an arbitrary limit on the quality and detail of a 3D scene were not chosen for these problems, they would require infinite computation to solve. Most physics, chemistry, and interesting artificial intelligence problems involve this kind of computations and are inherently hard to solve or even approximate with a modern serial processing computer. It was an interest in these kinds of problems that led me to the videogame industry and resulted in my efforts at Microsoft to create Direct3D and use gaming to generate a consumer market for massively parallel 3D hardware acceleration.

The reason a GPU *vastly* outperforms the most advanced serial processing CPUs at rendering 3D graphics is that they are wildly parallel processing chips specifically designed to solve intrinsically parallel computing problems. A GPU doesn't have two or four cores; it has millions. The most powerful and complex computer we know of, the human mind, has much in common with modern GPU architecture. It is for these reasons that I've predicted in the past that the GPU will evolve to become dominant and the CPU will ultimately be relegated to the role of glorified I/O processor.

There are no interesting future computing problems that GPUs are not better suited to solving quickly. Recent advances in GPU architecture to increase the general-purpose programmability of each processing unit and enable support for complex floating-point math clearly shows the march of processing emphasis away from the CPU and onto the GPU.

Although Intel frequently exhorts developers to thread their programs to take advantage of its multicore architectures, this is a fundamentally crude and inefficient approach to better performance. Placing the burden on developers to make otherwise simple serial programs take advantage of a little more parallelism quickly makes software more complex and less stable. I observe that many of the security and stability problems we see in Windows today are the result of trying to force a fundamentally serial computing device to support an extremely complex real-time OS environment. These systems are too complex for our primate minds to grapple with; the result is an unwieldy abomination like Vista. The burden of benefiting from parallelism should be shouldered by the tools and the silicon, not passed off onto developers to cope with.

This progression toward smarter development tools and greater processing parallelism will lead us inexorably to intrinsically parallel programming languages that automatically "thread" code for us to take advantage of massively parallel processing chips like the GPU to solve a whole new domain of computer science problems that are unapproachable with the tools and CPUs we rely on today. For example, what is it about a human brain that enables it to self-program to begin with, and then to pass along that property to machines?

If Intel and AMD are smart, they'll put a GPU onto the CPU die ASAP or risk waking up one day soon to find that they've been displaced by Nvidia. ▲

Alex St. John was one of the founding creators of Microsoft's DirectX technology. He is the subject of the book "Renegades of the Empire" about the creation of DirectX and Chromeffects, an early effort by Microsoft to create a multimedia browser. Today Alex is president and CEO of WildTangent Inc., a technology company devoted to delivering CD-ROM-quality entertainment content over the Web.

Send your feedback to thesaint@cpumag.com

CPU COMPUTER POWER USER

Reprinted with permission from Computer Power User *magazine.*
For more information, visit www.computerpoweruser.com or call (800) 848-1478.