A viewpoint from NVIDIA

If you are at Siggraph, there's a good chance you will be attending Intel's presentation today on Larrabee. If you're not at Siggraph there's a good chance that you'll still be covering it or have maybe been briefed already. With so much information and opinion currently out there, we felt it appropriate to offer a viewpoint on what has already been claimed and in the spirit of furthering the discussion, we'd also like to throw out some questions that we think everyone would benefit from having answered.

A lot of the current press releases and statements focus on instruction sets and "new languages" as the solution to parallel computing. Intel claims the X86 instruction set makes parallel computing easier to accomplish but as any HPC developer will tell you, this hasn't proven true with multi-core CPUs as applications struggle to scale from 2 to 4 cores. Now with even more cores, this same technology is claimed to solve parallel computing - we'd like to know what changed. After all, if it'll be easy to program 32 cores with 16-wide SIMD, why aren't more developers using quad cores with 4-wide SIMD? And if Ct is the answer, then why not use it on their CPUs?

The real challenge in parallel computing, in our opinion, lies in a different place. Developers have to decide how to divide a problem in parallel and then design software to use a parallel processor. GPUs have been used to solve one class of parallel computing - graphics processing - with a highly successful architecture. As graphics evolved, developers now write very sophisticated programs to do everything from graphics processing to physics within the standard graphics pipeline.

The next evolution of the GPU took place in 2006. The computer architecture group at NVIDIA added instruction sets and new architectural concepts to the GPU to make the computing architecture even more general. We call this the CUDA computing architecture.

CUDA is a C-language compiler that is based on the PathScale C compiler. This open source compiler was originally developed for the X86 architecture. The NVIDIA computing architecture was specifically designed to support the C language - like any other processor architecture. Comments that the GPU is only partially programmable are incorrect - all the processors in the NVIDIA GPU are programmable in the C language. Given this, why is Intel calling the CUDA C-compiler a "new language"?

Intel claims that the X86 base of Larrabee makes it seamless for developers. But with conflicting statements coming from Intel themselves on whether or not there will be a new programming model or not, there are several important questions.

- Will apps written for today's Intel CPUs run unmodified on Larrabee?

- Will apps written for Larrabee run unmodified on today's Intel multi-core CPUs?

- The SIMD part of Larrabee is different from Intel's CPUs - so won't that create compatibility problems?

NVIDIA's approach to parallel computing has already proven to scale from 8 to 240 GPU cores. This allows the developer to write an application once and run across multiple platforms. Developers now have the choice to write only for the GPU or write and compile for the multi-CPU as well. In fact, NVIDIA demonstrated CUDA for both GPU and CPU at our annual financial analyst day and ran an astrophysics simulation on an 8-core GPU inside a chipset, a G80-class GPU and a quad core CPU. Exactly the same binary program was used for the range of GPUs. And exactly the same source code for the CPU and GPU.

CUDA has been described as "hard for developers". Visit www.nvidia.com/cuda for a sampling of applications written in the C language for the GPU across a great number of fields. Virtually all of these applications were written by developers without any assistance from NVIDIA. They just downloaded the compiler, documentation and examples. Since CUDA runs across all NVIDIA GPUs introduced by NVIDIA over the last 2 years or so - developing a parallel application is simple and inexpensive. Plus with an installed base of more than 90 million C-language enabled GPUs, developers can already target a large base of consumer, workstation and now HPC customers.

To date, Intel has not described Larrabee's development environment. While focusing on one aspect of the architecture - the X86 instruction set - any differences or new challenges on top of the existing problems with multi-threading have yet to be revealed. With a new SSE architecture, new software layers to manage threads, perhaps another new language with Ct - developers are not simply using the X86 instruction set--they need to learn the way around a different computing architecture. Parallel computing problems are not solved with device level instruction sets, these problems are solved in computing languages with a computing architecture that is quick to learn and easy to use.

Computing on the GPU now has a critical mass of languages, developers and interest from the leading providers of operating systems.

Other Things that make us go hmmmm……..

Intel has spent a lot of energy telling the world that the GPU is dying or that it is not a growing market - why then are they investing so heavily on Larrabee and talking so much about it? Larrabee, like NVIDIA GPUs, has enormous floating point processing power. Wouldn't this encourage all the supercomputing clusters in the world to adopt GPU/Larrabee-style architectures and subsequently hurt Intel's CPU market?

Larrabee is positioning itself as a great GPU. Yet, users and developers alike have expressed frustration and disappointment with their IGP technology for many years. Why hasn't Intel used some of their investment and expertise to fix some of these problems for their 200M+ IGP customers? Also, will they be able to achieve the fine balance between both power and cost in graphics?

Ray Tracing (RT). NVIDIA's CUDA-enabled GPUs can do raytracing, and more. Even if Intel can do raytracing on Larrabee, how would developers achieve that on the hundreds of millions of Intel IGPs shipped each year? Is Intel abandoning its IGP customers?

In summary, Intel knows that moving to powerful, floating point rich parallel architectures is the future - in so doing they will inevitably encourage more developers to develop on GPUs as they too will see this move from Intel as a major industry shift and will want to target the hardware where their software has the greatest chance of success. NVIDIA will have shipped over 150 million CUDA capable parallel processors by the time Larrabee ships and Intel knows they will hurt their CPU business by making this transition, but this is truly the era of visual computing and this shift is a necessary move.