

June 05, 2008

## **In-Socket Accelerators -- When to Use Them**

by Kevin Urban, Director of Business Development, XtremeData, Inc.

---

Today's system architects have a tough enough job solving difficult architectural problems for applications like 40G line cards, HD Video Transcoding Systems, and next generation RADAR applications. However the most demanding part of their job is they also need to draw the difficult line between hardware and software; keeping manpower requirements in the equation, costs in check, and architecting a solution that can be built inside the market required timeframe. This is all in a day's work for some, but an almost impossible task for many.

Since software is generally cheaper to develop and the number of software engineers outnumber hardware engineers by almost an order of magnitude, the explosion of software-based solutions on x86 platforms has arguably become the de facto standard for many platforms. Many of these platforms were (and still are) built from the previous generations of CPUs with reusable software, and thus leveraged faster next-generation CPUs or multicores for their roadmap. That is, until now.

For 2008, the industry buzzwords are "hardware acceleration." CPU vendors are integrating custom integrated IP into their chips. AMD and Intel are creating an ecosystem for third party accelerators, named Torenzza and QuickAssist. GPU vendors are setting their sights on general purpose functionality. Meanwhile, a host of other chip companies, too many to mention, are developing new products that target this high performance computing (HPC) market.

A little known fact in the HPC community is that the embedded computing market has always solved their problems with a combination of CPUs and accelerators. Due to different space, weight, power, and environmental requirements for high performance embedded computing (HPEC), a large portion of those accelerators are implemented with field programmable gate arrays (FPGAs) from companies like Altera, Xilinx, and others.

As these two markets collide, the emergence of some of the world's largest companies from HPC and HPEC are now working together (Intel, AMD, Altera, and Xilinx) to make x86 CPUs with their own internal accelerators and easy to access external FPGA accelerators the solution of choice for both HPC and HPEC. As CPU-bound, software-only solutions require the benefits of hardware acceleration to remain competitive, architects must figure out what to accelerate and how to make the price, performance, and power trade-offs that meet market requirements.

### **What do I accelerate?**

In these examples, we will concentrate on FPGAs, but many of the concepts can be applied to other types of accelerators. A simplistic and very common place for designers to start is to profile your C/C++ code, find the routines that take most of your clock cycles, and start your efforts to increase performance or remove bottlenecks there. Hardware designers have done this for years and found the following types of applications that make sense to run on FPGA

hardware. Here are some typical applications that can be parallelized beyond a factor of 10x improvement:

- 1) Filters – FIR, IIR, Poly-Phase.
- 2) Fast-Fourier Transforms (FFT).
- 3) Encryption – AES, TDES, DES, etc.
- 4) Video Transcoding – MPEG2, H.264, VC-1, and others.
- 5) Compression – ZLIB, GZIP, etc.
- 6) Bioinformatics – Smith Waterman, BLAST, ClustalW.
- 7) Random Number Generation (RNG) – SOBOL and Mersenne Twister for Monte-Carlo.
- 8) Medical Imaging – CT Back Projection.
- 9) Packet and Network Processing (IPv6, Deep Packet Inspection).
- 10) Market Data – FIX, FAST FIX, OPRA, etc.

Another way to describe or find these algorithms in your code is to think of them in one of two ways:

- 1) Bit-level processing with deep instruction pipelines.
- 2) Vector-based processing of large amounts of data.

Both of these are trying to analyze data in ways that are not the standard 32 bit or 64 bit instruction, which creates overhead for the CPU or GPU because it has a fixed data and instruction set size. Additionally, many of these functions can have very deep instruction pipelines, which in an FPGA can be defined as deep as necessary. FPGAs can be programmed to be 3 bit machines in the example of BLAST that can be parallelized hundreds of times with instruction pipelines over one hundred deep. This creates a single resulting chip which can contain a 1000-core "machine" running at 300MHz or 100x faster than a single 3GHz CPU core.

Another example is encryption, which contains many XOR functions at the bit level, which again can be created easily in a FPGA, parallelized thousands of times, to create a machine that can encrypt data at 4GB/s (Bytes, not bits) using less than 50 percent of a large 65nm FPGA.

A secondary effect of FPGA technology is substantial power savings over today's high-end CPUs or GPUs. The largest FPGA built on 65nm technology can consume 25 to 30 watts of power versus a x86 CPU, which can run over 100 watts, or worse, a GPGPU, which can easily run over 200 watts. Combine this power savings with any of the examples above and you can have two orders of magnitude improvement on a performance/watt metric. This can be hard to ignore if you are in a design environment that has a limited power budget, like a UAV (Unmanned Aerial Vehicle), or a financial datacenter with major power/cooling concerns.

### **How do I design my FPGA accelerator?**

In order to design an accelerator, you need to decide if you need acceleration in the first place. If someone can achieve what they need with just moving from dual to quad cores or with a C-code rewrite, then that is usually the easier. FPGA acceleration and other exotic technology come into play for those designers who are really pushing the envelope.

The recent push towards industry-standard APIs for accelerators allows architects to build their own accelerated systems with CPU+FPGA rapidly, and avoid "single vendor" lock-in and end-of-life (EOL) issues. Additionally, open standard organizations like OpenFPGA ([www.openfpga.org](http://www.openfpga.org)) are making a concerted effort to help take FPGAs into the mainstream of accelerated computing. By taking an open standards approach for API's and benchmarks, and by bringing the community together, they are bringing the combination of x86 CPU+FPGA closer to mainstream HPC and even the desktop.

As an alternative to the standard programming API approach, many designers would prefer to just purchase a working acceleration technology and use vendor-supplied building blocks to implement their algorithms. Products like those from Cavium and LSI/Tarari are attractive here. Similarly, out-of-the box FPGA IP solutions like XtremeData's Floating-Point Vector Math Library, Fast-Fourier Transform, and Random Number Generation offer rapid development options. These solutions are pre-programmed hardware solutions that fit into the design flow very easily. No hardware expertise is needed to use them, which is why they are gaining a lot of attention in the marketplace. Plug them into your system, compile a library, or make a function call in your C/C++ routine, and you are off to the next task.

Finally, a third solution is to hire a hardware expert to help build your own secret-sauce algorithm. FPGA and ASIC consulting companies help customers build exactly what they want, make HW/SW trade-offs, and in some cases help train their customers to maintain and enhance the hardware design. If you need acceleration to be part of your differentiation, then this can be a very profitable and easy way to approach new technologies.

### **It's faster... now what? (How do you measure your performance, price, power gains?)**

Typically, designers who are looking at acceleration techniques are looking for about a 10x performance improvement of their total application to make it worth the effort. Depending on the goal of the system, the units of measure for success can vary wildly.

#### Power

For example, every watt that can be saved in a UAV will yield many different advantages to the system: Power and cooling savings, which equals less weight, which means that weight can be replaced by fuel, which means the UAV can fly further or faster, which might be critical to a new mission. Thus a solution that leverages an accelerator might be of equal or "only 2x the performance" to the existing system, but if that yields solutions that are half the size or power, then that "only 2x" savings becomes quite significant.

#### Performance

Another example of how applying acceleration techniques can pay off is raw performance. For example, Investment banks, hedge funds, and boutique trading firms know that increased performance of Monte-Carlo simulations for derivatives pricing and risk assessments can have big money implications. The key requirement in this space is maximum performance of double-precision floating point calculations. This is one of the main reasons why these institutions are

turning toward multicore CPUs, and when maximum performance is required, increasingly towards FPGA-based solutions. For example, today's largest 65nm Stratix III FPGAs from Altera can support up to 40 gigaflops of sustained double-precision floating point throughput at around 30 watts. Importantly, from the end-user point of view, the ability to finish these calculations faster and more accurately than your competitor can yield tens of millions of dollars in increased profits from exploiting arbitrage opportunities.

### Human Benefit

Other examples include FPGAs used for algorithms such as BLAST in the bioinformatics space, which if done faster, can help unravel the codification and functionality of genes and proteins, with major implications for the drug-discovery process. In medical imaging, back-projection algorithms implemented in FPGAs allow for smaller, cheaper, and faster x-ray CT scanners, which can put life-saving information in the hands of more doctors that need it. Increased precision in a next generation radar system can lead to better missile defense or earlier warning of possible threats, which can save lives.

### Other

In general, there are dozens of ways to measure acceleration success or lack of it. It will take some effort to evaluate, implement, and then to maintain them. However, many companies are having great success "however they measure it," using some of the techniques mentioned so far in this article. To help you find other terms for success, you can address these questions: Does 100x performance break a paradigm? Does this create an entirely new market? Does it provide value that cannot be defined by dollars saved, but by lives saved, breakthroughs made, or other similar metrics? These questions really measure the intangible value of acceleration, and the business-savvy people are the ones who can extract tangible value from these type of differentiators.

### **Think Differently: Re-look at the algorithm**

Lastly, the industry is beginning to realize that many of the implementations that we use today are the way they are because of 40 years of von Neumann programming and thinking. Since college (or even earlier) many of today's programmers have been taught, trained, and lead into thinking that a serial approach is the best and only approach to every problem. The majority of today's engineers tackle problems the way they were taught or the ways that have worked in the past. Now that the "core wars" are virtually over that 40 years of legacy is breaking. GPUs now have 128 cores and growing. FPGAs can be designed to have thousands of cores, and the word "accelerator" has become a mainstream term once again. Questions like "Can we use direct linear filtering rather than transforming to the frequency domain?"; "Can my accelerator put compression/decompression closer to the CPU so bottlenecks are removed?"; "Do I use double-precision floating point because it is easy or because I need it?"; and many others need to be asked and answered at the architectural level, before our preprogrammed way of thinking gets in the way.

Regardless of what questions your company needs to ask itself or how you decide to solve your problem, programmers and architects are realizing that having 8, 16, or 32 cores isn't going to solve all their problems anymore. Additionally, as new generations of FPGAs are available, they will have an increased performance per watt advantage over future generations of GPGPUs and manycore CPUs. An example is the recently announced 40nm Stratix IV family from Altera, which is again 2x larger and reasonably faster than previous generations. So step back, think of the algorithm, look at it through a parallel set of eyes, and enjoy. The world is parallel once again, shouldn't your implementation be?

### ***About the Author***

*Kevin Urban, director of business development, XtremeData, Inc., has been actively involved in the creation and marketing of advanced communication systems for over 15 years. He holds undergraduate and graduate degrees in electrical engineering from the University of Michigan and a graduate degree in business from the University of Chicago. Additionally, he owns patents for his work in wire-speed latency sensitive scheduling algorithms.*