The problem with disruption is that it is so... disrupting. It’s also what’s so attractive about it. The important question is what’s on the intended receiving end of that disruption? If it is your competitors’ business models and market positions, then OpenCL provides a remarkable lever with which you can actually achieve those objectives.

In short, OpenCL is a familiar, C-like language and framework that allows developers to exploit high-performance compute engines (OpenCL executes across heterogeneous platforms spanning CPUs to GPUs to FPGAs), like Pico Computing’s HPC modules and clusters. When OpenCL is targeted to such systems, it can dramatically transform a product’s value proposition—and get that value to market faster.

OpenCL is a breakthrough precisely because it enables developers to quickly and easily achieve significant acceleration in the real-time execution of their algorithms—particularly those that lend themselves to the considerable parallel processing capabilities of FPGAs (which in turn yield superior compute densities and far better performance/Watt relative to CPU- and GPU-based solutions).

From a product enablement standpoint, OpenCL becomes even more interesting as developers can now build truly optimized, “dialed-in” systems on which their software can run, as opposed to limiting product performance by running on generic, general purpose—and consequently low-performance—hardware. By hardwiring the program execution in dedicated hardware, the application can realize orders of magnitude improvements. Software running on a CPU—or even on general purpose GPUs—can’t touch that, especially when power consumption is factored in.

In the case of FPGA-based hardware, OpenCL code automatically compiles to the native description of the targeted device, e.g., Verilog and its synthesized bitstream file. These are details, though, from which the OpenCL developer is shielded. And that’s just the point: there is no need for the developer to understand hardware description languages (HDL) or any esoteric and proprietary FPGA design flows. Rather, developers simply get to reap the benefits that have formerly been the exclusive province of FPGA designers. In other words, OpenCL democratizes high-performance computing, making high-performance technology available to “the masses.”

Which brings us to 10 reasons why this ought to matter to you right now...
1. **It’s easy** • If you know C, then you essentially already know OpenCL. FPGAs, on the other hand, are famously difficult to design, especially via the respective vendors’ tool flows, which have proven to be a significant impediment to their wider deployment. As such, the many advantages of FPGAs have gone largely untapped by the many developers who lack these very specific skills. OpenCL changes all that: FPGAs now enjoy the programmability of X-86, putting extreme acceleration within reach of the average developer.

2. **It’s flexible** • OpenCL is hardware-agnostic—it can be compiled to CPUs, GPUs, DSPs, APUs, or FPGAs. It is up to the individual device vendors to create and support the APIs that allow OpenCL to compile to their respective devices. What’s more, once your code has been created, it can be targeted to any of the supported device families at any time for purposes that span performance comparisons to product migration. Whether you are targeting your code to server-class hardware or a desktop acceleration appliance the process is the same.

3. **It supports evolving designs** • FPGAs, because they are reconfigurable, allow high-performance systems to evolve without replacing hardware. And for those sections of a design that are subject to change throughout the design process, implementing those changes might be as simple as recompiling the code targeted to the affected FPGA(s). Even after the product is deployed in the marketplace, upgrades or custom configurations can be done remotely in the field, increasing the lifetime value and serviceability of the product. It’s a strategy that just makes sense.

4. **It delivers immediate performance improvements** • Can OpenCL produce the same sort of optimized results of hand coding? Of course not. But it may not have to. If you could realize a 10-50X improvement in performance in a matter of weeks using an OpenCL-based design approach, as opposed to a 100X improvement that may take many months of HDL coding, it just might be more than good enough. The combination of OpenCL with FPGAs can afford to waste some peak performance and yet still dramatically outperform CPUs or GPUs. This is due to the inherent parallelism of FPGAs—a hardware solution that breaks with the conventional sequential execution of CPUs, and performing far more work per low-power clock cycle. What would even a 2X improvement do for your competitive profile?

5. **It shortens time to market** • In terms of engineering management, the combination of a C-like language with reconfigurable FPGA-based hardware provides a new design paradigm that simultaneously reduces risk and shortens time to market. Here is a product design process that not only facilitates rapid prototyping, but provides the ability to produce a better product faster—and one that allows for painless incremental improvement. Using an OpenCL approach means you can get a product to market quickly, and then, to the extent that it makes sense to do so, optimize the FPGA implementations with hand-coded HDL. In either case, you’ll save months of development effort and realize revenue faster.

6. **It solves real design challenges** • With the high-performance computing world pressing aggressively toward exascale performance, you already know that the old CPU- and GPU-based product strategies won’t get you there. The requisite bandwidth, power consumption, memory performance, and resilience, among other ambitious design objectives, combine with time pressures to present a deep set of challenges. OpenCL targeted to FPGA-based high-performance computing systems provides a path forward. In the case of Pico Computing’s acceleration solutions, that path also includes easy scalability.

7. **It makes systems easy to maintain** • In an era of ever-shortening product lifecycles, application requirements can change every year—or even more frequently—which can present real difficulties in managing high-performance, algorithm-specific hardware. OpenCL targeted to FPGA-based systems eliminates this challenge. Again, because the FPGA fabric is reconfigurable, developers need only recompile new software; the
OpenCL code and the device frameworks take care of all the configuration and bitstreaming flows \textit{in situ}. In the meantime, because the OpenCL-based software is abstracted well above the particulars of the hardware, it makes it much easier to maintain diverse code bases—indeed, independent of their implementation in hardware. That takes risk out of the equation, too.

8 \textbf{It “thinks parallel.”} • OpenCL, according to its keeper, The Khronos Group, is “the open standard for parallel programming of heterogeneous systems.” The keywords here are \textit{standard, parallel}, and \textit{heterogeneous}—the three things that will move HPC forward in our post-CPU world. CPU-based serial computing has hit the wall. And while GPUs (also supported by OpenCL) have, to some extent, been exploited to address this wall with their parallel processing capabilities, they simply consume too much power. That leads us straight to FPGAs, where OpenCL’s inherent parallelism maps naturally to the FPGA fabric. At this point, the task is actually less about the OpenCL language and more about how you adapt your algorithm: you’ll port to the acceleration system only those segments of code that lend themselves to parallel processing, whether that parallelism is achieved via task or data parallelism—or both architectures simultaneously. Offloading these sorts of functions from the host CPU provides instant performance improvements while consuming less power. In fact, the host can be taken out of the loop on a great many operations as the FPGAs can pass data among themselves and to and from the outside world without any host involvement.

9 \textbf{It’s versatility protects your investment} • The versatility of the OpenCL/FPGA approach cuts many ways. In terms of applications spaces, the OpenCL/FPGA paradigm benefits all fields addressed by HPC, from high-speed trading to bioinformatics to cryptography to oil and gas and beyond. It applies in solutions that span products embedded at the edge of the network to the desktop to the data center. Put into the context of a PCI Express framework, it provides for modular and easily scalable high-performance computing resources. Also, because FPGAs are field-upgradable, keeping up with future revisions is easy. Moreover, it’s a design approach that leverages “use and reuse” strategies, making the most of existing OpenCL kernels and HDL libraries. And while you’ll also be able to conserve legacy code, you can also easily augment that code with new features and functions—and do so without disruption or risk.

10 \textbf{It delivers the best of two worlds} • FPGA technology charts an ideal middle path between conventional hardware systems (generic and low performance with little differentiation, but correspondingly low lead time) and ASICs (high cost, long lead time, least flexibility, and greatest risk). With OpenCL compiled to an FPGA-based system, you can get the best of both worlds: product differentiation with short time to market, and a flexible design process with low risk. When you cut time to market with a superior product, you’ll also realize revenues sooner—an important aspect that also must be factored into cost estimation.

If you’re up against tighter market windows, performance challenges, and fewer development resources, it might be time to consider a better way forward: OpenCL.

\textbf{About Pico Computing}

Pico Computing is the technology leader in high-performance computing. Our modular, highly scalable FPGA-based HPC and embedded systems solve the biggest of the big data computing challenges—from the edge to the data center to the desktop. Whatever your hardware acceleration needs, we make it work, we make it scale, we make it easy.