PowerEdge Product Group

Direct from Development

FPGA’s – Some Use Cases in the Data Center

Tech Note by:
Robert Hormuth
Duk Kim
Nelson Mak
Krishna Ramaswamy

SUMMARY
Emerging workloads and use cases will continue to drive the need for new and different compute.

Field Programmable Gate Arrays (FPGA’s) can be a part of this journey, when applied to the right problem.

FPGA’s are not a panacea for all compute problems, but when applied selectively and appropriately, they can deliver significantly better business outcomes, as described in this tech note.

Previously it appeared that the impact of Moore’s Law on Field Programmable Gate Arrays (FPGA’s) would be more profound than ever. It seemed that FPGA’s were never quite big enough, couldn’t run fast enough and were difficult to program. However, technology moves quickly and those attributes of FPGA’s have changed lot – they are certainly big enough now, clock rates are up, you can even get an embedded ARM core, and lastly the programming has improved a lot. OpenCL has made it easier and more portable. Note the use of the term “easi*er*” and not “easy” – but the results for the right problem make it worthwhile.

To find the right kind of problem that makes it worthwhile, let me do some context setting on where FPGAs work best – this is not an absolute but rather some high-level guidance. If we take a step back, it’s clear that we’ve been operating in a world of Compute Intensive problems – meaning, problems and data that you can move to the compute because you are going to crunch on it for a result. Generally, this has been a lot of structured data, convergence algorithms and complex math, and general purpose x86 has been awesome at these problems. Also, sometimes we throw GPUs at the problem – especially in life science problems.

A law of opposites

But, there is a law of opposites. The opposite of Compute Intensive is Data Intensive. Data Intensive is simple data that is unstructured and only used for simple operations. In this case, we want the compute and simple operators to move as close to the data as possible. For example, if you’re trying to count the number of blue balls in a bucket that’s a pretty simple operation that’s data intensive – you’re not trying to compute the next digit of π. Computing the average size of each ball in the bucket would be more compute intensive.

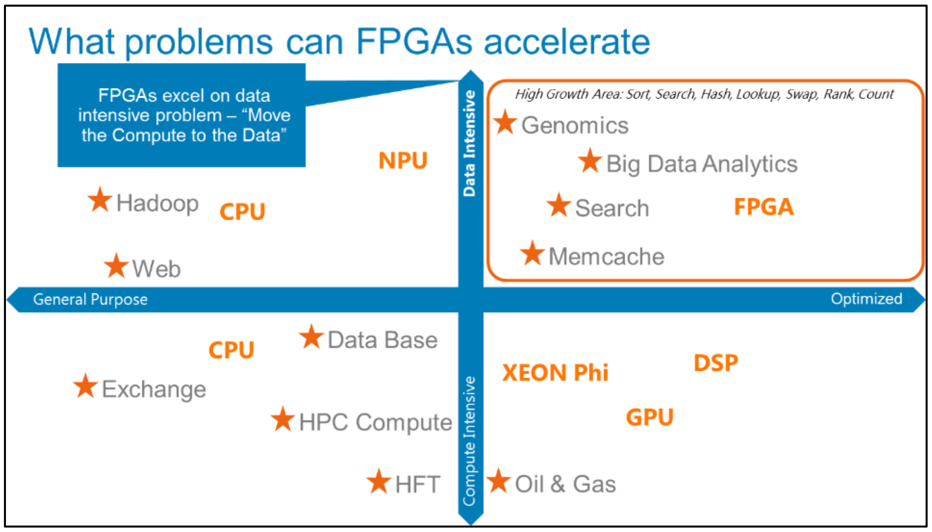


Figure 1: FPGA’s excel on data intensive problems shown in the upper right quadrant.

If we treat the spectrum of Compute Intensive and Data Intensive as one axis, and overlay that with the "law of opposites" axis of General Purpose compute and Optimized compute, we get the four quadrants in **Figure 1 above**, showing where various technologies fit best.

But why are CPUs not great for everything, and why do FPGAs need to be brought into the discussion? One reason is that CPUs are very memory-cache hierarchical centric to get data in and out from DRAM to Cache to registers for the CPU to do an operation – as it takes just as much data movement to do complex math as simple math with a general purpose CPU. In this new world of big unstructured data that memory-cache hierarchy can get in the way.

If you think about the link list pointer chasing problem shown in **Figure 2 below** – in a general purpose CPU when you need to traverse the link list every time you do a head/tail pointer fetch due to the data's unstructured nature, you get a cache miss, and thus the CPU does a cache line fill - generally 8 datum's. But only the head/tail pointer was needed, which means 7/8th's of the memory bus bandwidth was wasted on unnecessary accesses – potentially blocking another CPU core from getting datum it needed. Therein lies a big problem for general purpose CPUs in some of these new problems face today.

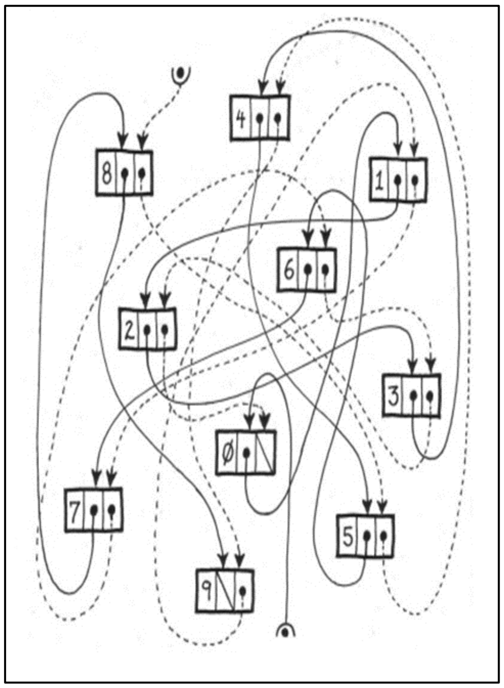


Figure 2: Schematic of a link list pointer chasing problem

Some real world examples

As mentioned earlier, programming is now simpler and easier (not simple and easy, but simpler and easier). Open Computing Language (OpenCL) is a framework in C++ for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, DSPs, and FPGAs. OpenCL provides a standard interface for parallel computing using task- and data-based parallelism. A quick example and flow is shown in **Figure 3 below**.

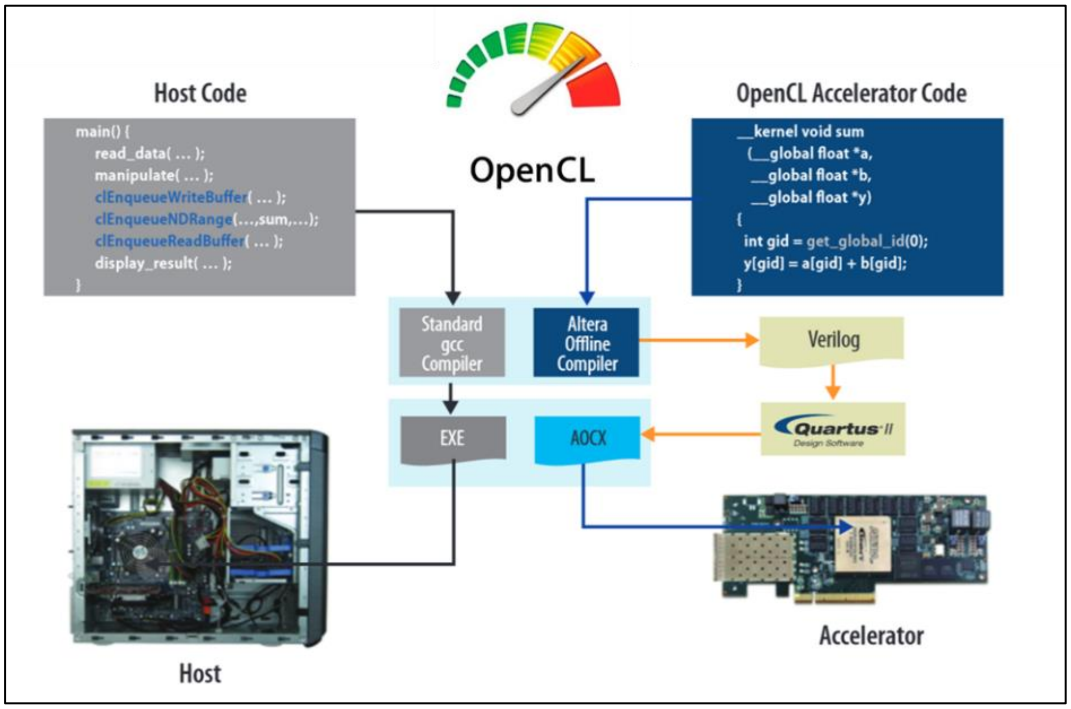


Figure 3: OpenCL provides a standard interface for parallel computing

Now, consider two examples that we've worked on in the PowerEdge Server Solutions Group to prove out FPGA technology and make sure our server platforms intercept the technology when it's ready.

Problem #1: "Drowning in pictures, save me….."
Say you're a picture gallery site, social media, etc., and want end users to upload full size images from their mega pixel smart phones, so that they can enjoy them on a wide range of devices/screen sizes. How do you solve this problem? The typical approach is using scale out compute and resize as needed for the end device. However, as shown above, it's not a great fit for general purpose compute, as it scales at a higher cost and you must manage scale out. Other options are batching processes and saving static images of all the sizes you needed – so it becomes a blowout storage problem. Or, force the end user device to resize, but you must send down the entire image – blowing out your network and delivering a poor customer experience.

To avoid any of the above options, we decided to do a real time offload resizing on the FPGA. For large images, we saw around a 70x speedup and about 20x speedup on small images. We replaced 20-70 servers into just 1 server and saved power, cost, and increased performance – easy TCO. So, now the CPU is handling the request for resized images and delivery but using an FPGA to process the images, as shown in **Figure 4 below**.

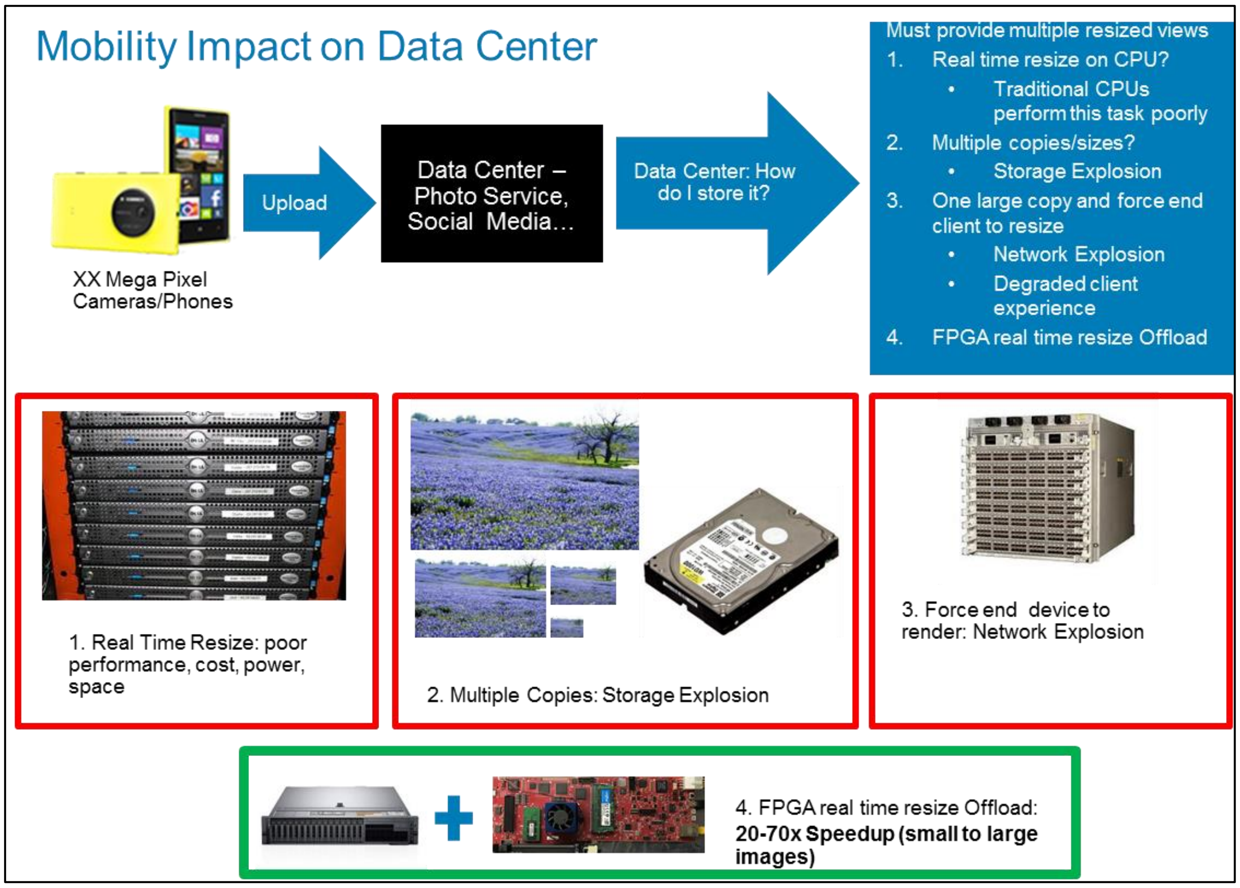
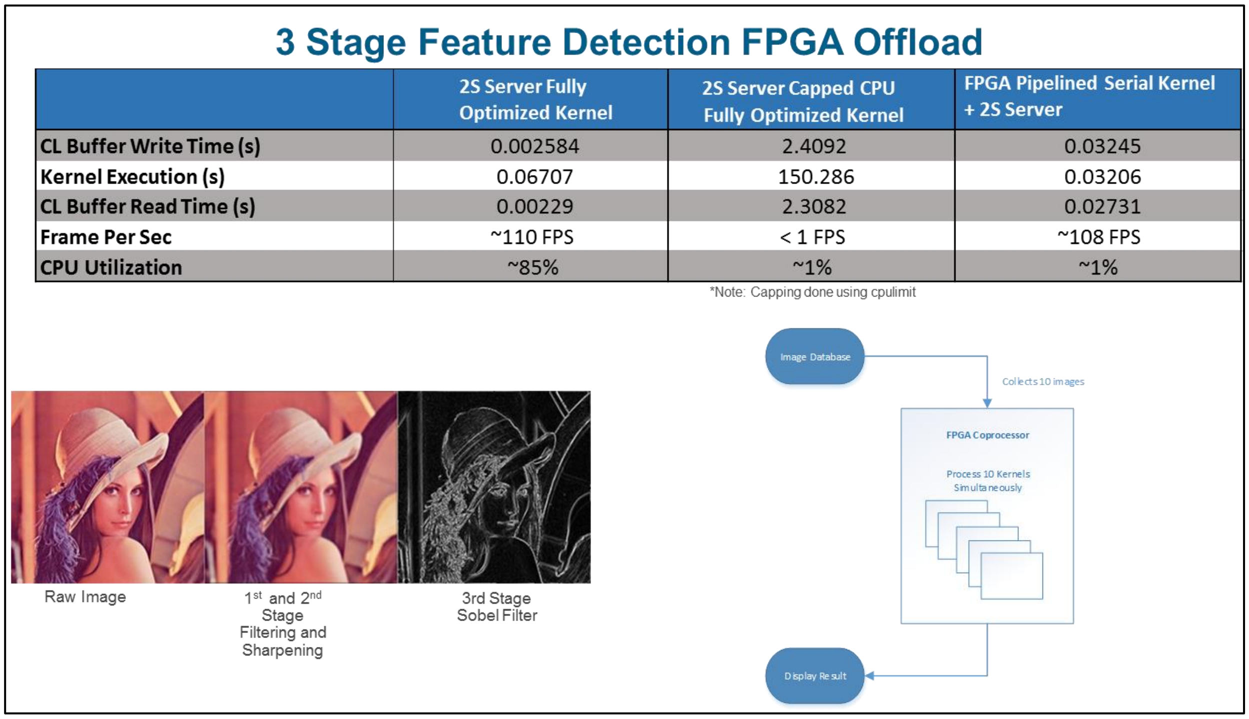

Mobility Impact on Data Center
XX Mega Pixel Cameras/Phones
Upload
Data Center – Photo Service, Social Media…
Data Center: How do I store it?
Must provide multiple resized views
1. Real time resize on CPU?
• Traditional CPUs perform this task poorly
2. Multiple copies/sizes?
• Storage Explosion
3. One large copy and force end client to resize
• Network Explosion
• Degraded client experience
4. FPGA real time resize Offload
1. Real Time Resize: poor performance, cost, power, space
2. Multiple Copies: Storage Explosion
3. Force end device to render: Network Explosion
4. FPGA real time resize Offload: 20-70x Speedup (small to large images)

Figure 4: Real-time offload resizing on the FPGA

Problem #2: "I have all these images, and I'd like to sort them by feature"

Digital content is everywhere, and we're moving from text search to image search. Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is also used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. In this example, we simply wanted to see what we could accomplish on a CPU and FPGA. We started on the CPU with OpenCL and quickly discovered that the performance was not up to par…. less than 1FPS (frames per second). The compiler was struggling so we manually unrolled the code to swamp every core (all 32 of them) and got up to 110FPS. But at 85% CPU load across 32 cores you could barely move the mouse.

The next step was the same OpenCL code (different #defines) and targeted an FPGA. With the FPGA and parallel nature of the problem we could hit 108FPS. In the FPGA offload case the CPU was *only 1% loaded*, so we had a server with compute cycles left to do something useful. To experiment, we went back to the CPU and forced a 1% CPU load limit and found we could not even get 1FPS. The point being made here is that in this new world of different compute architectures and emerging problems, "it depends" will come up a lot. **Figure 5 below** shows the data for the various results described:



3 Stage Feature Detection FPGA Offload

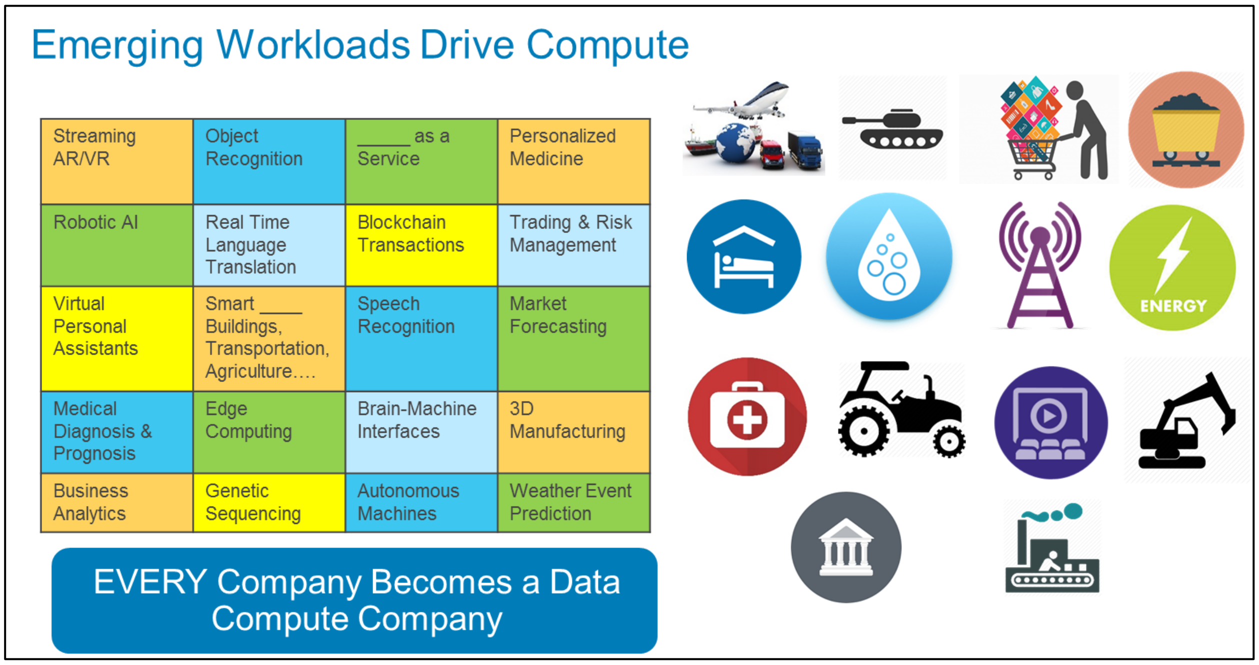
	2S Server Fully Optimized Kernel	2S Server Capped CPU Fully Optimized Kernel	FPGA Pipelined Serial Kernel + 2S Server
CL Buffer Write Time (s)	0.002584	2.4092	0.03245
Kernel Execution (s)	0.06707	150.286	0.03206
CL Buffer Read Time (s)	0.00229	2.3082	0.02731
Frame Per Sec	~110 FPS	< 1 FPS	~108 FPS
CPU Utilization	~85%	~1%	~1%

*Note: Capping done using cpulimit

Figure 5: In this example, the FPGA offload delivers outstanding performance for a specific task while freeing up the CPU for other purposes.

Future Problems

In the future, emerging workloads and use cases including those in **Figure 6 below** will continue to drive the need for new and different compute. Every company will become a data compute company and must optimize for these new



Emerging Workloads Drive Compute

Streaming AR/VR	Object Recognition	_____ as a Service	Personalized Medicine
Robotic AI	Real Time Language Translation	Blockchain Transactions	Trading & Risk Management
Virtual Personal Assistants	Smart ____ Buildings, Transportation, Agriculture….	Speech Recognition	Market Forecasting
Medical Diagnosis & Prognosis	Edge Computing	Brain-Machine Interfaces	3D Manufacturing
Business Analytics	Genetic Sequencing	Autonomous Machines	Weather Event Prediction

EVERY Company Becomes a Data Compute Company

Figure 6: Emerging workloads and use cases will drive the need for new and different compute.

uses. If not, they are open to disruption by those who embrace change more aggressively. FPGA’s can be a part of this journey when applied to the right problem. Machine learning inference, network protocol acceleration/inspection, image processing (discussed above) and other tasks are great examples of challenges that can benefit from the reprogrammable nature of FPGAs.

Summary

FPGA’s can be very useful in solving real-world problems. As the saying goes, however, “use the right tool for the right job”. That is, FPGA’s are not a panacea for all compute problems, but when applied selectively and appropriately, they can deliver significantly better business outcomes. To see how FPGA’s can benefit your business or organization, contact your Dell EMC Sales Representative. Maybe we can help you, too.