Low Cost OpenCL™ Acceleration For Science And Engineering Projects

Presented By
John Adair

Why Accelerate

- Processors not increasing performance significantly
- We need more complex processing e.g. autonomous vehicle
- We need answers more quickly e.g. genome results
- We want to be green and use less power
- We want to beat that other guy
- We want to make more money

Traditional large scale FPGA Acceleration

- Expensive entry point
- Can be BIG
- You might need a new power substation
- You need lots of air conditioning
- Expensive custom FPGA development

Traditional small scale FPGA Acceleration

- Entry cost low
- Great power/performance
- Lots of FPGA development time and cost
- Needs FPGA engineer

GPU Acceleration

- Low entry cost
- Uses x10-20 power of FPGA solution
- Relatively easy to use with OpenCL™ and CUDA™ programming
- Power/heat/reliability limits where it can be used

Project FPGA Problem

- I’m not an FPGA engineer
- FPGA engineers are hard to find
- I have to describe what I want to do to an engineer
- I have to wait for the engineer to be available

Project GPU Advantage

I can write my algorithm myself now in C/C++ using OpenCL or CUDA.

FPGA Fightback

- OpenCL for FPGA
- HLS for FPGA (more control aimed at level FPGA engineer)

Starting OpenCL

- FPGA Board with communications link – PCIe, USB, Ethernet
- FPGA BSP giving you OpenCL kernals
- SDK

Doing A BSP

- Needs a good understanding of vendor FPGA tools and timing closure.
- Needs an understanding of software, scripting and software tools.
- Documentation available but limited.

Boards With Existing BSP

- Often expensive Stratix or Virtex based
- Some FPGA boards with ARM cores have support
- Our project bringing low cost Cyclone™10-GX with PCIe and 10G Ethernet support now and USB3 coming.
- Cyclone10-GX USB3 standalone soon

OpenCL Performance

- Don’t expect this to be as good as a design optimised by an FPGA engineer
- It won’t be as fast
- It will use more resources
- It will use more power

OpenCL Win

- You get a working design fast
- You get a working design cheaply
- You get results fast
- You beat the other guy
- Boss gives you a rise for delivering on time

OpenCL Operation

- Common approach is to keep a “live” coms interface and partially reconfigure rest of FPGA with OpenCL kernals.
- Allows kernals to be changed but will have some latency in changeover.
- Simple version - data interchanged via a dual porting memory.

When To Not Accelerate with OpenCL

- When the combination of data transfer latency and FPGA processing is more than your equivalent software routine.
- PCIe has latency
- USB has latency
- Ethernet has latency

When To Accelerate With OpenCL

- When software routine is long or has a lot of iterative actions e.g. CRC32 calculation of input data 32 or 64 bit wide.
- Math/DSP functions
- Streaming data processing e.g. facial recognition with improved frame rates.

Multiple Threads

- OpenCL can support multiple threads on one or more kernals.
- A single thread can queue actions.
- Threads can run in parallel concurrently on multiple kernals.
- Can get more throughput using pipelining.

Mathworks™

- Going from designing a function in Mathworks to running on OpenCL is possible.

Questions