# Adaptive Optics Real-time Control in the ELT Era

Durham University
Centre for Advanced Instrumentation

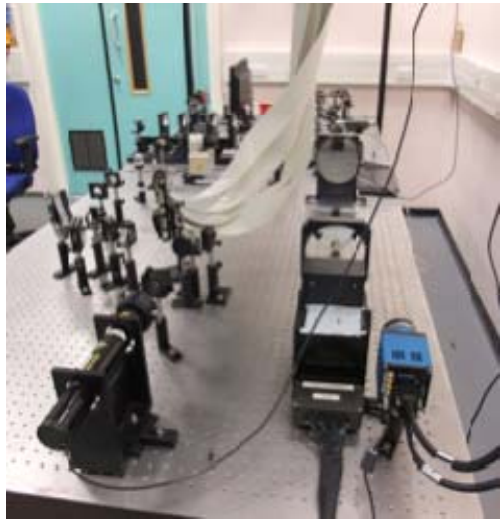
Nigel Dipper
Durham University

# Overview

- General thoughts on RTC for ELT scale AO
  - Durham CfAI research program – DRAGON test bench
- The problem
  - Low latency, low jitter, massive computing power
  - EPICS! – See Doelman talk later today
- The hardware
  - Pixel handling - Smart cameras?
  - FPGAs, GPUs etc
- Abstracting the Hardware
  - Software development tools - OpenCL
- Simulation and Real-time control
- The Software crisis

# DRAGON – Test Bench

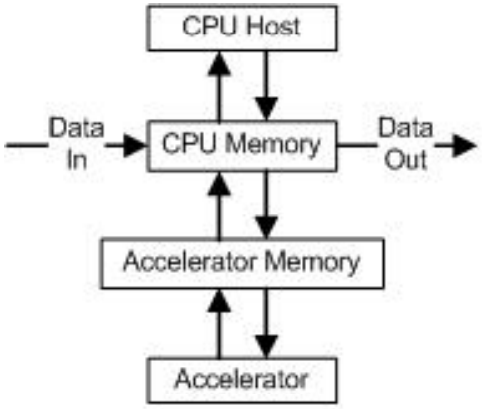
- Multiple WFS cameras
- Atmosphere simulator with multiple phase screens
- 1024 Actuator Boston DM
- 97 Actuator Xinetics DM
- DARC RTC



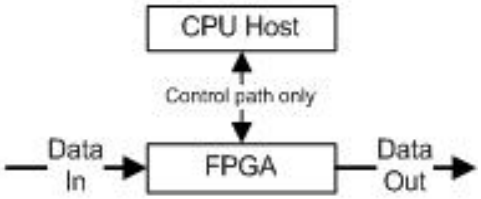
# Hardware

- CPU, GPU, FPGA, a hybrid or something new?
- What drives the decision?
  - Latency and jitter impact on AO error budget
    - Enrico Fedrigo talk
  - The development cycle speed
  - The cost
- What path to follow?
  - Commercially available hardware
  - Watch the roadmap of HPC and data centres
  - Many of the requirements are similar
  - BUT budget (EG at Google) may be higher

# Embedded vs Acceleration - Jitter




Accelerator Data Flow




Embedded Data Flow

**Data flow cartoons**

# Pixel Handling

- We would like 'Smart Cameras'
  - Pre-process pixel data in the camera? (In FPGA?)
  - Current cameras use 10 tap CameraLink (~7Gb/s)
    - But: bulky cable – We would prefer fibre out
  - Ethernet cameras – We already use them – GigE vision
- Remote pixel handling may then be acceptable
  - So do pixel calibration and SH centroiding in FPGA?
  - This is the SPARTA solution used for SPHERE
  - Or can we do it in GPU?
    - The GPU is an accelerator, not embedded
    - Or is it? GigE vision to GPU via GPUdirect

# Current DRAGON test camera

- PCO.Edge – sCMOS
  - 2560x2160 pixels
  - Frame rate 30/100 fps at full frame
  - 10-tap CameraLink

- Silicon Graphics microEnable IV VD4-CL frame grabber
  - 4 lane PCIe (2) – 900 MB/s
  - Programmable on-board FPGA

- Use commercial frame grabbers?
  - No 'in-house' controller or camera??

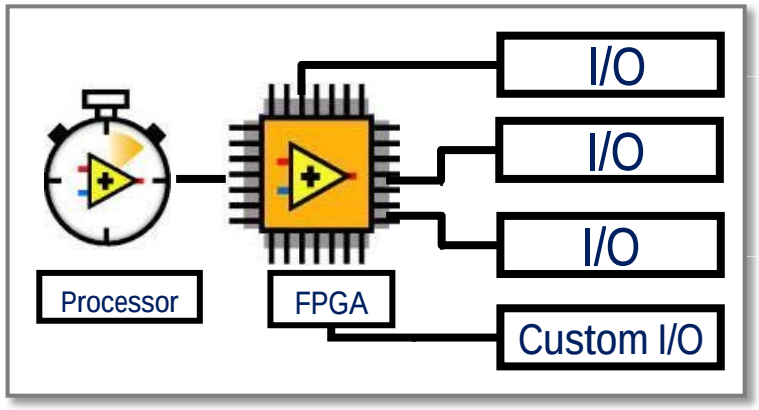
# Pixel handling in FPGA

- Problem: Development cycle is long

- Solution 1: Do it in LabView!
  - National Instruments provide an integrated (expensive?) solution
  - We have tried this using:
  - LabViewFPGA + FlexRIO + NI FPGA card + CameraLink core
  - Full ISE toolkit is used but abstracted from the programmer
  - We have read out our sCMOS camera via FPGA programmed entirely in LabViewFPGA – VERY quickly. Not yet sure how efficient such code is…

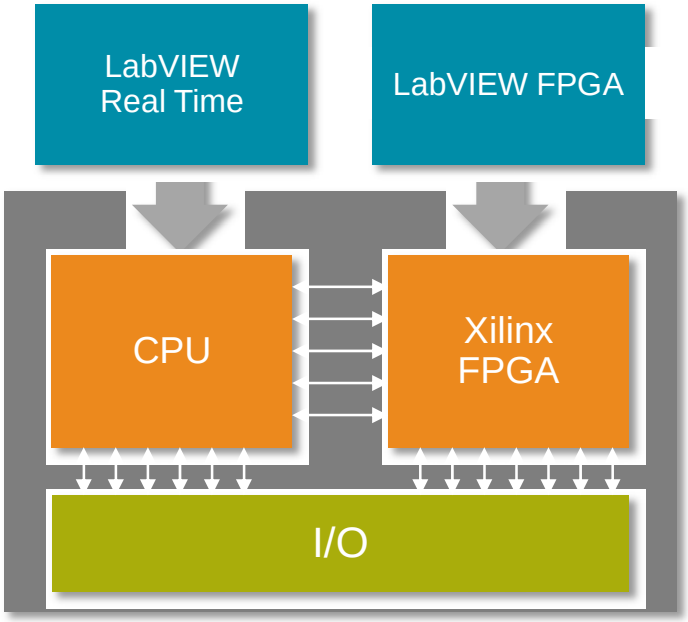
- Solution2: Do it in OpenCL? – Return to this later…

# System Design Today with Xilinx + NI




## LabView User Experience

- Hardware design details hidden
- Focus on innovation; not implementation
- Design reuse with 1000s of designs




## Customers Are Asking for More

- Integrated design and debug
- Seamless SW-HW interface architecture
- Scalable family of platforms
- Scaling to deeply embedded
- From prototype to high volume
- Lower power and cost

# Nallatech

- An FPGA accelerator card…
- PCIe-385N - Altera Stratix V FPGA Computing Card
- PCIe Gen-3 8 lane
- 16 GB DDR3
- dual ports supporting 10GbE
- Supports OpenCL

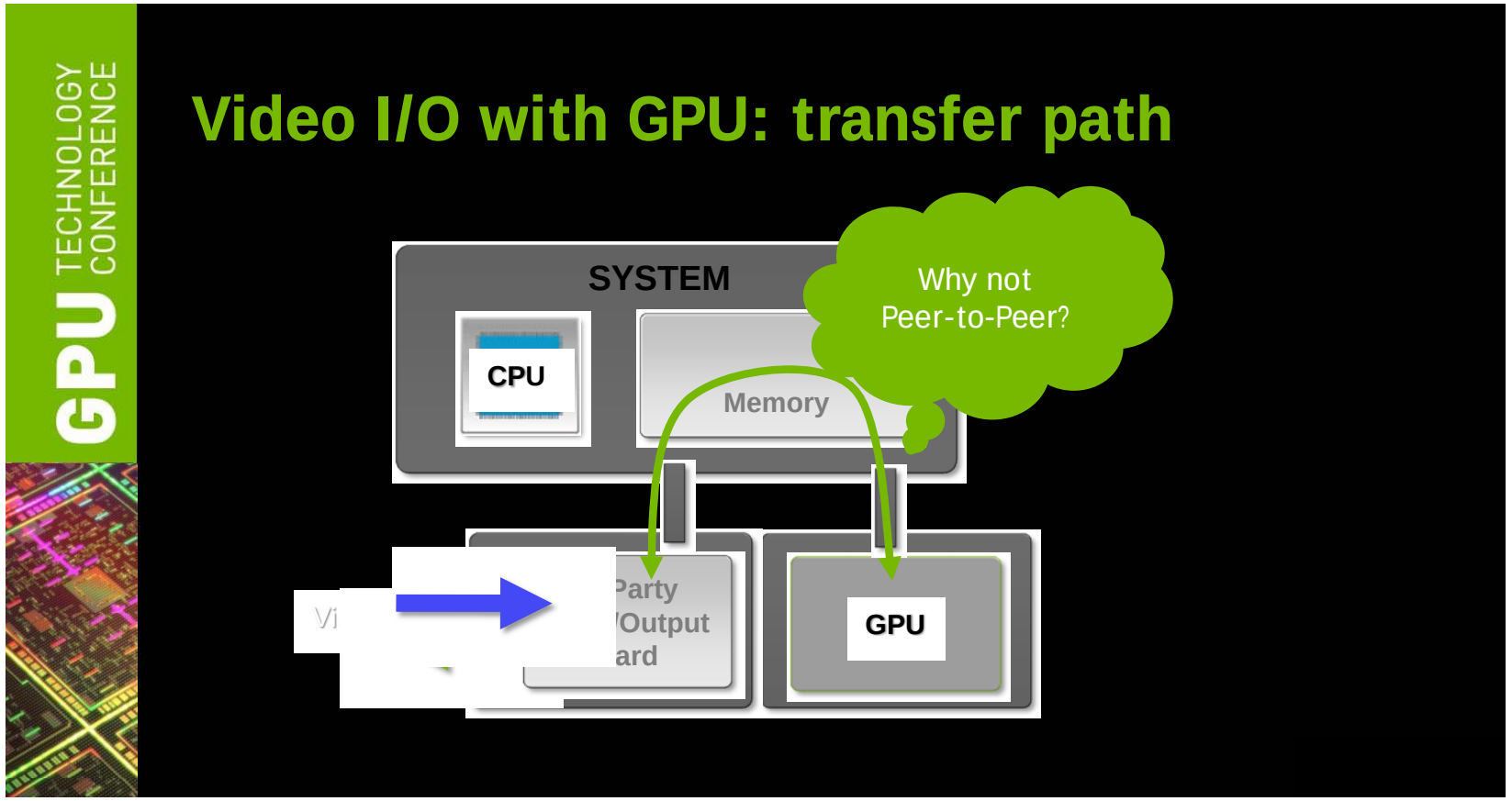
# Pixel handling in GPU

- Problem: We need direct data transfer

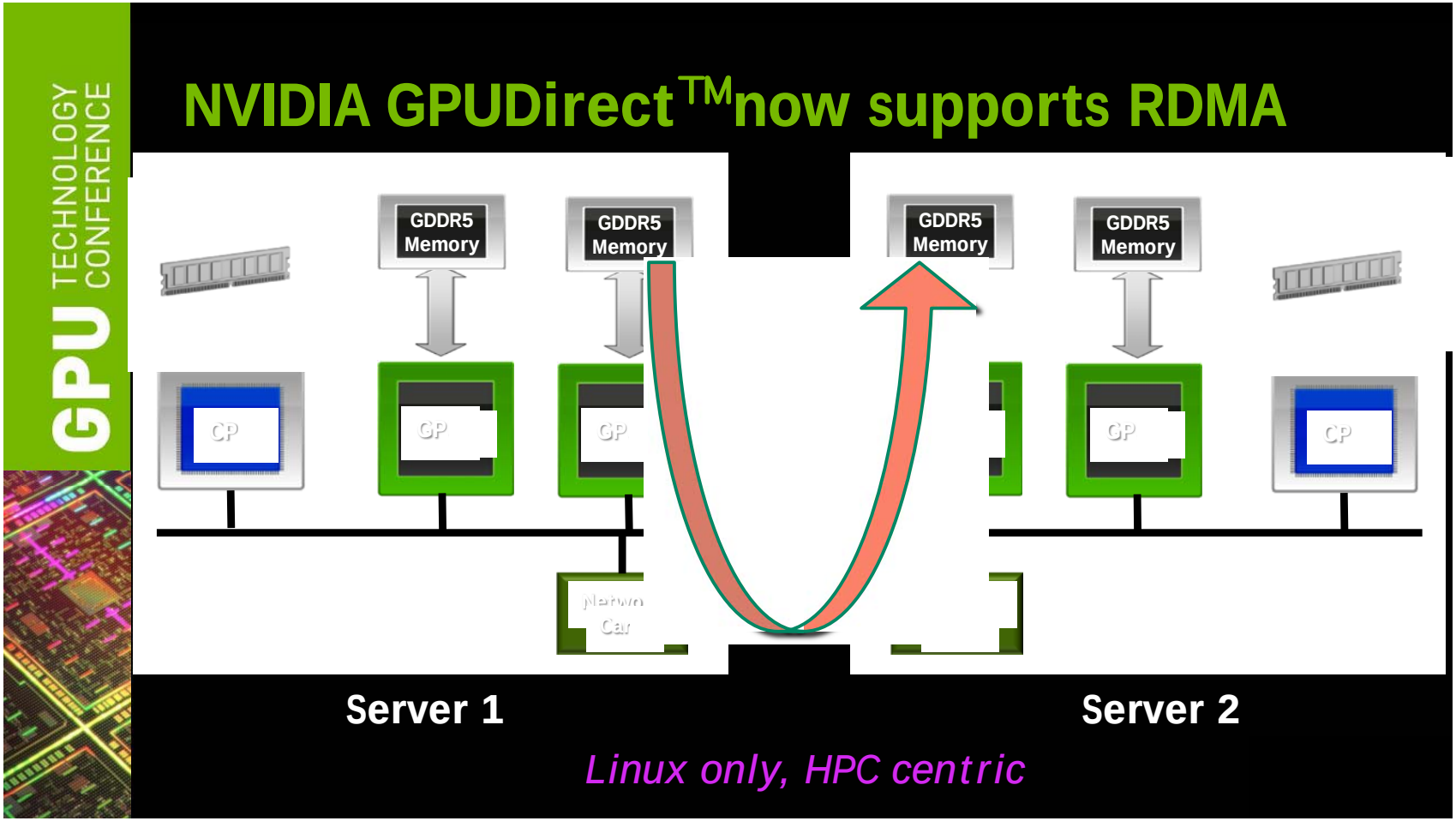
- Solution: GPUdirect
  - Peer to peer data transfer
  - Initially between GPUs
  - Now from interface cards both Infiniband AND Ethernet cards
  - This will be tested on the DRAGON testbench at Durham

- Allows fully embedded GPU based data pipeline
  - Not yet implemented

# Current input path – Via CPU



# Pixel data via GPUDirect



# Are GPUs the 'Silver Bullet'?

- NO!!! – But they are an excellent option in 2012
- Kepler GPU: ~1500 cores each + Dynamic Parallelism

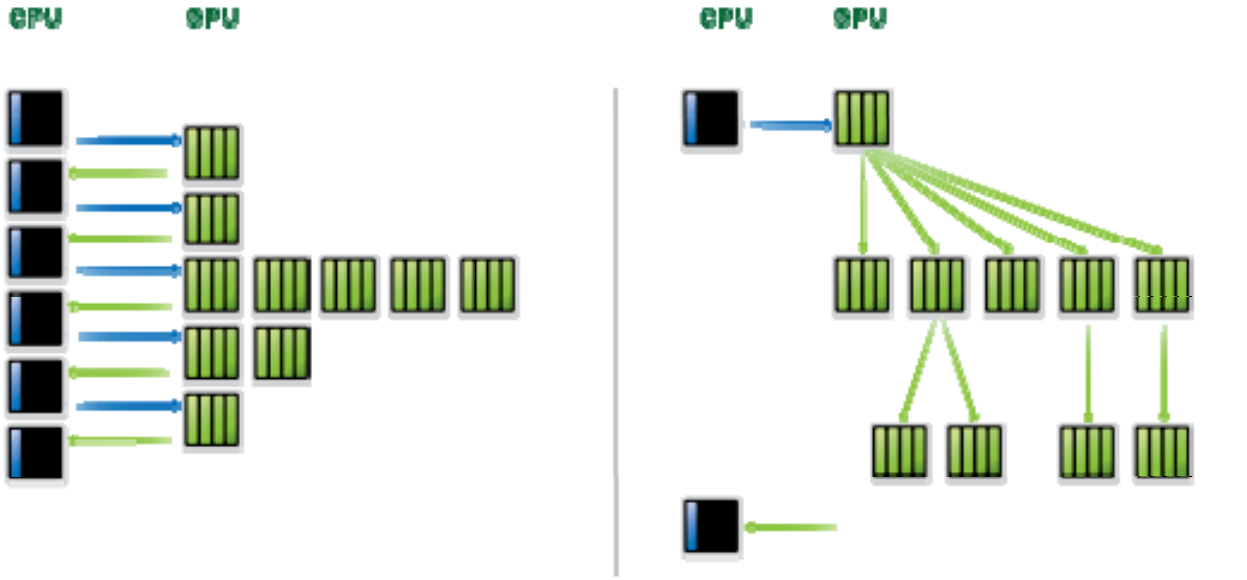



Figure 3: Without Dynamic Parallelism, the CPU launches every kernel onto the GPU. With the new feature, Kepler GK110 GPU can now launch nested kernels, eliminating the need to

# Hardware Abstraction

- ELT RTC is likely to be a heterogeneous computing environment (cf SPARTA)
- All code must be abstracted as far as possible from the hardware
- 2008 – OpenCL appeared
- 2012 – Can now be used on both FPGAs and GPUs
- Standard C library with OpenCL extensions
- Some early results:

# OpenCL on GPUs

- Work by Durham student – Andrew Richards
- MVM wavefront reconstruction in two different GPUs:
  - Nvidia GT 525m (Gforce)
  - AMD Radeon HD 7970
- Matrix size: 10240 x 10240
- Code in OpenCL is unchanged when ported between cards
- Standard BLAS library used in both cases
  - clAmdBlas on the AMD; cuBLAS on Nvidia
- No exact efficiency figures yet but:
  - OpenCL version on Nvidia is slightly slower than the CUDA version
- OpenCL code can also run CPU
  - Just change target device type. No code changes required

# OpenCL on FPGA

- Nallatech PCIe-385N
  - Supports OpenCL
- PLDA XpressGX5LP
  - PCI Express FPGA card with quad-10Gb Ethernet
  - Data centres use these…
  - OpenCL??

- To be tested…

# Abstraction: Conclusions

- YES!!! – We need abstraction

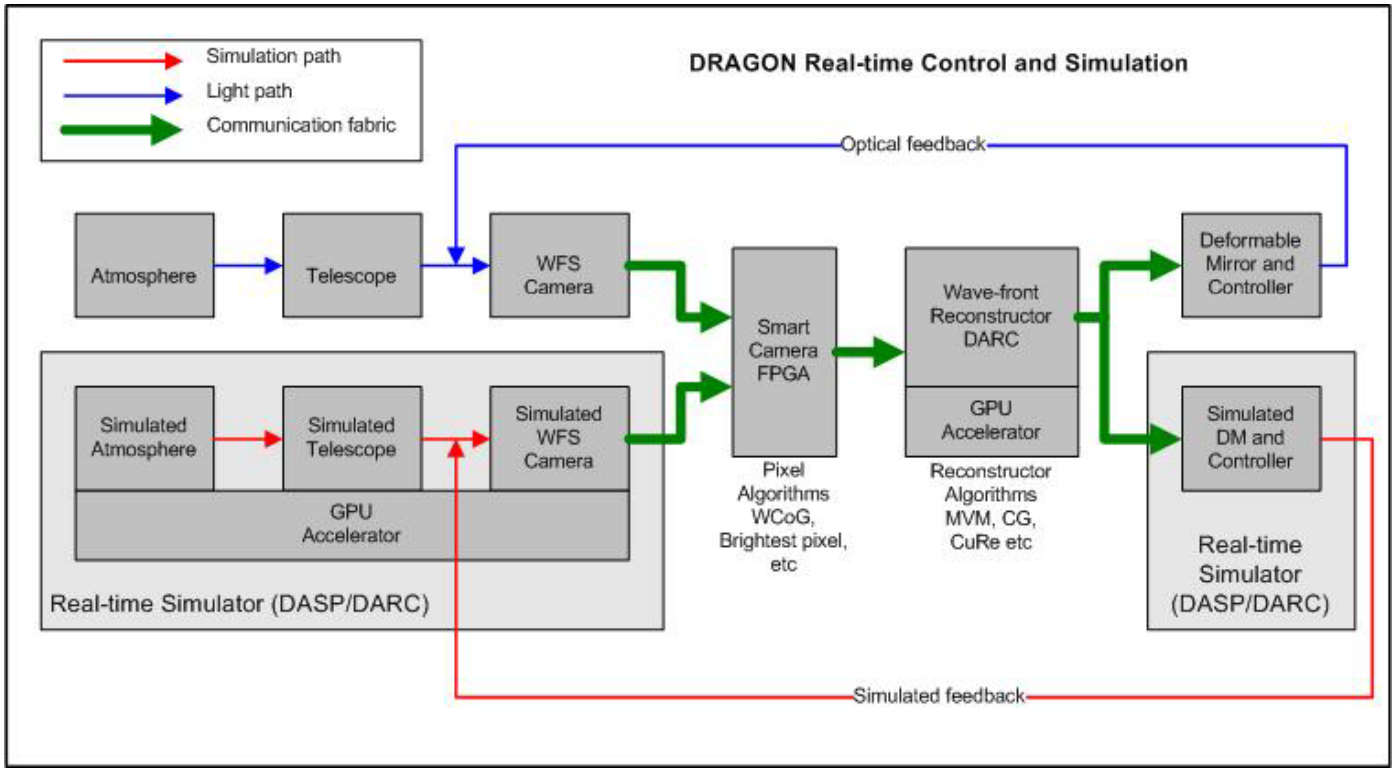
- Could take many different forms
  - LabView is an option for FPGAs
  - OpenCL looks very promising
  - On GPUs, CUDA based abstraction methods are being tested in Paris at LESIA

- See talks by A. Sevin and D. Gratadour

# Simulation and RTC

- Synergies between simulation and RTC code
  - At Paris: D. Gratadour talk
- At Durham, we propose combining DARC and DASP code to provide hardware simulation on the DRAGON bench
- AIM: RTC testing at full frame rates both in the presence of and in the absence of hardware components
- R & D program now funded by STFC to 2015

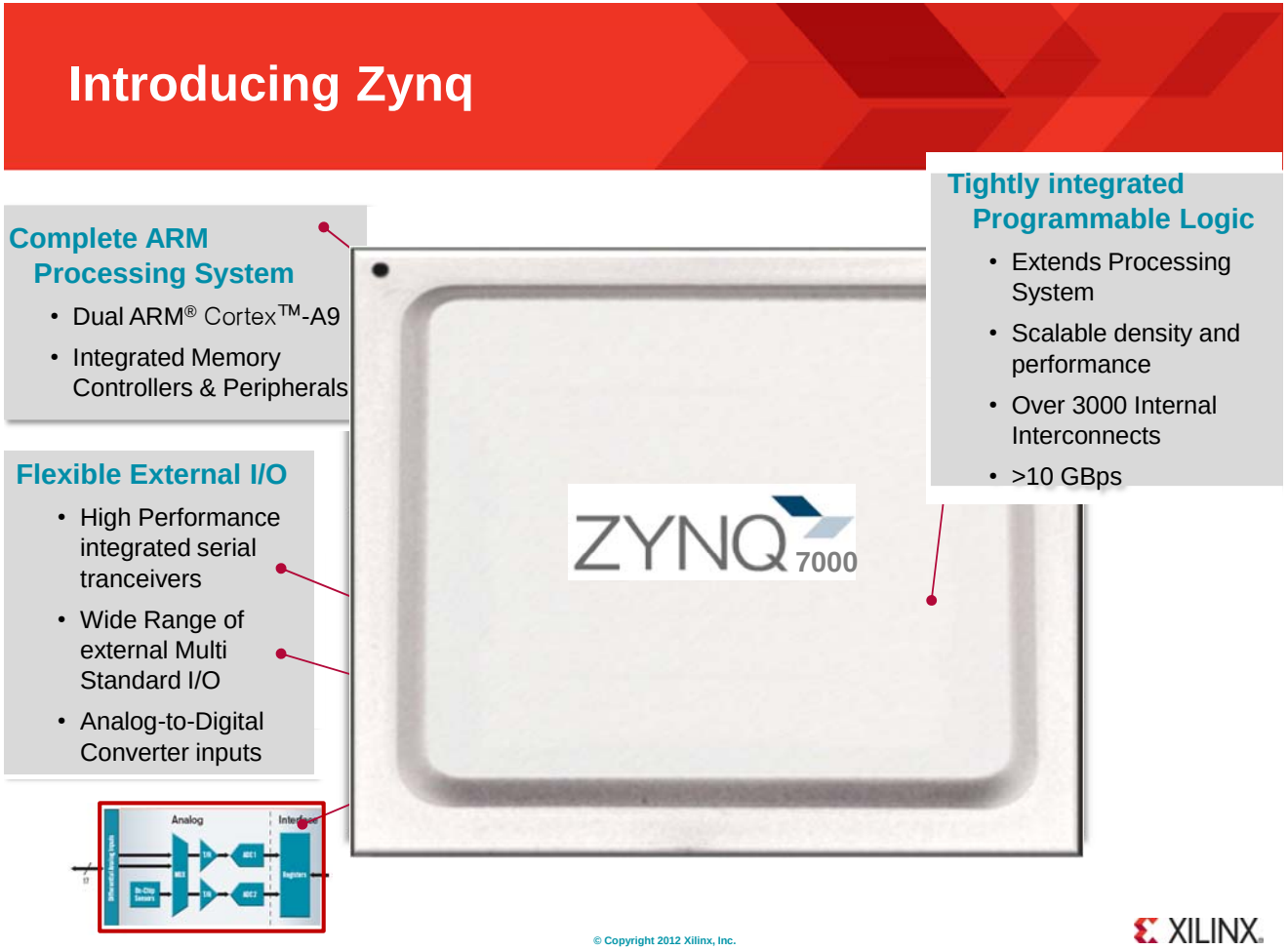
# DRAGON Integrated RTC and Simulator



# New hardware…

- There will be plenty by 2020!
- If we are sufficiently abstracted, we will just buy it and plug it in with the existing software.
- EG – The Xeon Phi – CPU based – 64 cores
  - Develop in C (or OpenCL)

# New FPGA/CPU hybrid:The Kitchen ZYNQ



# The 'Software Crisis'

- Software budget for all AO projects is always under-resourced
- Reusable code – We always say we will but rarely do!
- ELTs will require a vast software set
- The 'hard real-time' is always a small fraction of the total software.
  - Soft-realtime: Calibration, configure and control (*CCC*); optimisation
- 'Instrument abstraction' – A core of ELT code that is common to all AO enabled instruments
- There will be 'requirements creep'
  - We need the resources to handle this

CfAI
Centre for Advanced Instrumentation

Durham University

# Conclusions

- Commercial hardware solutions following the Datacentre and HPC roadmap
- Hardware abstraction – to be future proof
- Rapid development cycles – To reduce cost
- RTC requires advanced simulation capabilities

**Software resources !!!**