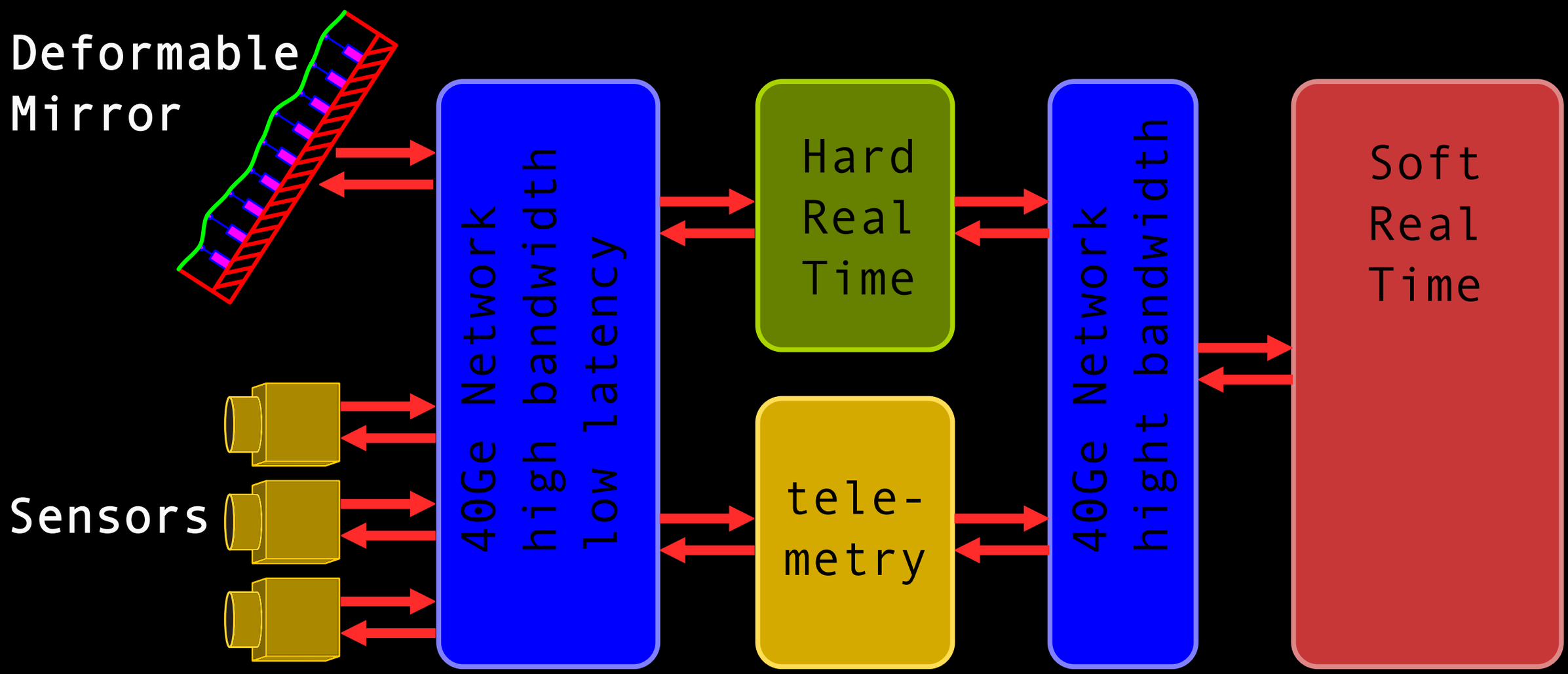
Low-latency data acquisition to GPUs using FPGA-based 3rd party devices

Denis Perret, LESIA / Observatoire de Paris

Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

RTC_4_AO workshop
PARIS 2016

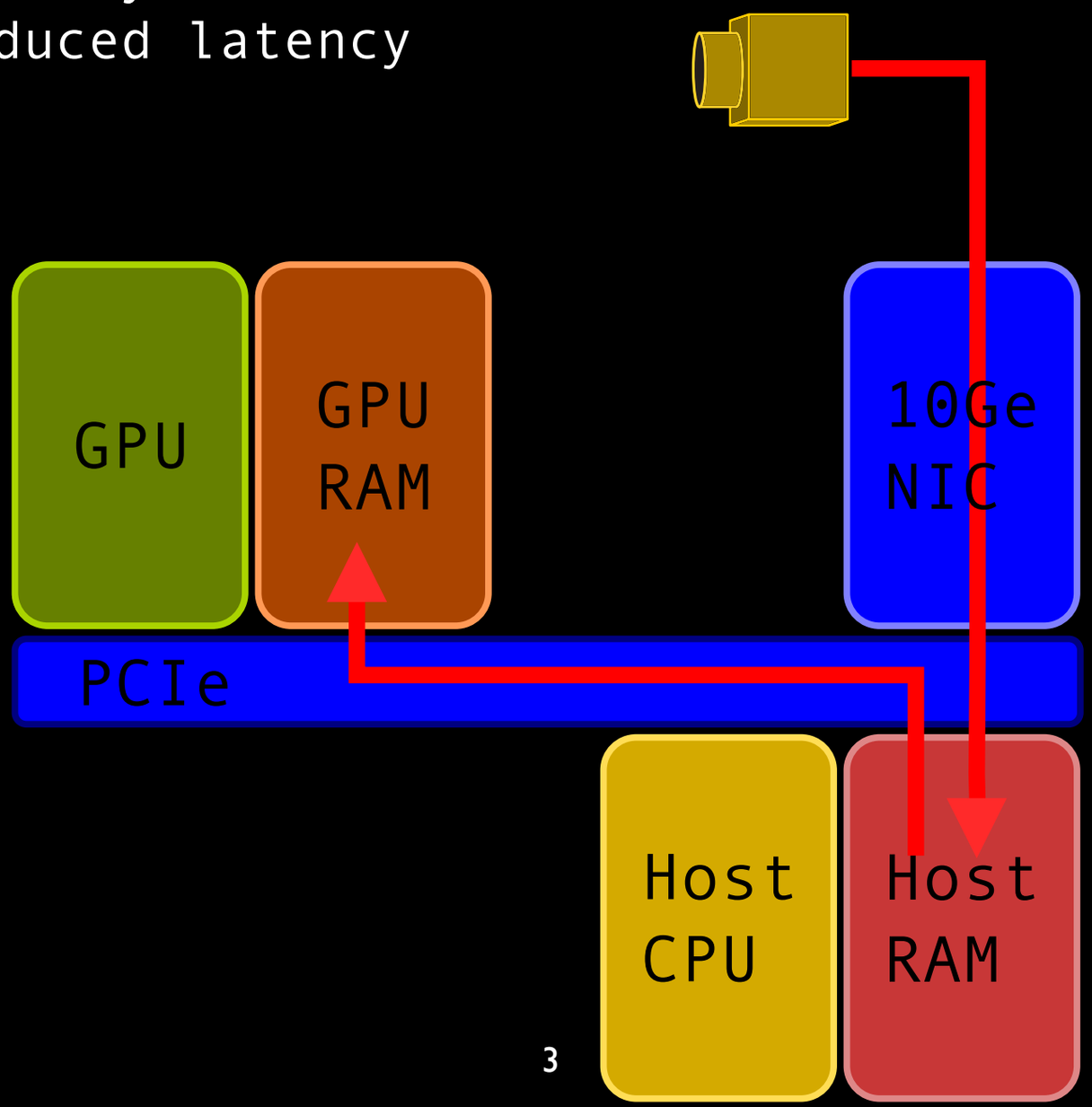
RTC for ELT AO



Using FPGA to reduce acquisition latency

~ Interface : 10 GbE, low latency acquisition interface

~ W/O Direct Memory Access to the GPU ram : multiple data copy = introduced latency

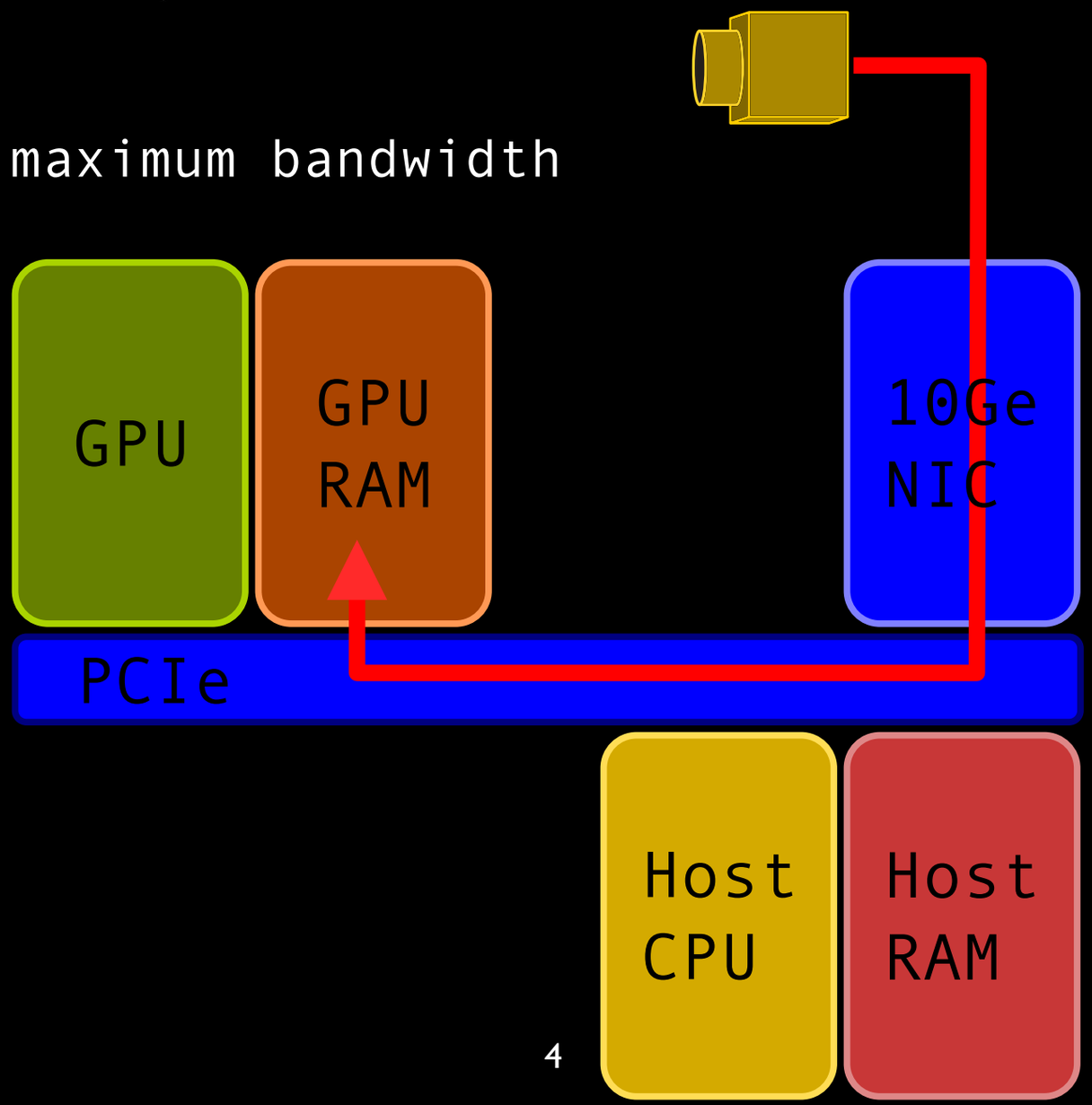


Using FPGA to reduce acquisition latency

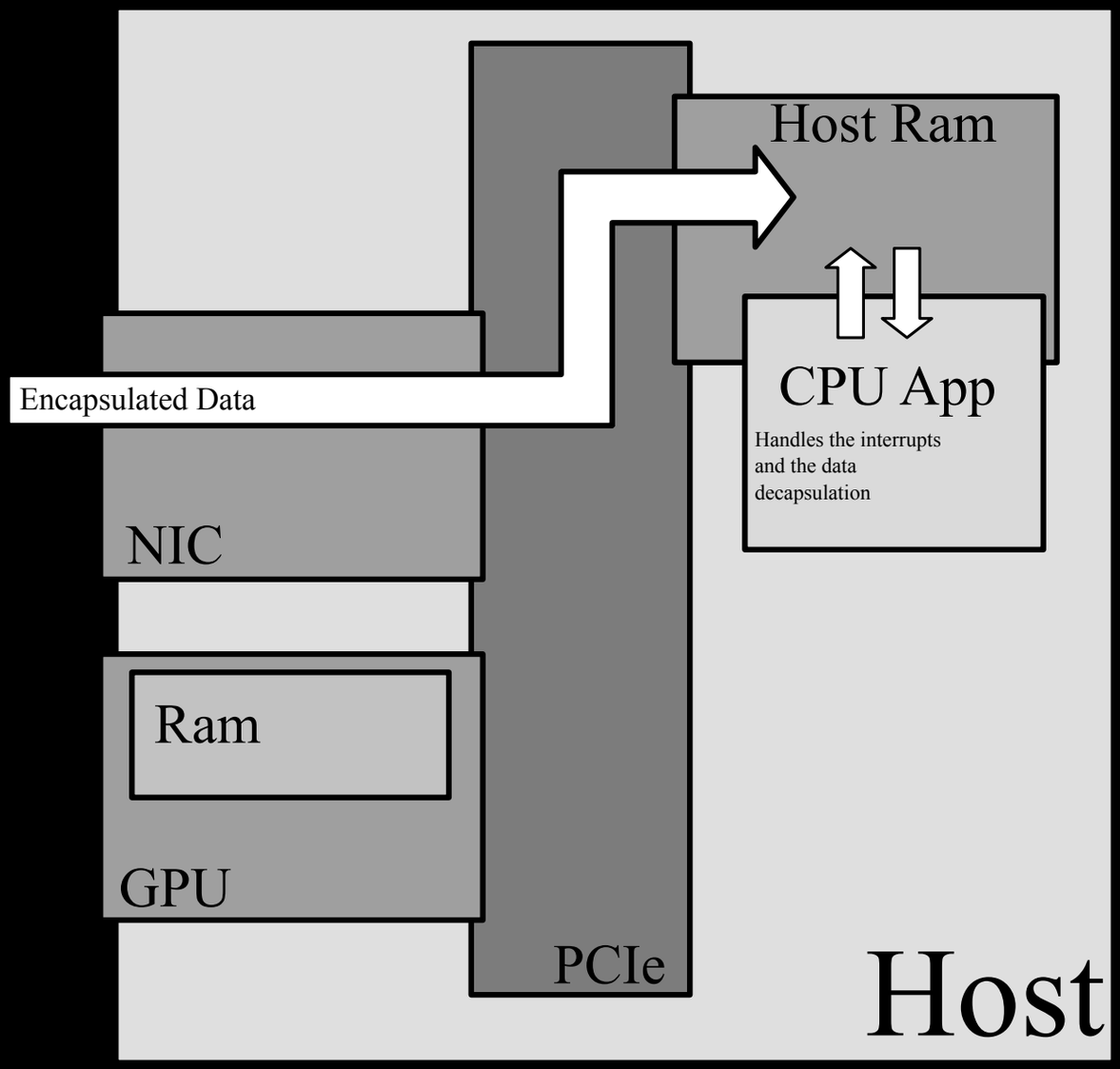
~ Interface : 10 GbE, low latency acquisition interface

~ With Direct Memory Access to the GPU ram : single data transfer

= low latency, maximum bandwidth



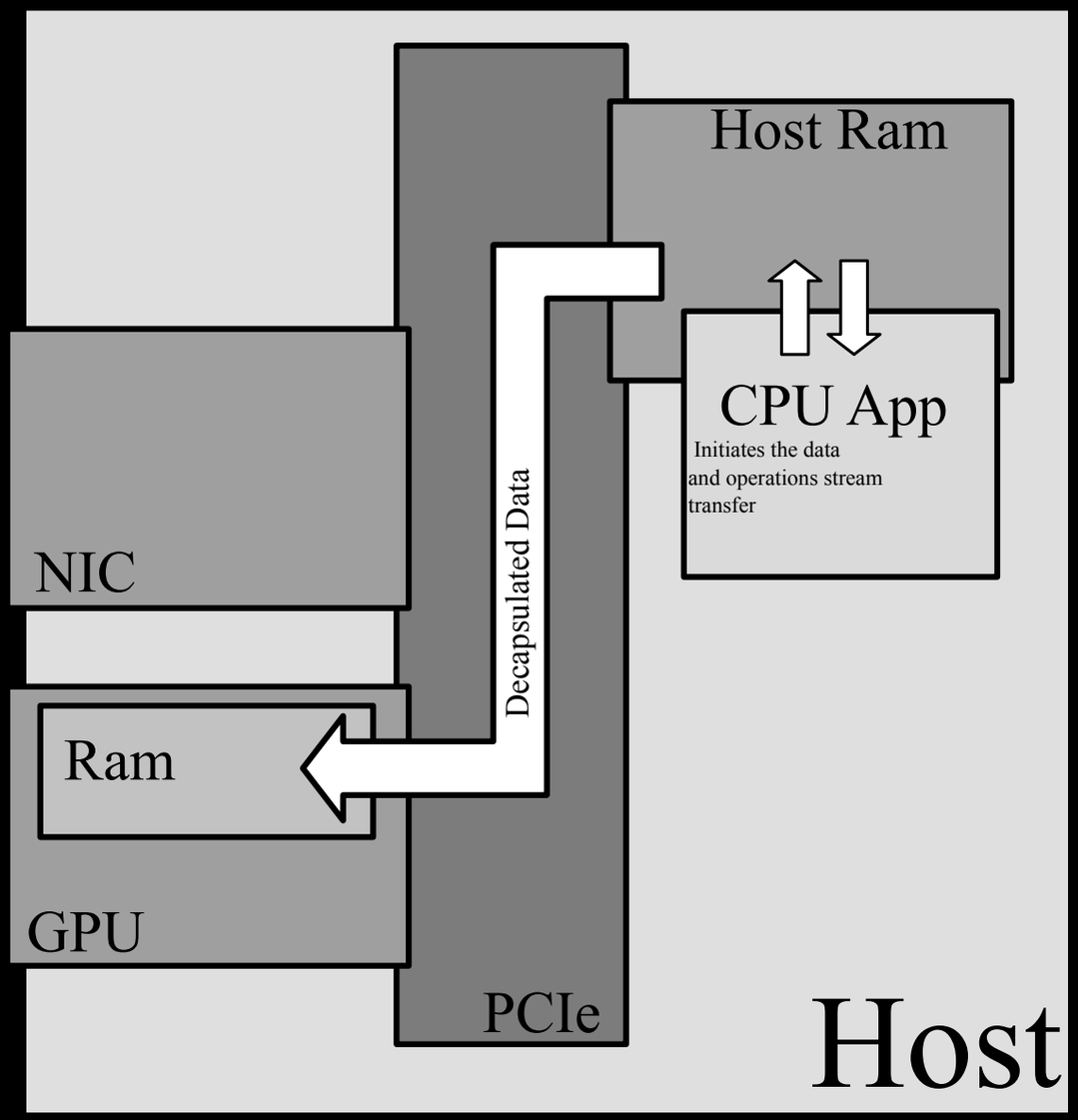
No PCIe P2P at all.



- The NIC DMA engine sends data to host memory and sends an interrupt to the CPU when the buffer is (half)full.
- The CPU suspends its tasks and saves its current state (context switch).
- The CPU decapsulates the data (Ethernet,TCP/UDP, GigeVision…).

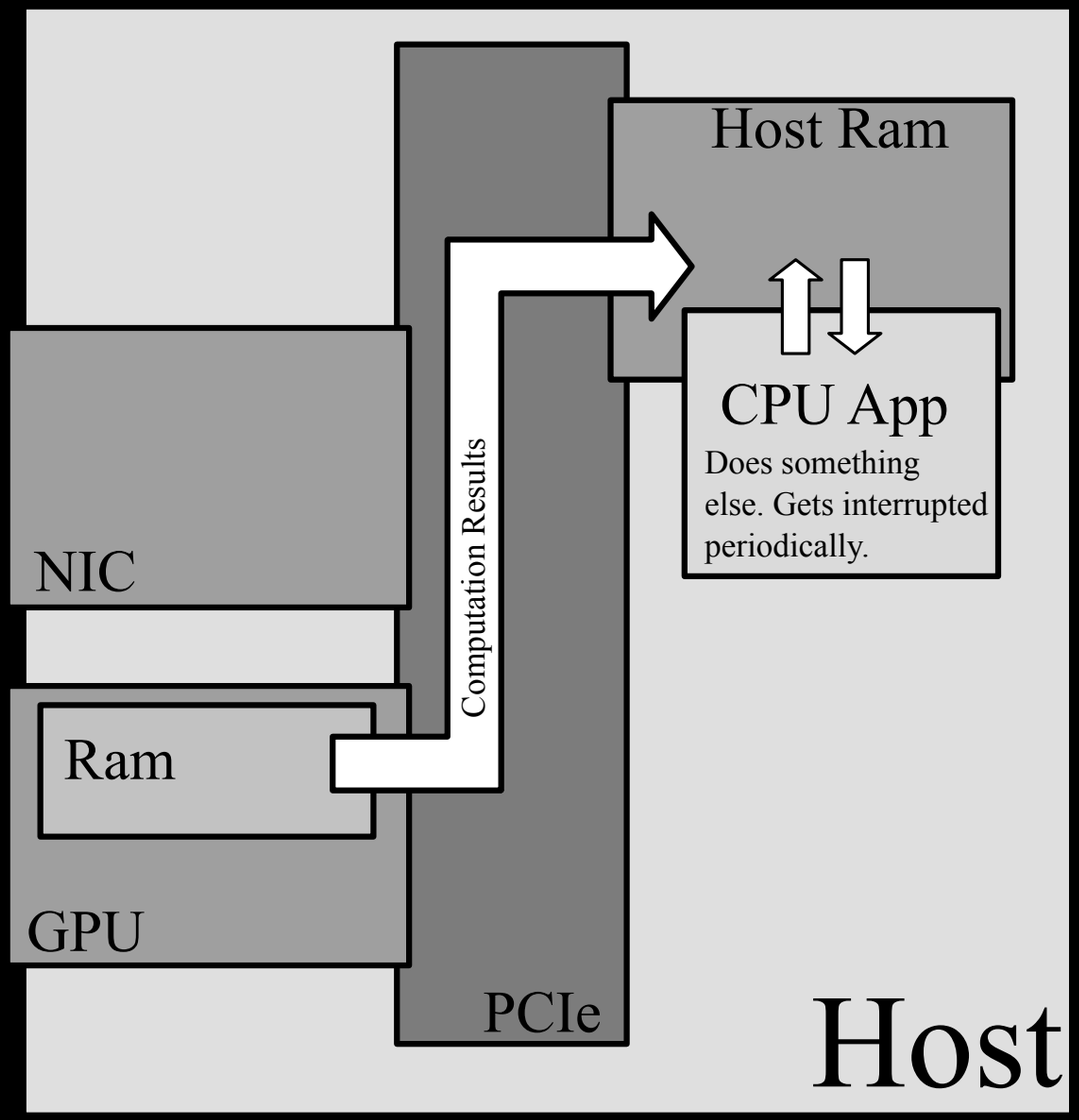
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

No PCIe P2P at all.



- The CPU builds a CUDA stream containing GPU operations as kernel launch and memory transfer orders.
- The stream is sent to the GPU
- The pixels are copied on the GPU RAM (GPU DMA, reading process over PCIe)

No PCIe P2P at all.

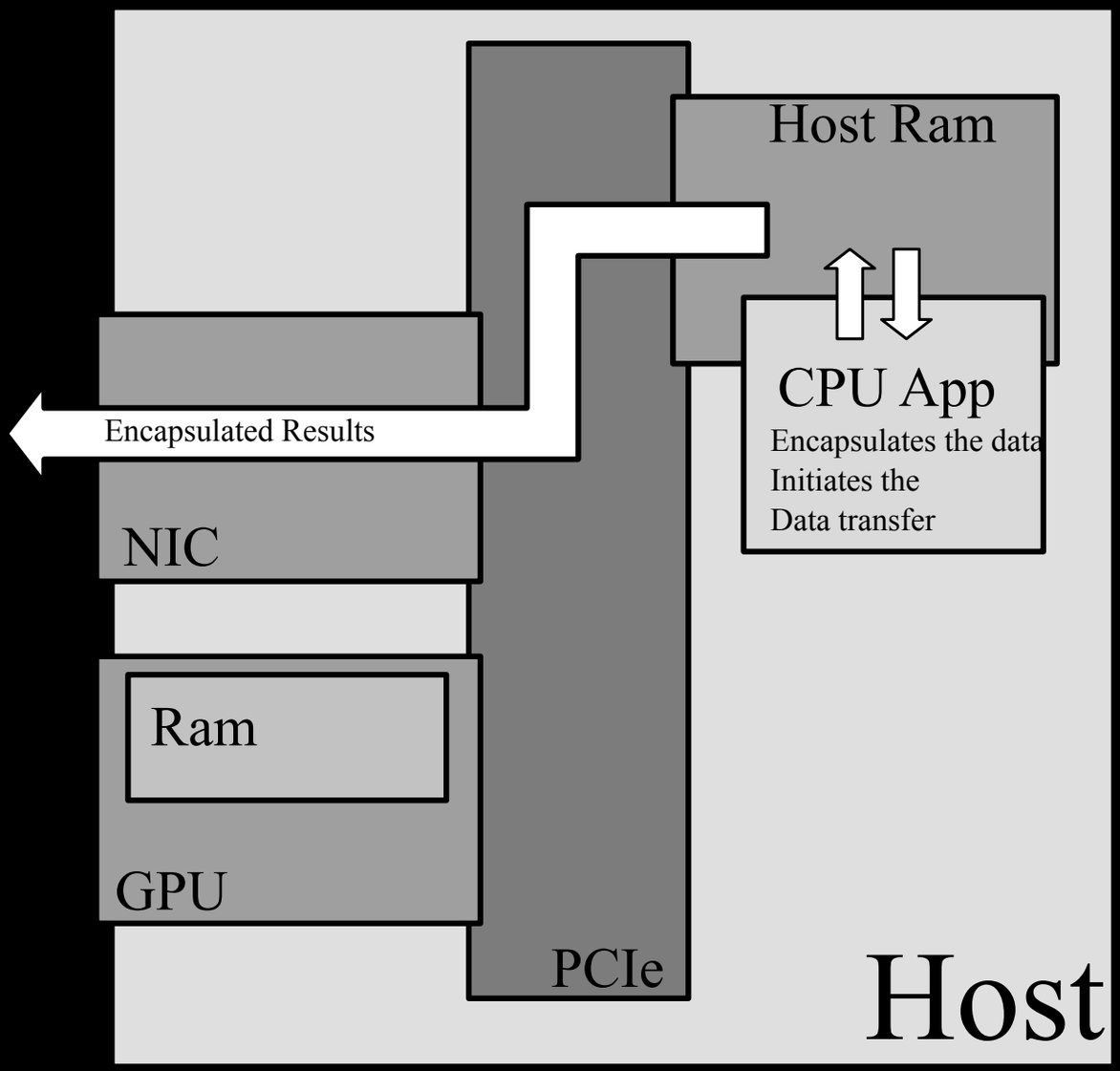


- The CPU waits for the computation to end (or does something else)
- The results are sent to the CPU RAM.
- The synchronization mechanisms between the GPU and the CPU are hidden (interrupts…).

l'Observatoire de Paris | LESIA

Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

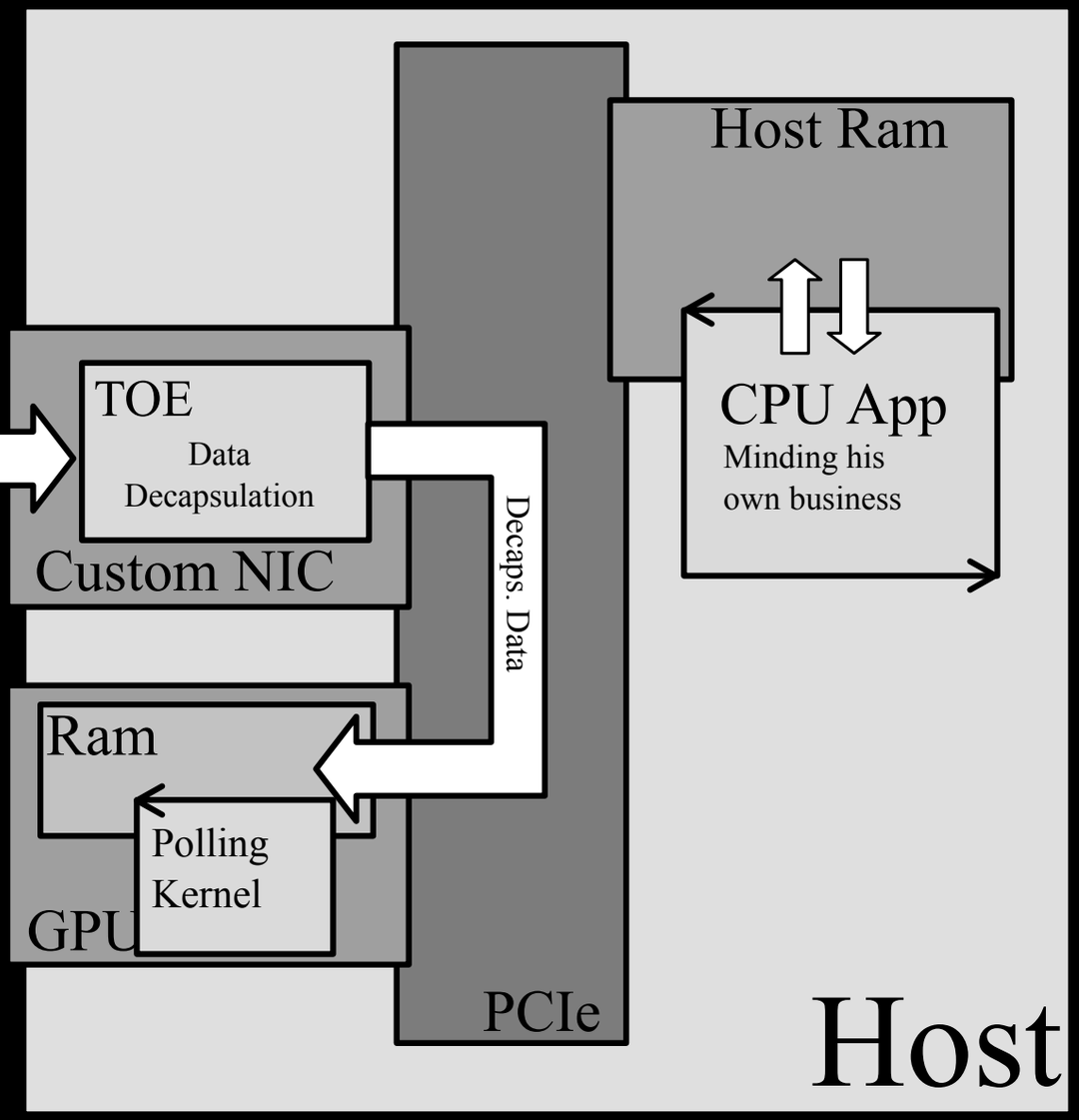
No PCIe P2P at all.



- The CPU encapsulates the results (Ethernet, TCP/UDP, …)
- The CPU initiates the transfer by configuring and launching the NIC DMA engine (reading process).

Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

With PCIe P2P and Custom NIC

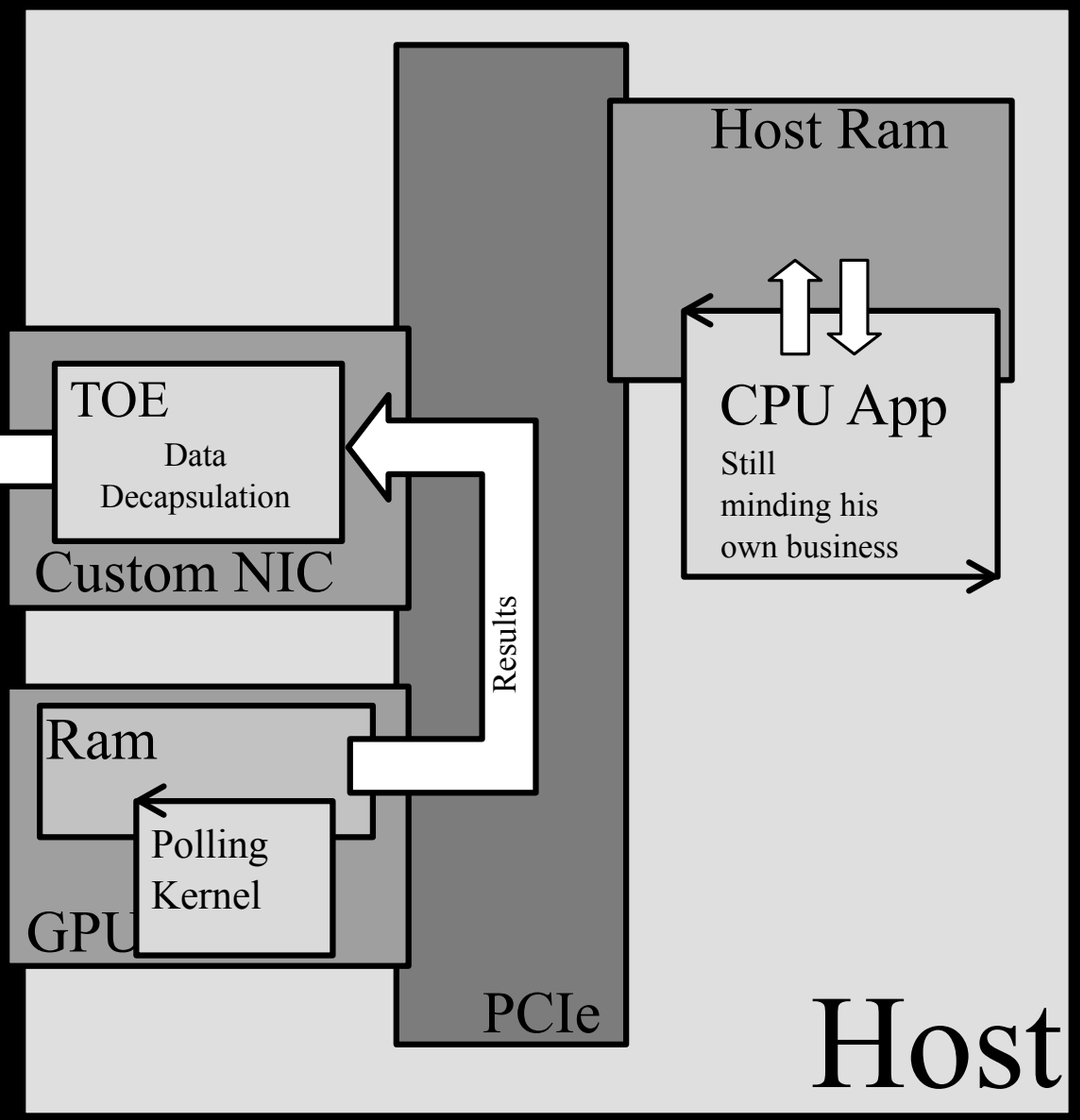


~ The CPU gets the address and size of a dedicated GPU buffer and use it to configure the NIC DMA engine.

~ The data are written directly to the GPU memory.

~ The GPU infinitely detects new data by polling its memory (busy loop), performs the computation and fills a local buffer with the results.

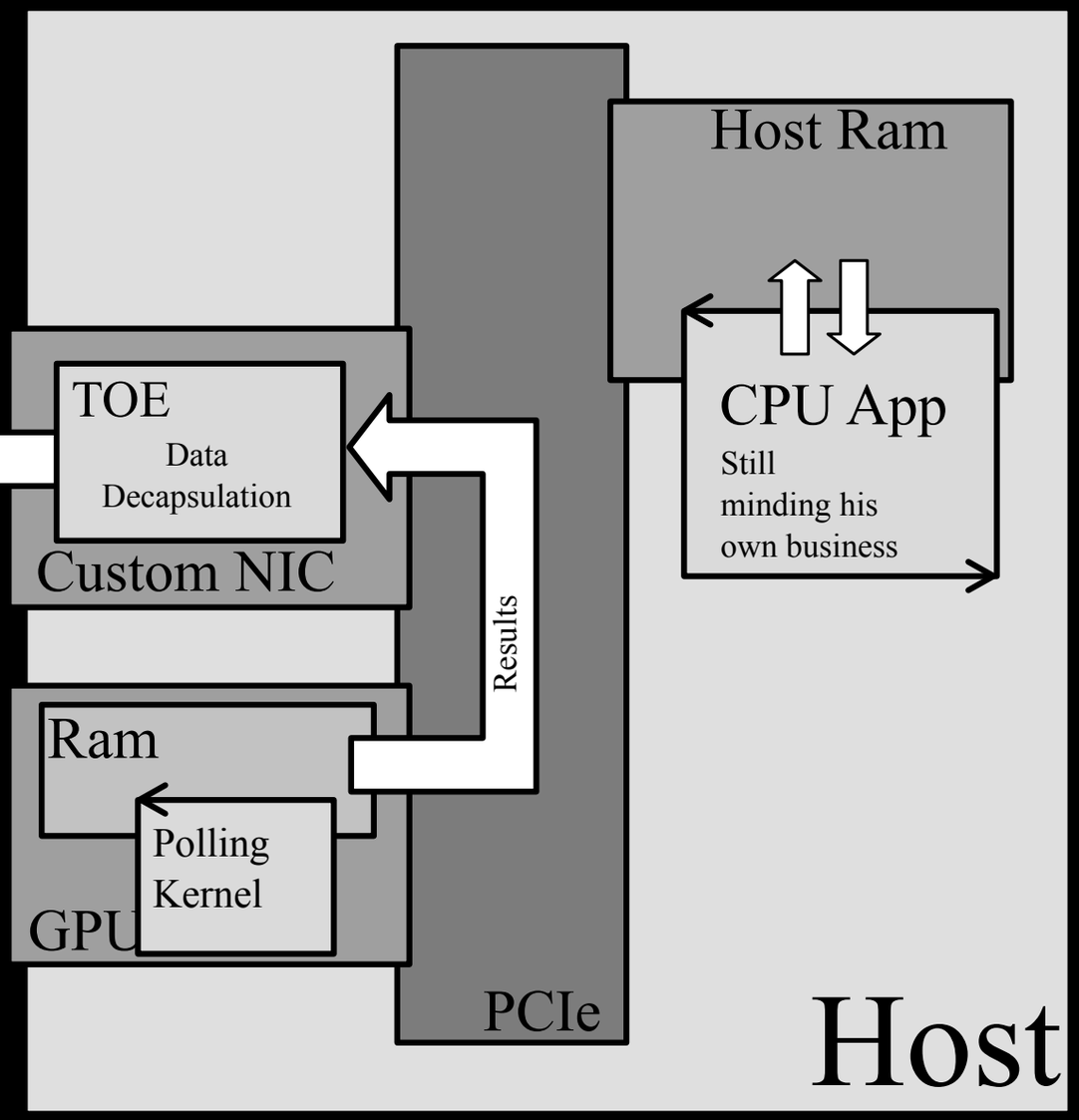
Getting back the results: 1rst way



- The CPU gets notified in a hidden way that the computation.
- The CPU initiates the transfer from GPU to FPGA (FPGA DMA engine -> reading process over PCIe).

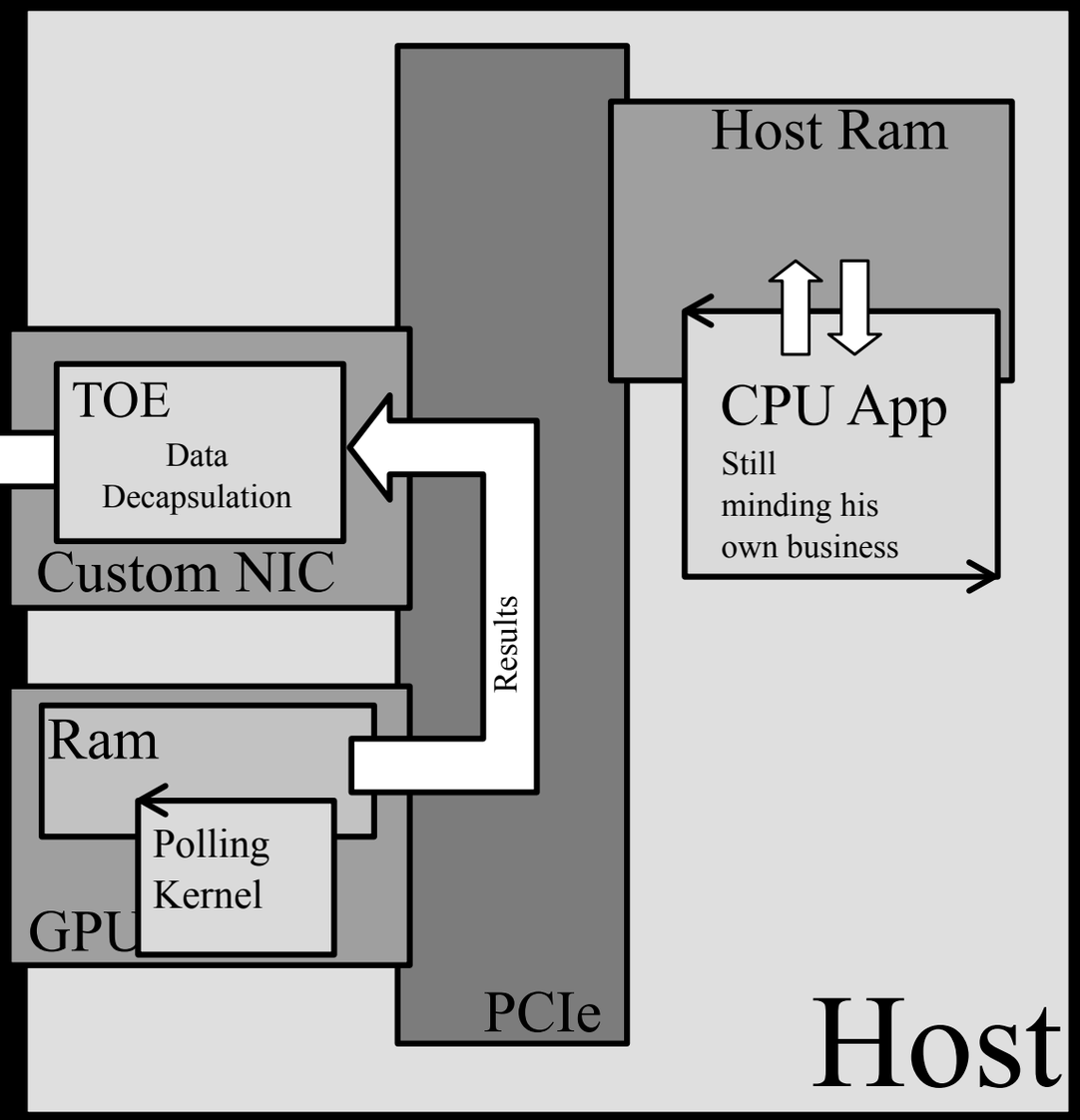
l'Observatoire de Paris | LESIA
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

Getting back the results: 2nd way



~ The GPU sends directly the data by writing to the NIC addressable space (GPU DMA -> write process over PCIe).

Getting back the results: 3rd way



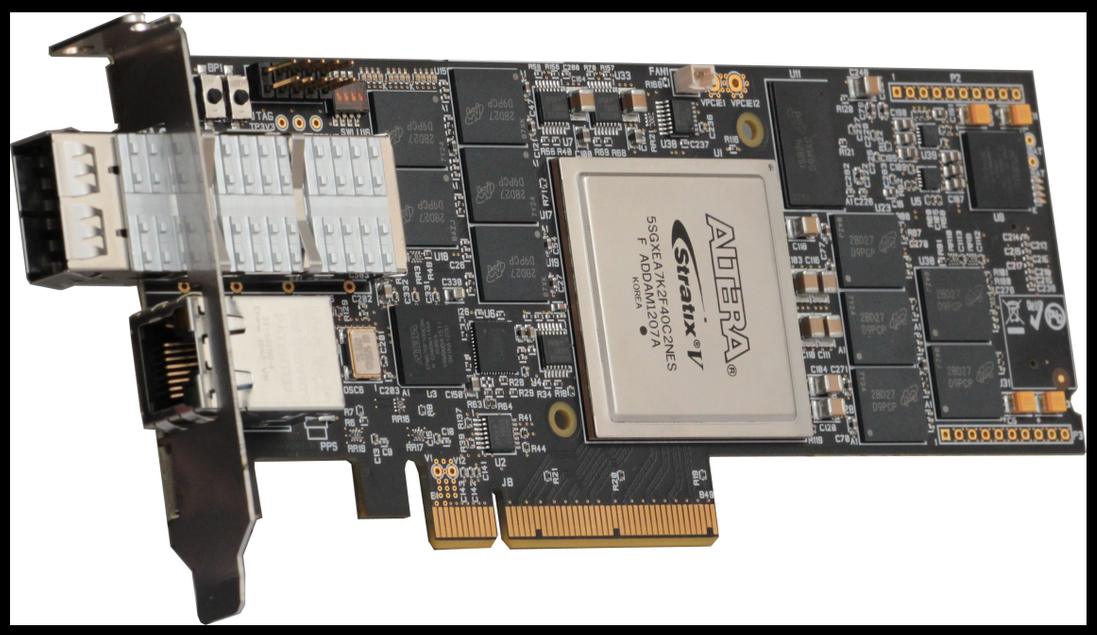
- The GPU notifies the FPGA by writing a flag on the FPGA addressable space.
- The FPGA gets the data back (FPGA DMA engine -> reading process).

Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

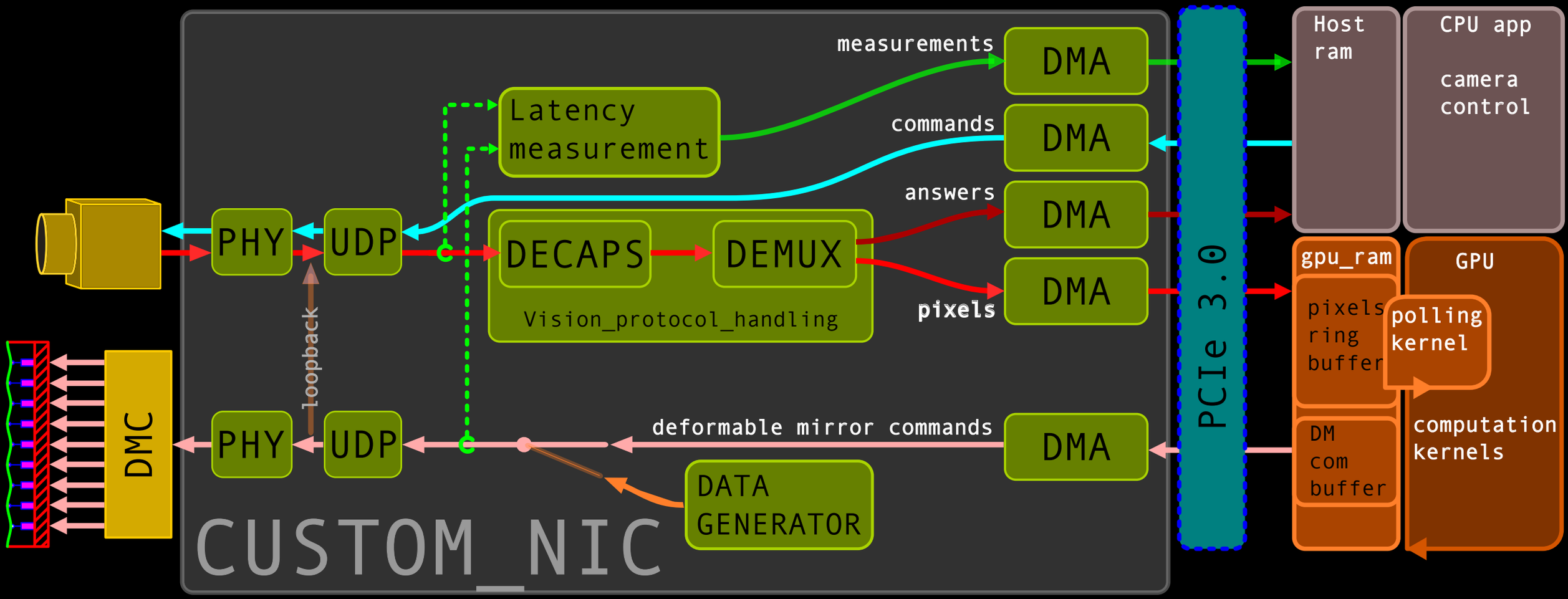
Using FPGA to reduce acquisition latency: First tests

~ Stratix V PCIe development board from PLDA (+ QuickPCIe, QuickUDP IP cores)
  42 Gb/s demonstrated from board to GPU; 8.8 Gb/s per 10GbE link in loopback mode

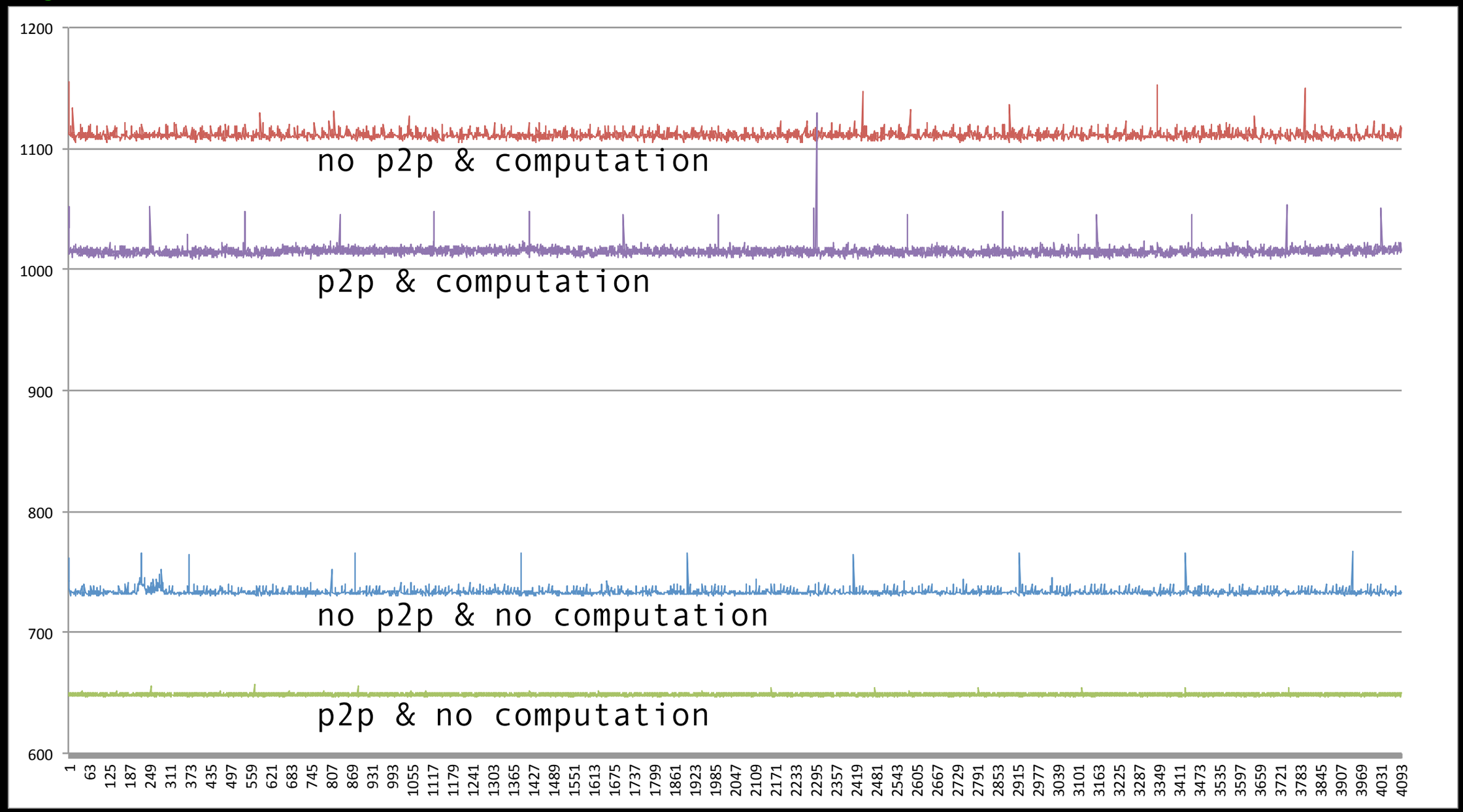
~ 10 GbE camera from Emergent Vision Technologies (8.9 Gb/s to GPU mem), GigEVision protocol, 1.5 kFPS in 240x240 pixels coded on 10bits
  (360 FPS in 2k x 1k on 8bits or 1k x 1k on 10bits)



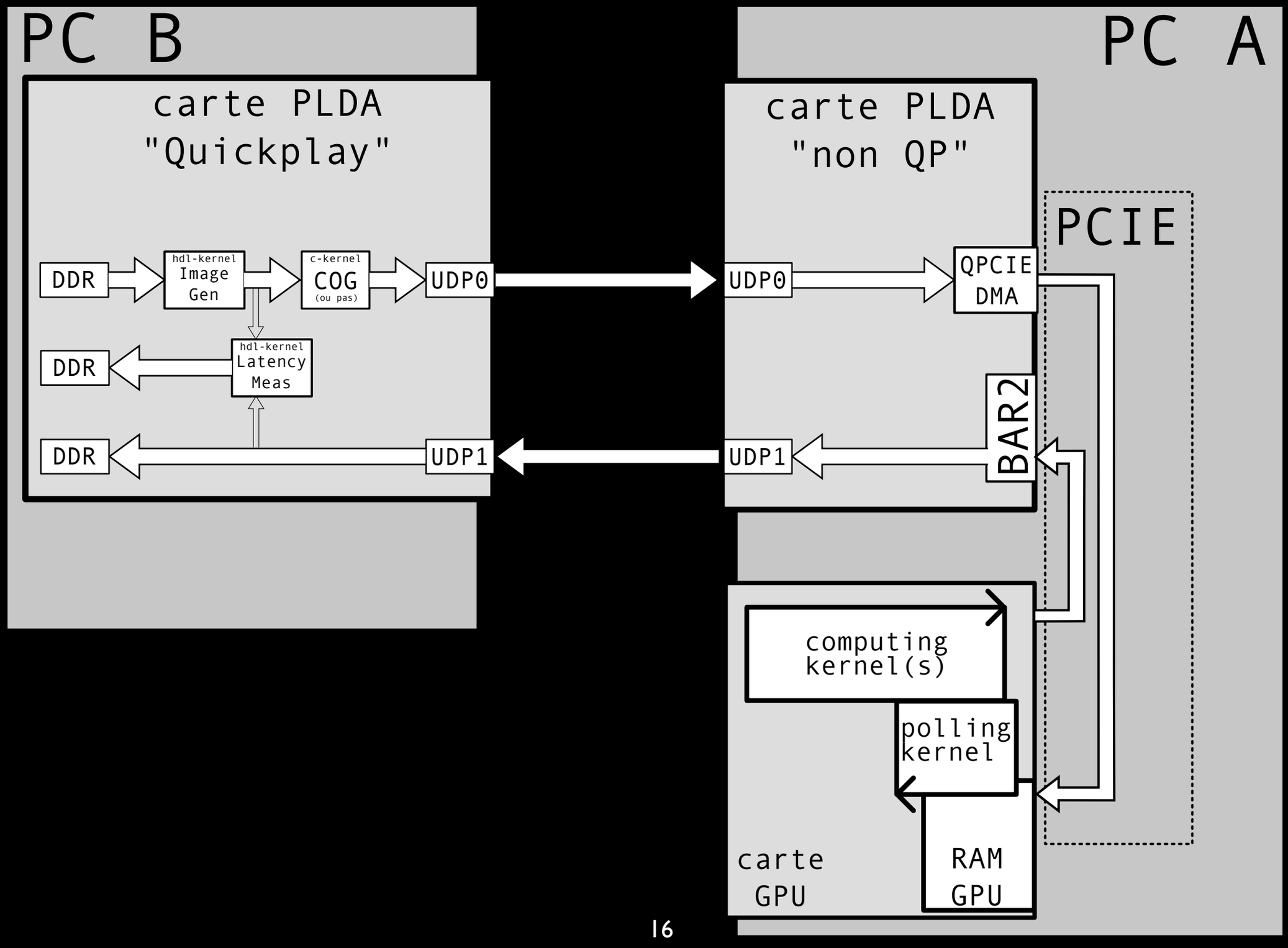
Using FPGA to reduce acquisition latency



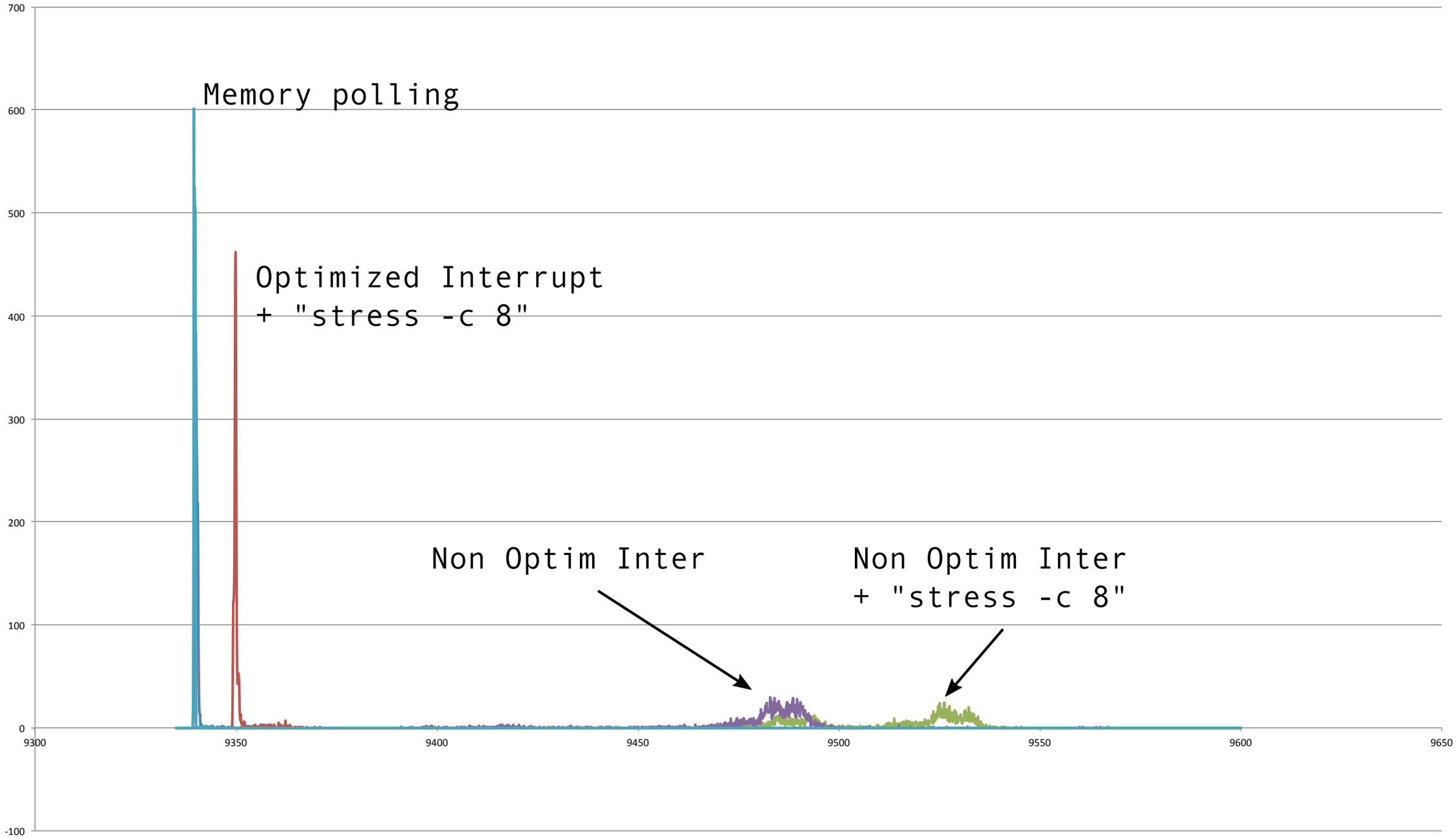
P2P from FPGA to GPU (way back still launched by the CPU)



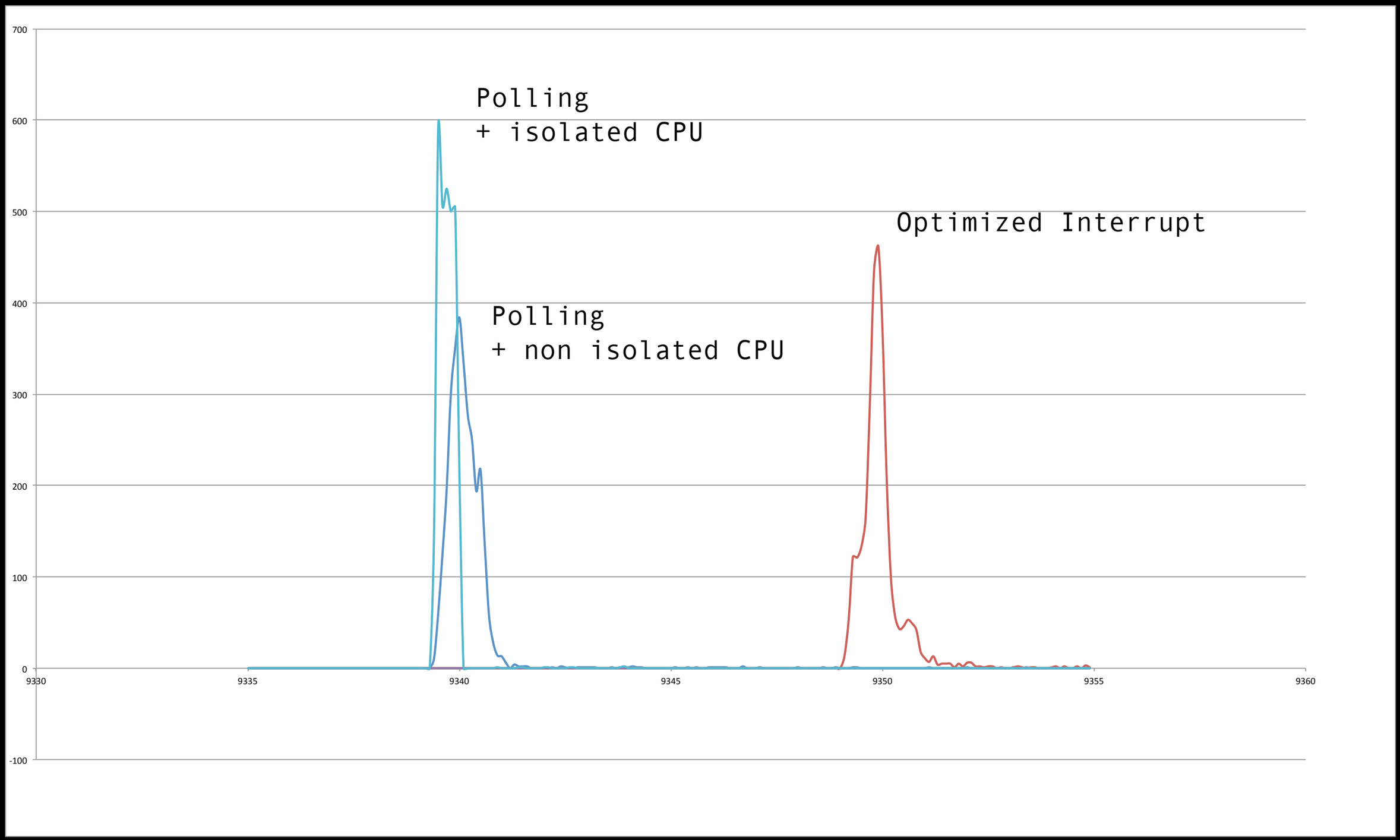
Interrupts vs polling (on the cpu)



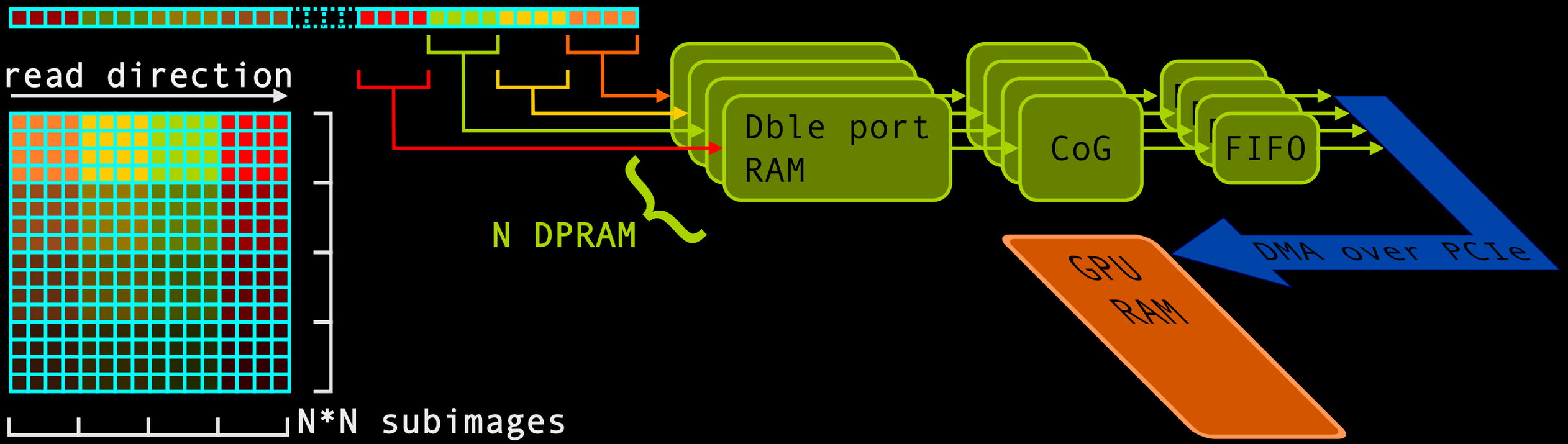
Interrupts vs polling (on the cpu)



Interrupts vs polling (on the cpu)



Using FPGA to reduce the load on GPU/PCIe/network…



- ~ FPGA are well suited for on-the-fly decapsulation, data rearrangement and highly parallelized computation.
- ~ Less load on the PCIe, smaller buffers on the GPU ram, less latency
- ~ Could be done in the WFS, so we lower the load on the network

…or even do everything with FPGAs

- Exploring the possibilities with high level development environments:
  - Vendor specific HLS. Xilinx Vivado,…
  - Quickplay: based on KPN. Has its own HLS. Is gonna be able to use other HLS. They are planning to integrate PCIe P2P.
  - OpenCL: is available for FPGAs (Altera), CPUs, GPUs. Can now handle network protocols: Altera introduced I/O channels allowing kernels read and write network streams that are defined by the board designer. P2P over PCIe should be possible (synchronization…?). Integration of custom HDL blocks?
  - Matlab to HDL. Did someone try it? Maybe useful to help developing IPs.

-> Interactions with HPC tools as MPI, DDS, Corba are quite challenging