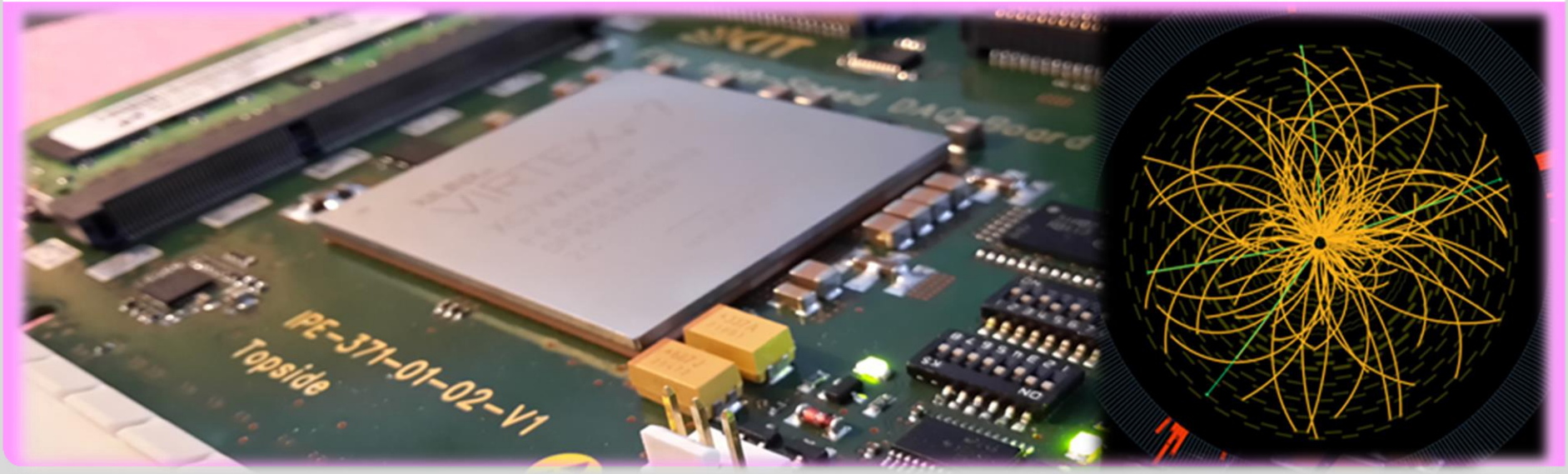
Karlsruher Institut für Technologie

High-speed, low-latency readout system with real-time trigger based on GPUs

IEEE- 20th Real Time Conference, 5-10 June 2016. Padova, Italy

M. Caselle, L.E. Ardila, M. Balzer, S. Chilingaryan, T. Dritschler, A. Kopmann, H. Mohr, L. Rota, M. Vogelgesang, M. Weber

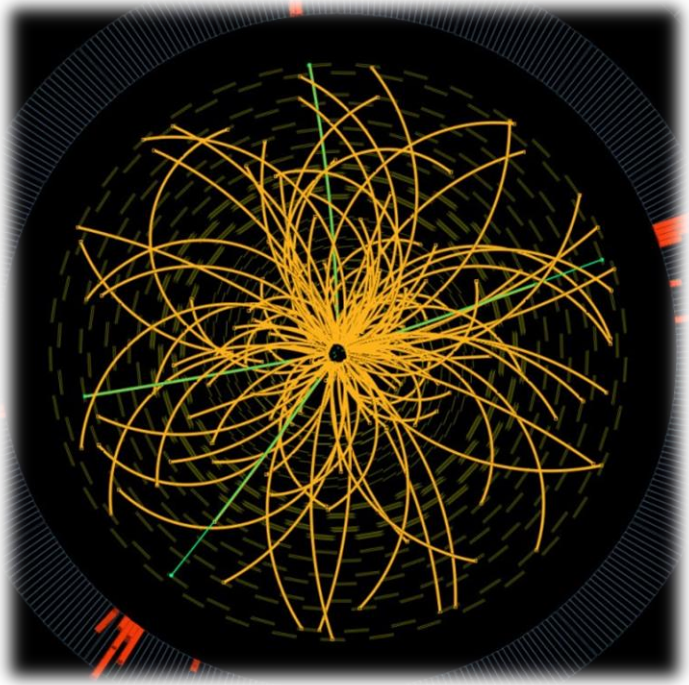
KIT, Institut für Prozessdatenverarbeitung und Elektronik
M. Caselle



Outline

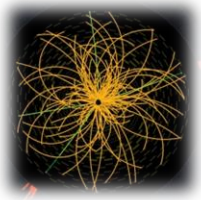
- Motivations
- Hardware implementation based on “Direct-GPU” technology
- Performance: Bandwidth & Latency
- Track finding algorithm based on GPU
- Results and GPU limitations
- Conclusions & what’s next

See: Thomas Schuh,
this conference ID: RTA1_59



L1 trigger will require **reconstruction of charged particles**
with transverse momentum > ~2 GeV/c

CMS – L1 track trigger system



Processed off-module in the back-end

Data rates:
O(100 Tb/s)

L1 track finding system
Track reconstruction

Global Trigger → L1 accept to front-end

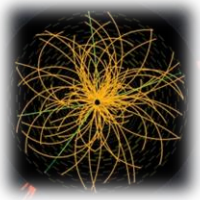
tracks from muons system and calorimeter information

Track reconstruction and fitting,
primary particles with $p_T > 2$ GeV. Latency: ~ **5 µs**

How to find the tracks in ~5 µs with high efficiency and acceptable fake rates?

	Patter recognition	*Fitting*	*Slices*
Associative Memory	Large bank of pattern stored in a dedicated **AM** chip	PCA, Hough transform, Retina (**FPGA**)	48 = 8x6 (φxη) Loading balancing time
Tracklet algorithm	conventional road-based track search (**FPGA**)	linearized χ^2 fit (**FPGA**)	168 = 28x6 (φxη) 4 x (BX)
Time-multiplexed architecture	Hough transform (**FPGA**) → ***See: Thomas Schuh***	**(FPGA**)	324= 36x9 (φxη) and time multiplexing of 24 x (BX)

CMS – L1 track trigger system



Processed off-module in the back-end

Data rates:
O(100 Tb/s)

L1 track finding system
Track reconstruction

Global Trigger → L1 accept to front-end

tracks from muons system and calorimeter information

Track reconstruction and fitting,
primary particles with $p_T > 2$ GeV. Latency: ~ **5 µs**

How to find the tracks in ~5 µs with high efficiency and acceptable fake rates?

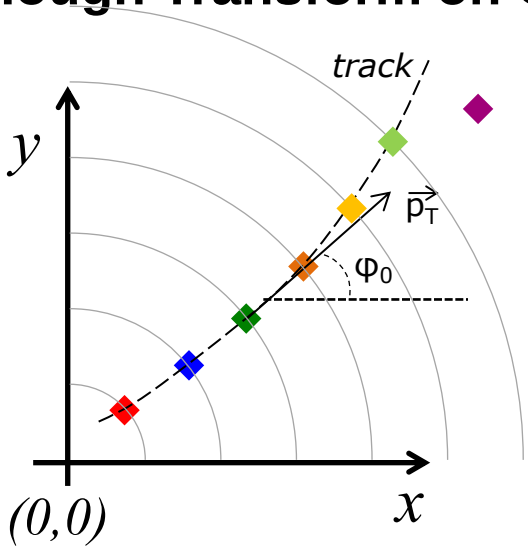
	Patter recognition	*Fitting*	
Associative Memory	Large bank of patter store		(φxη) Loading balancing time
Trackle	search (**FPGA**)	linearized χ^2 fit (**FPGA**)	168 = 28x6 (φxη) 4 x (BX)
Time-Multiplexed architecture	Hough transform (**FPGA**)	(**FPGA**)	324= 36x9 (φxη) and time multiplexing of 24 x (BX)

HW architecture with several **thousands** of associative memories (**ASICs**) and **thousands** of **FPGAs**

What about GPUs for L1 track finding ?

How performs a GPU L1 track system compared to current HW systems (AMs + FPGAs) ?

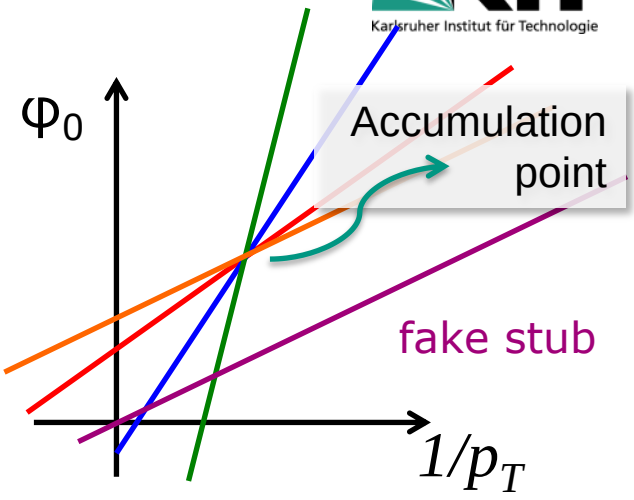
Hough Transform on GPU



$$\varphi_0 = \varphi - \frac{0.57\ q}{p_T} * \frac{GeV}{m} * r$$

Each stubs → corresponding line in the Hough Space
All stubs from a real track have same (p_T , production angle)

→ *See: Thomas Schuh*



Accumulation points with high vote will correspond to **real tracks**

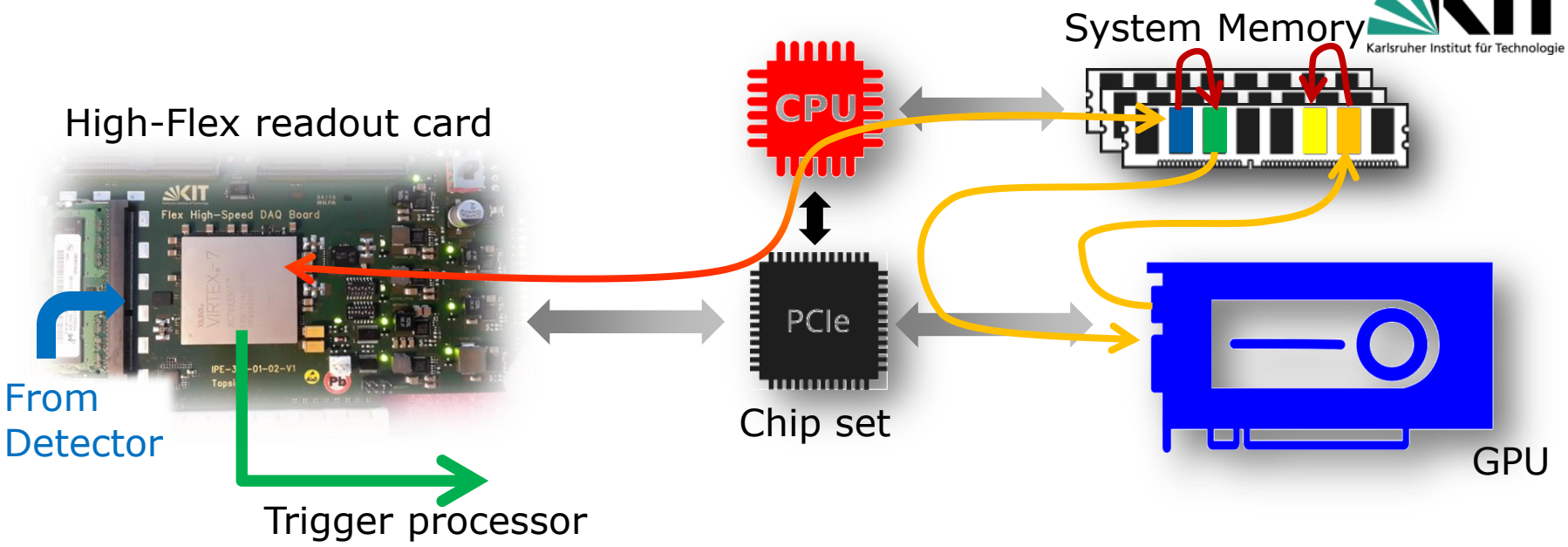
Why Hough transform on GPU?

Highly parallel computing → optimized to execute simultaneously the same operation on many different data (Single-Instruction on Multiple Data)

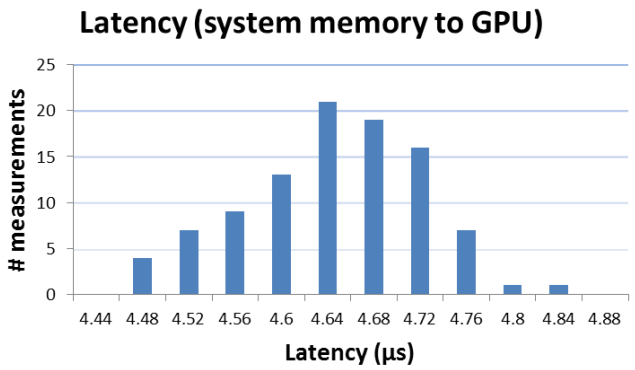
The Hough transform is naturally amenable to a high degree of parallelization, as the parameter space calculation for each hit (stub) is independent of all other hits (Stub) in the event/tracks.

→ Consequently, it is a natural candidate for implementation on a GPU

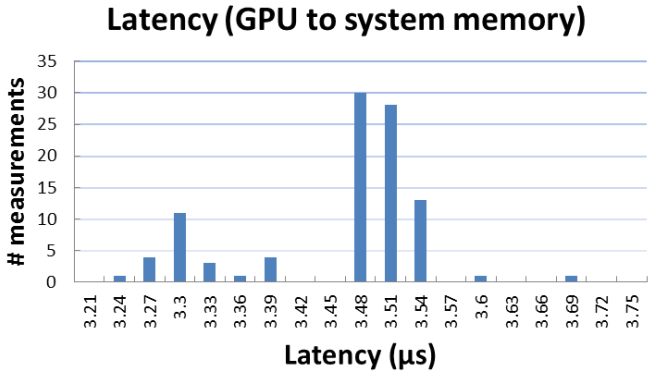
Data transfer: Detector – GPU



Total latency = latency (FPGA <-> Syst. Mem.) + latency (syst. Mem. <-> GPU) + latency (GPU process)



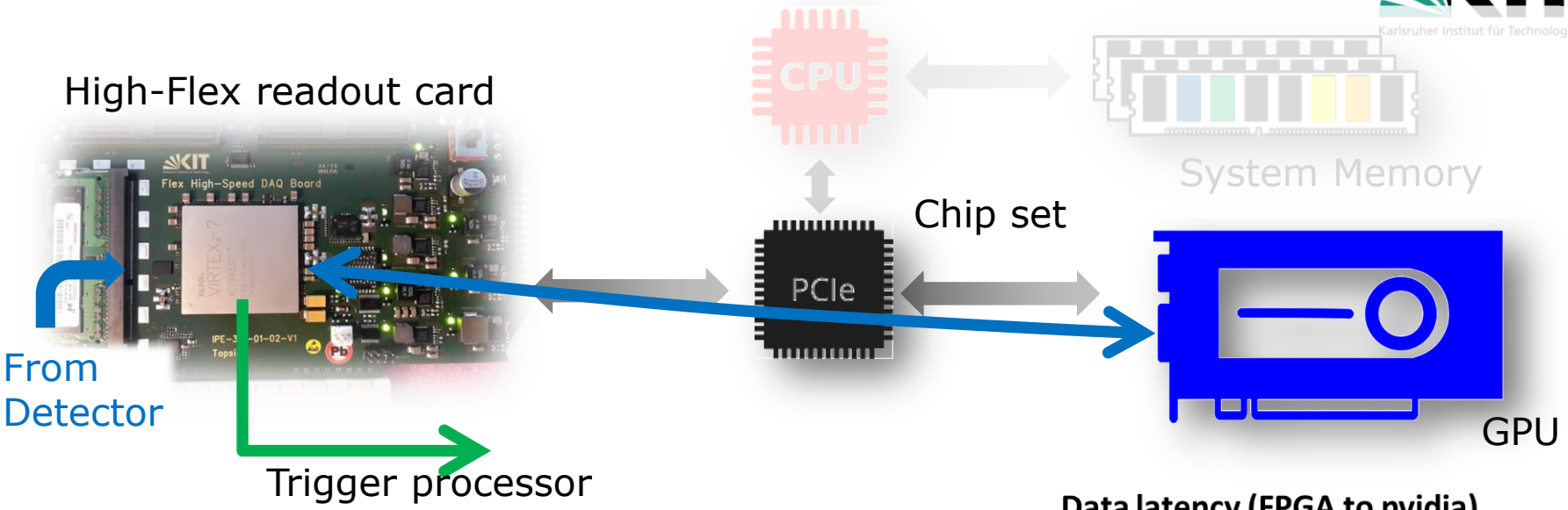
~ 4.6 µs for 4KBytes



~ 3.4 µs for 4KBytes

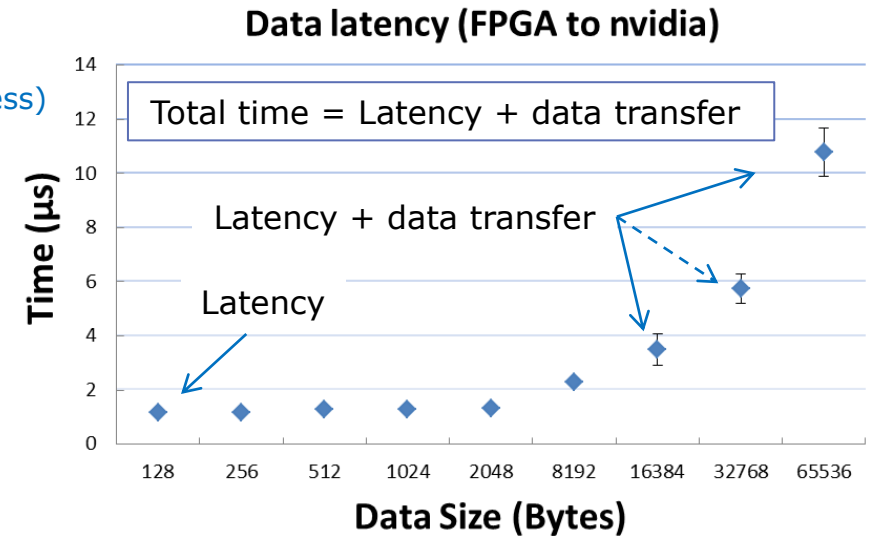
~ 8 µs *>> CMS low level trigger specification*

Advanced Data transfer with “GPU-Direct”



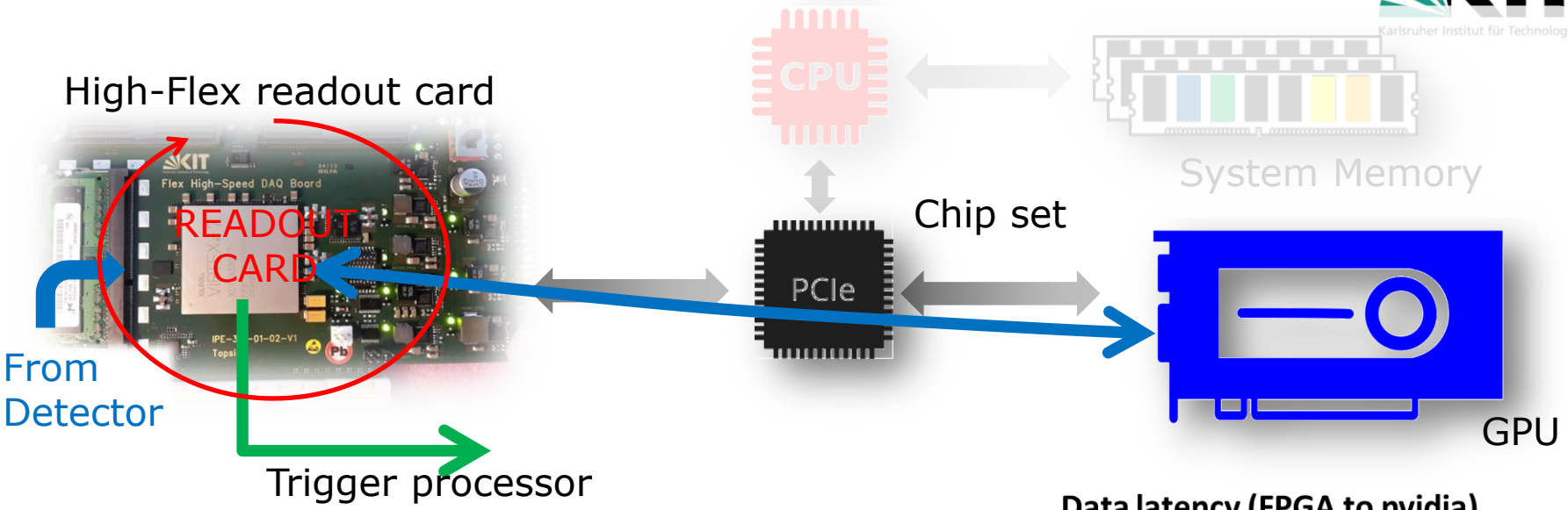
Total latency = latency (FPGA <->GPU) + latency (GPU process)

Using “Direct-GPU”, FPGA devices can read and write directly CUDA/OpenCL host and device memory, eliminating unnecessary memory copies, dramatically lowering CPU overhead and reducing latency



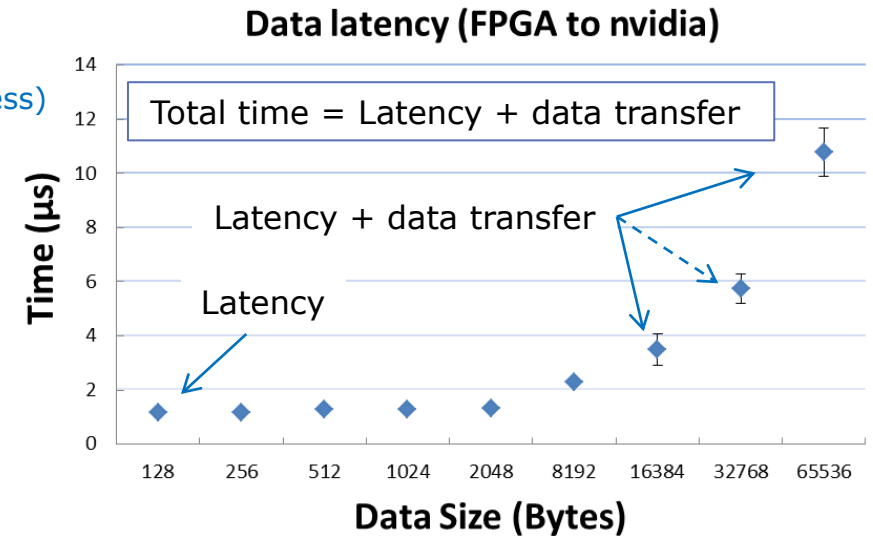
*One-sided data transfer latency **1.15 µs** (average), jitter < **100 ns***

“GPUDirect” and “DirectGMA” concepts



Total latency = latency (FPGA <->GPU) + latency (GPU process)

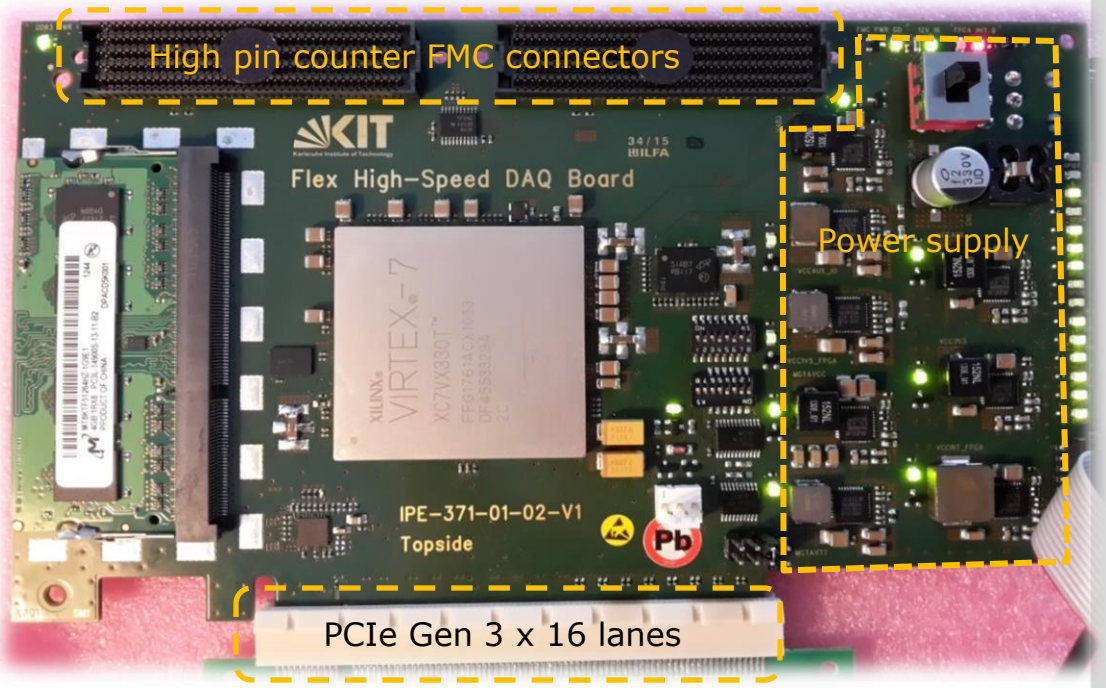
Using “Direct-GPU”, FPGA devices can read and write directly CUDA/OpenCL host and device memory, eliminating unnecessary memory copies, dramatically lowering CPU overhead and reducing latency



*One-sided data transfer latency **1.15 µs** (average), jitter < **100 ns***

High-flexibility readout card

- **Processing unit**
  - Xilinx Virtex 7 FPGA (XC7VX330T-2 FFG1761)
- **2 x High Pin Counter FMC connectors:**
  - VITA 57 compliant
  - 320 single-ended or 160 diff. signals @ 9 GHz
  - 12 MGT I/O @ 13.1 Gb/s
- **High performance Memory: DDR3**
  - 64 lanes @ 1866 Mb/s ➔ 119 Gb/s
  - 4 GByte
- **PCIe Gen 3 x 16 lanes**
- **PCBs:**
  - 16 layer metals stack / Nelco N4000-13 EP SI
  - Picosecond time controlled transmission lines

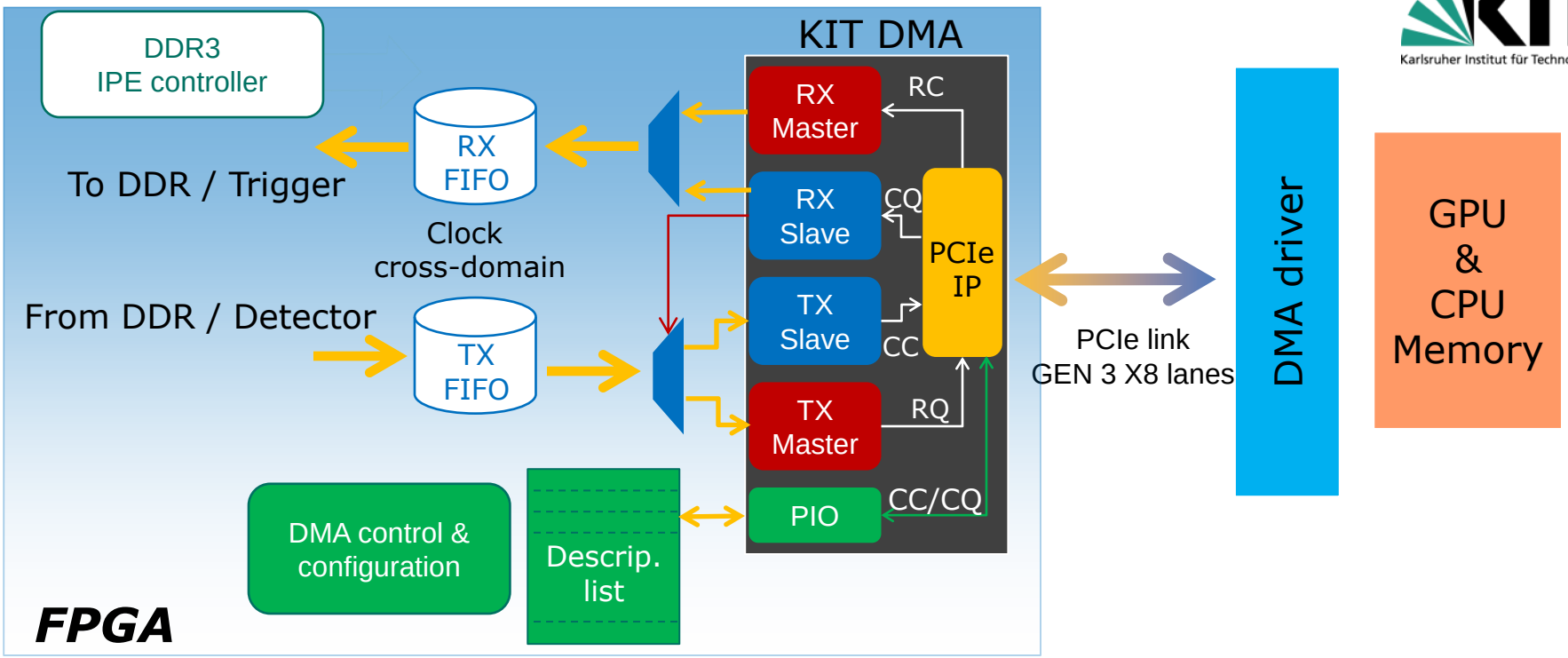


Data throughput up to 130 Gb/s full duplex

CPU/GPU DAQ

Ref: *A PCIe DMA Architecture for Multi-Gigabyte Per Second Data Transmission, DOI: 10.1109/TNS.2015.2426877, IEEE-Real time 2014 26-30 May. Nara Japan*

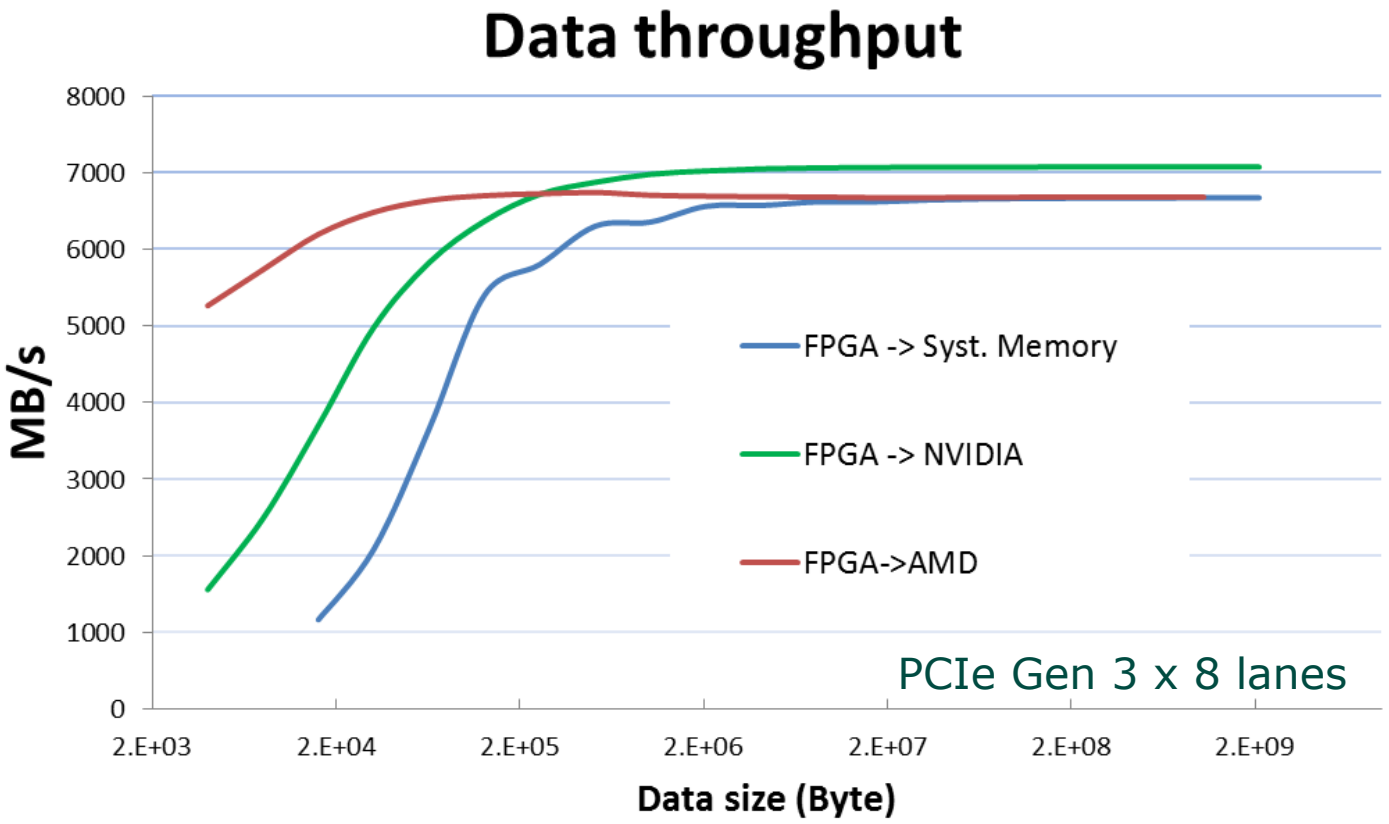
Firmware architecture



- ✓ KIT-Direct Memory Access → operating both **Bus Master/Slave** modes
- ✓ "**Scatter-Gather mechanism**" where descriptors located inside FPGA in both ring-buffer or memory dynamical allocation are possible.
- ✓ Compatible with (NVIDIA, AMD) **GPUs** and **system memory**
- ✓ PCI Express/DMA Linux 32-64 bits driver → **READY**

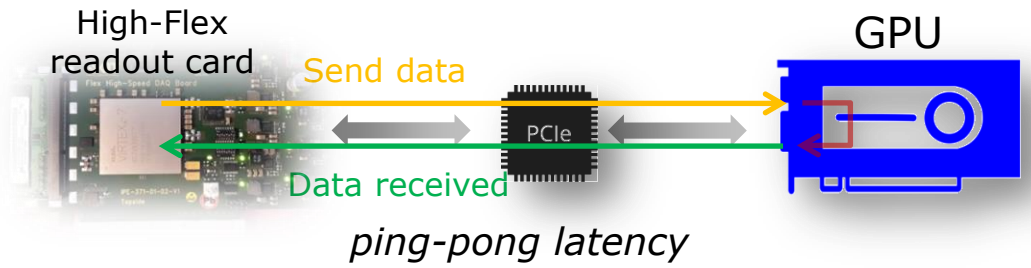
Readout system – performance / comparison

AMD → FirePro W9100 (OpenCL ver. 2.0) NVIDIA → Tesla K40 (CUDA ver. 7.5)



Data throughput over ***6.5 GB/s*** very close to maximum theoretical limit for PCIe Gen 3 (max payload limited to 128 Byte by GPUs)

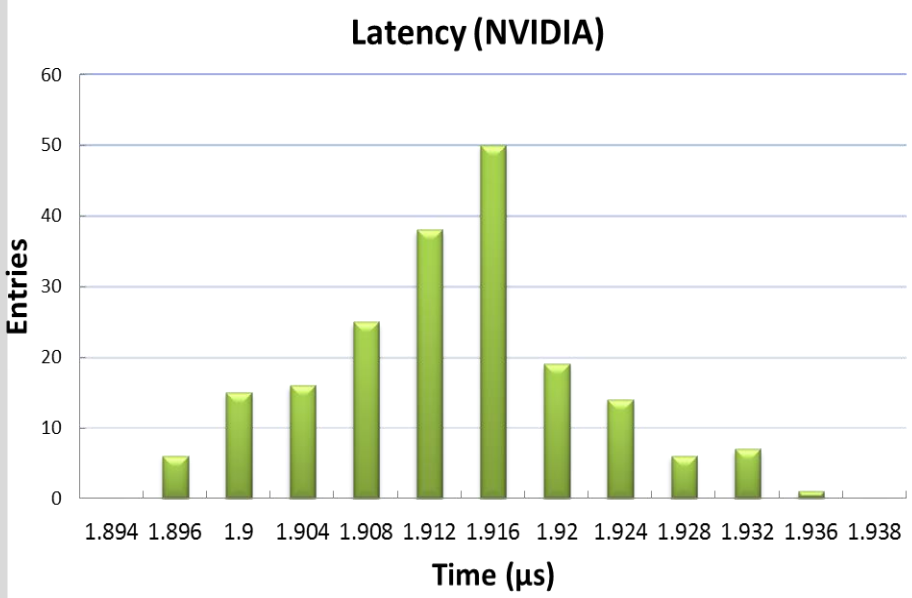
Readout system – performance / comparison

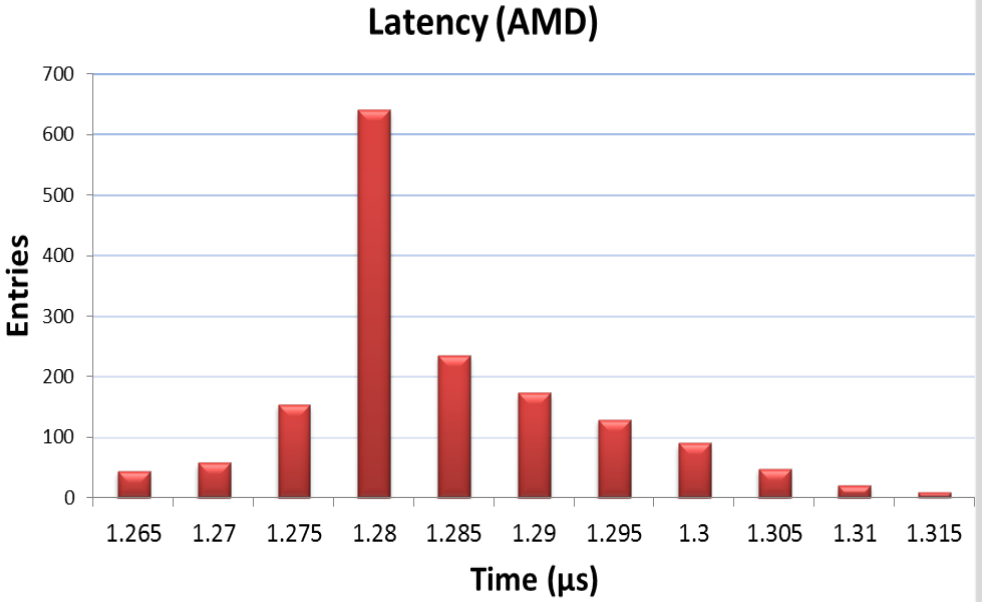


ping-pong latency

NVIDIA: FPGA as bus master (FPGA → GPU
GPU as bus master (GPU → FPGA)

AMD: FPGA as bus master (FPGA → GPU
FPGA as bus master (GPU → FPGA)





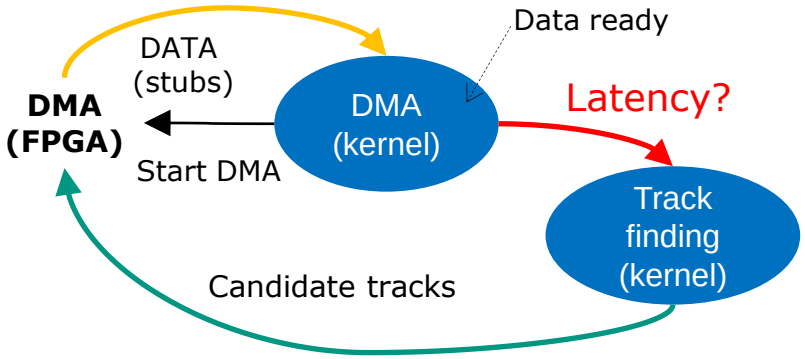
NVIDIA: Latency < 2 µs, jitter < 30 ns

AMD: Latency < 1.3 µs, jitter < 50ns
three PCIe transactions

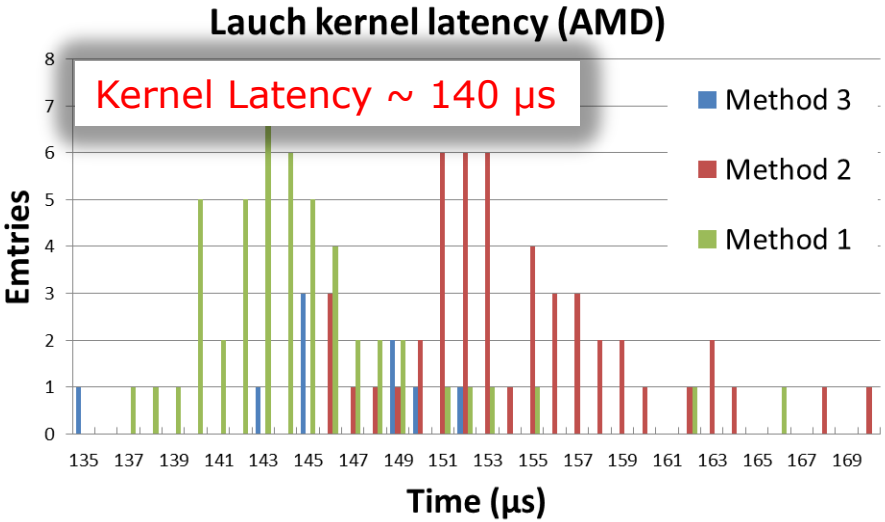
Both GPUs vendors present an excellent latency performance.

GPU limitations – kernel latency

How much time is necessary to synchronize the FPGA data with the launching of the kernel?



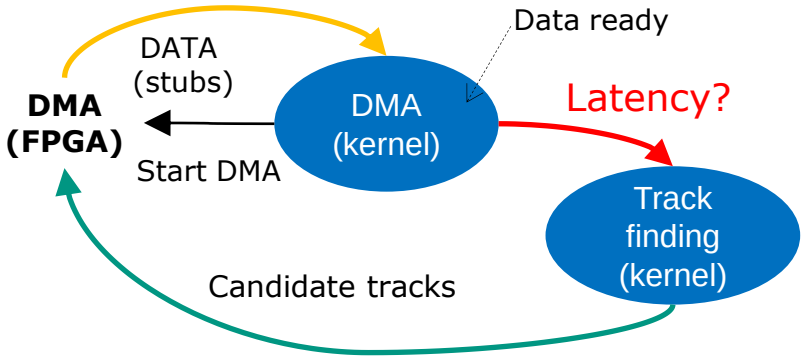
Launch data processing kernel (Hough transform)



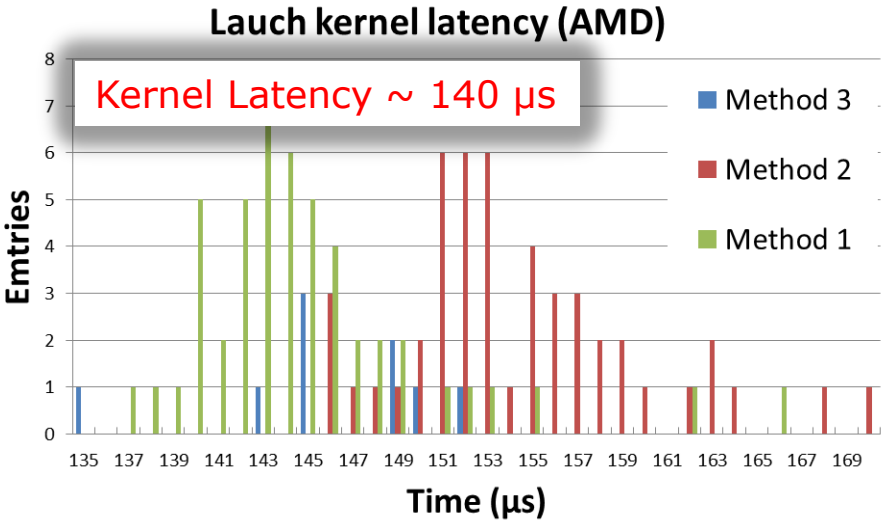
Expected GPU limitation for real-time application

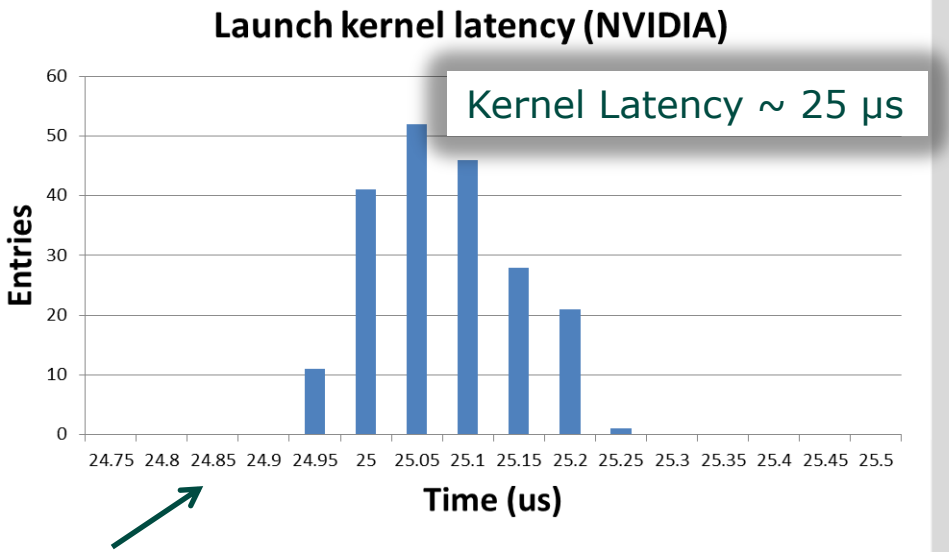
GPU limitations – kernel latency

How much time is necessary to synchronize the FPGA data with the launching of the kernel?



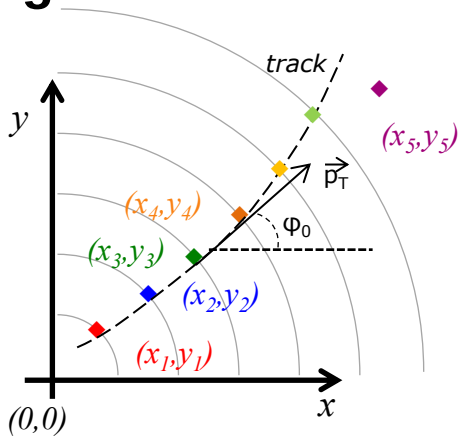
Launch data processing kernel (Hough transform)



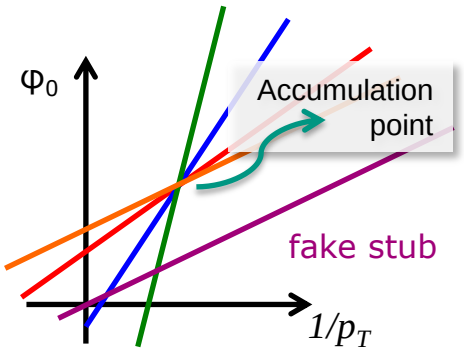


NVIDIA shows a very low kernel latency. Drastically reduction → expected new CUDA release (see: *state of GPUDirect technologies*, Davide Rossetti)

Hough Transform on GPU – preliminary results



Hough parameter space

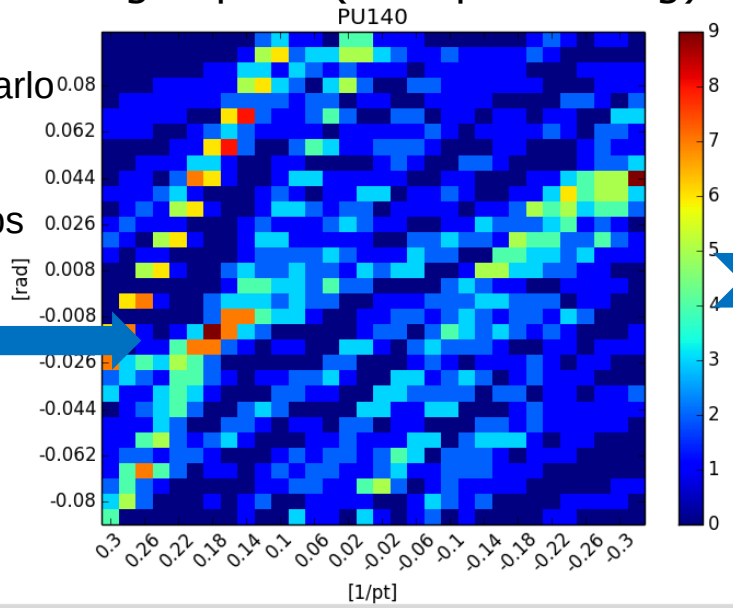


Tracker detector segmentation 288 sectors (32 in Φ and 9 in η) → like FPGA Hough transform implementation by *Thomas Schuh, this conference ID: RTA1_59*

→ Comparison between GPU and FPGA implementation using same algorithm and same data input

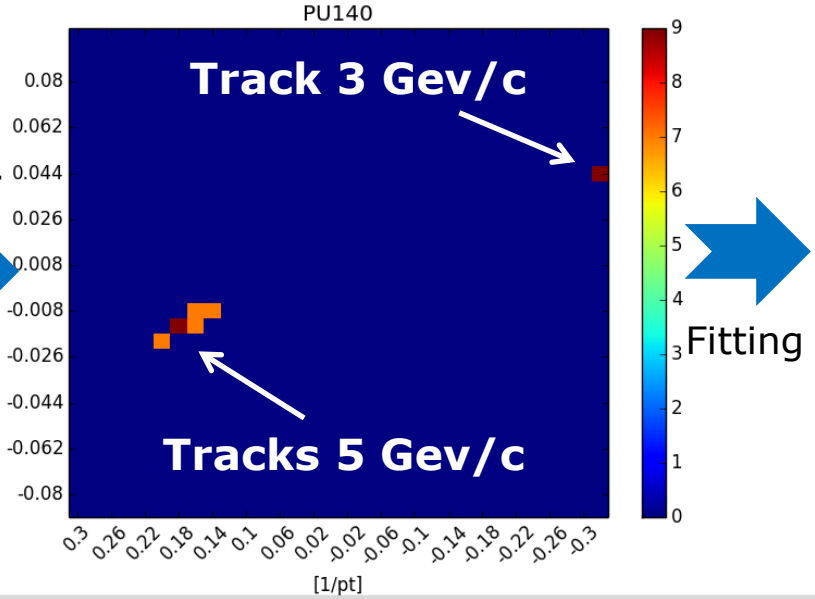
Hough space (GPU processing)

Monte Carlo (CMSW) PU:140 # 45 stubs



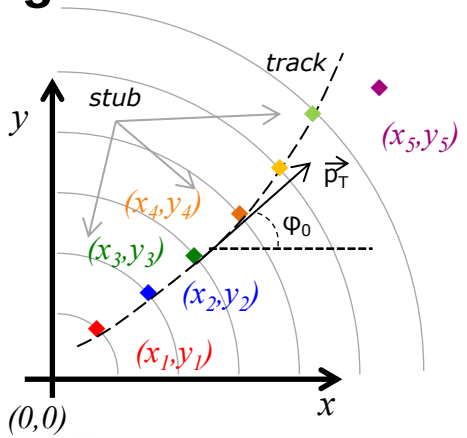
filter

Hough space (after filtering)

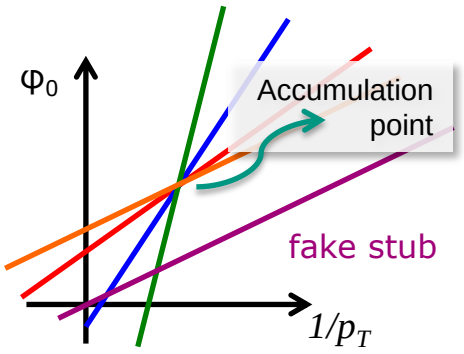


Fitting (η)

Hough Transform on GPU – preliminary results



Hough parameter space



Tracker detector segmentation 288 sectors (32 in Φ and 9 in η) → like FPGA Hough transform implementation by *Thomas S*

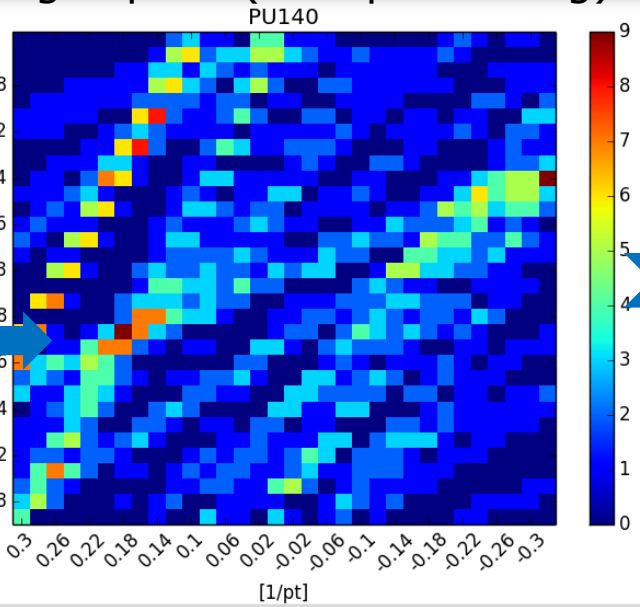
→ *Compa*

Current implementation in CUDA process **500** Stubs in **7 µs** (average time) on Tesla K40 GPU!

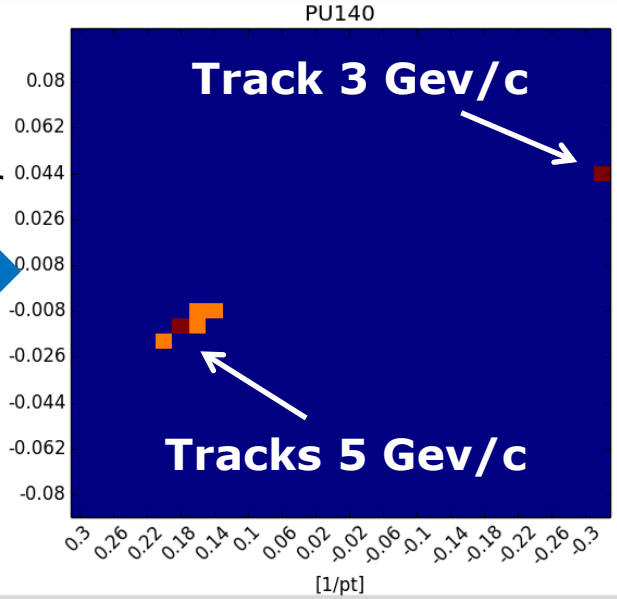
Hough space (GPU processing)

Hough space (after filtering)

Monte Carlo (CMSW) PU:140 # 45 stubs

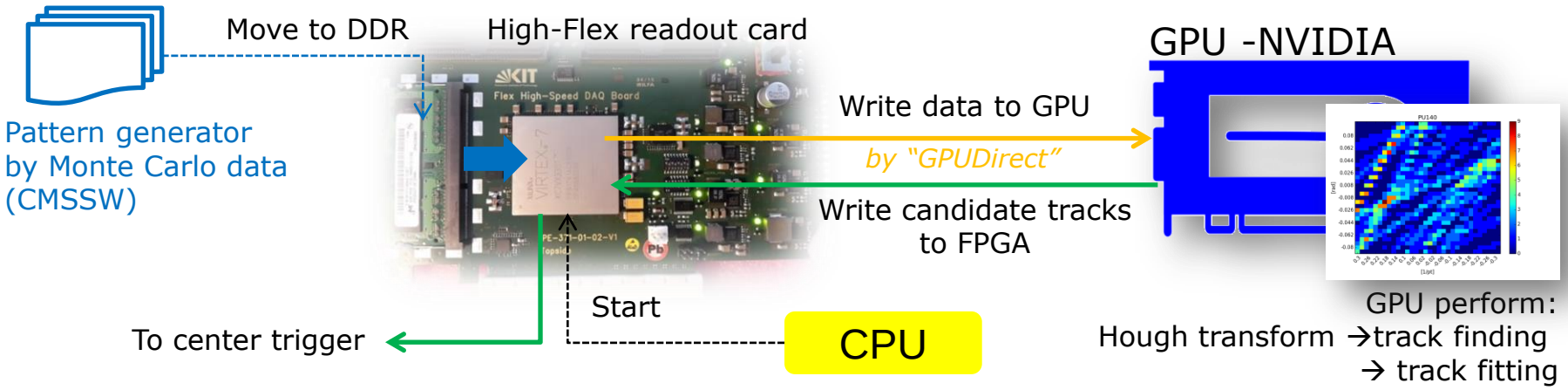


filter



Fitting (η)

pT track finding by GPUs – first demonstrator



1) Load the stubs to DDR (High-Flex)
2) Start data transfer FPGA -> GPU (start FPGA counter)
3) GPU launches the Hough transform → track finding
4) GPU sends candidate tracks to FPGA (stop FPGA counter)

→ Total time measured by a FPGA counter

Total latency > **150 µs** with AMD, the launching kernel is the major penalty

Total latency= 1.91 µs (data latency) + ~7 µs (data processing) + 23.19 (kernel latency) = **~32.1 µs by NVIDIA**

improvements → new CUDA 8

The total latency of ~ 30 µs is >> of 5 µs required by CMS → but very promising

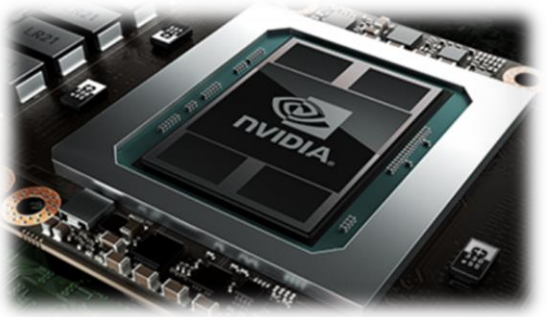
Conclusions & What's next

- ✓ First demonstrator of L1 track trigger for CMS based on Hough transform (GPUs) have been developed for both GPU vendors (NVIDIA and AMD):
  - ✓ A total latency of ***30 µs*** has been achieved with KIT-DMA and NVIDIA GPUs

What's next

- ✓ Characterization/optimization of the Hough transform algorithm:
  - ✓ Merging of DMA – Hough transform kernels → expected a total time ***~ 9 us***
  - ✓ Comparison between OpenCL and CUDA and FPGAs (Thomas Schuh, ID: RTA1_59)

- ✓ Significant technological evolution can be expected in the coming years, GPUs and FPGA can obtain full benefit with a timely development schedule.
  - ✓ Develop a next demonstrator based on Ultrascale+ Xilinx family:
  - ✓ Next generation of GPUs and NVLink @NVIDIA high-speed bidirectional bus protocol to exchange up to 320 Gb/s
- ✓ *NVLink in FPGA for high bandwidth zero latency communication*



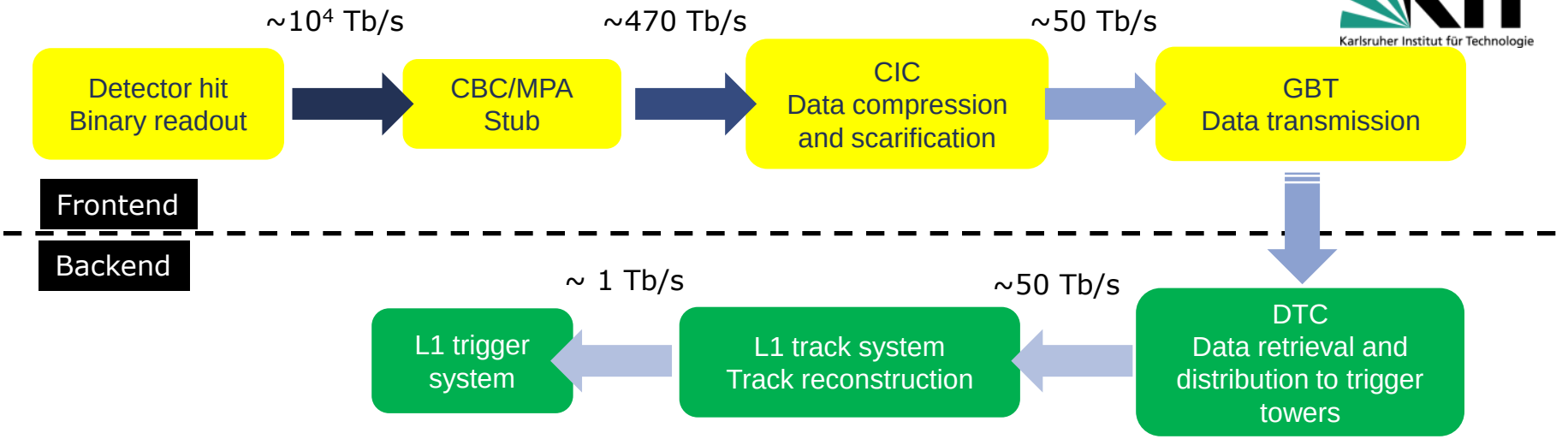
.. Room for improvements …

Karlsruher Institut für Technologie

Thank you for your attention

Backup slides

CMS phase II tracker readout chain



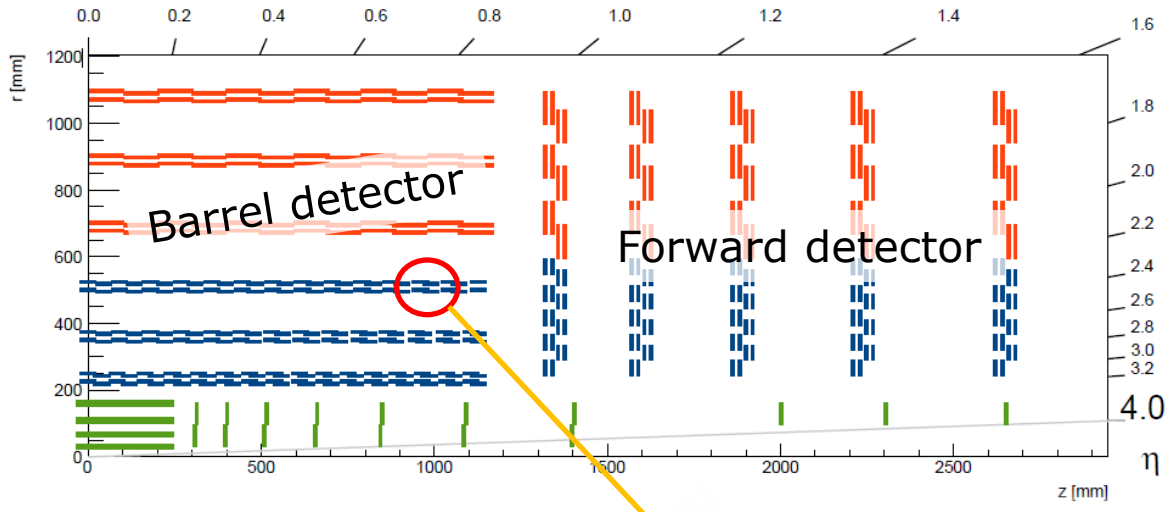
Extreme challenge: reconstruct *O(100)* tracks from *O(10k)* stubs at *40* MHz

	Patter recognition	*Fitting*	*Slices*
Associative Memory	Large bank of patter stored in a dedicated **AM** chip	PCA, Hough transform, Retina (**FPGA**)	48 = 8x6 (φxη) Loading balancing time
Tracklet algorithm	conventional road-based track search (**FPGA**)	linearized χ^2 fit (**FPGA**)	168 = 28x6 (φxη) 4x (BX)
Time-Multiplexed architecture	Hough transform (**FPGA**)	?	5 sectors in φ and time multiplexing of 24

What's about GPUs for L1 track finding?

To reduce the hardware devices (AMs + FPGAs) and increase the trigger flexibility

CMS and new tracker detector for fast pT dissemination

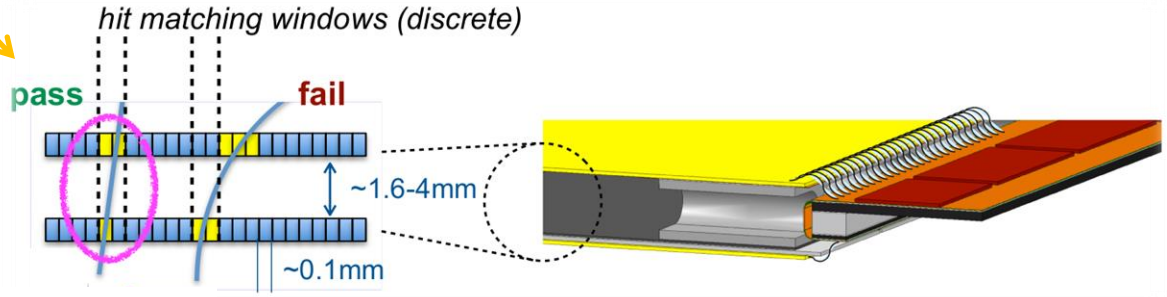


- Pixel detector
- Pixel-Strip (PS) module
- 2S (two strip sensors) module

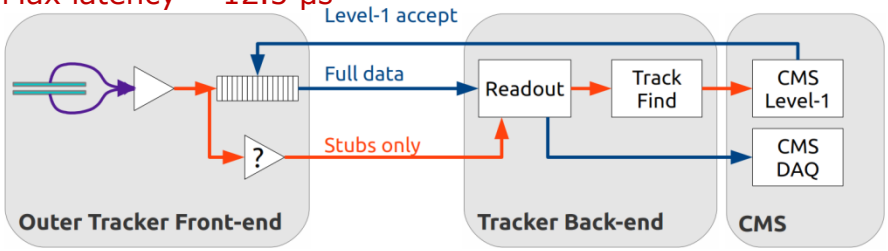
Barrel detector: **4130** PS modules (three layers) and **4464** 2S modules (three layers).

All outer tracker with **15 508** modules in total

L1 stubs are processed off-module, in the back end, to build L1 track primitives



Max latency = 12.5 μs



Track find latency = 5 μs

Each bunch crossing produces on the order of 10,000 stubs (PU 140). Only about 5 to 10% of these stubs actually belong to primary tracks with pT > 2 GeV.

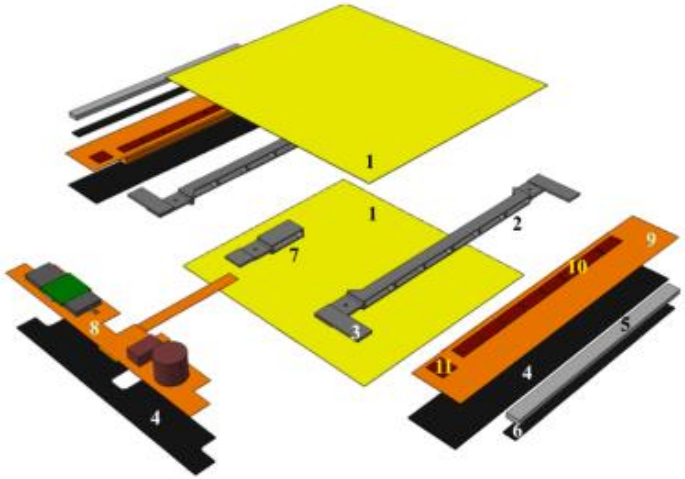
The goal of the L1 Track Finding system is to reconstruct the tracks of primary particles with pT > 2 GeV and discard as many as possible of all the other stubs.

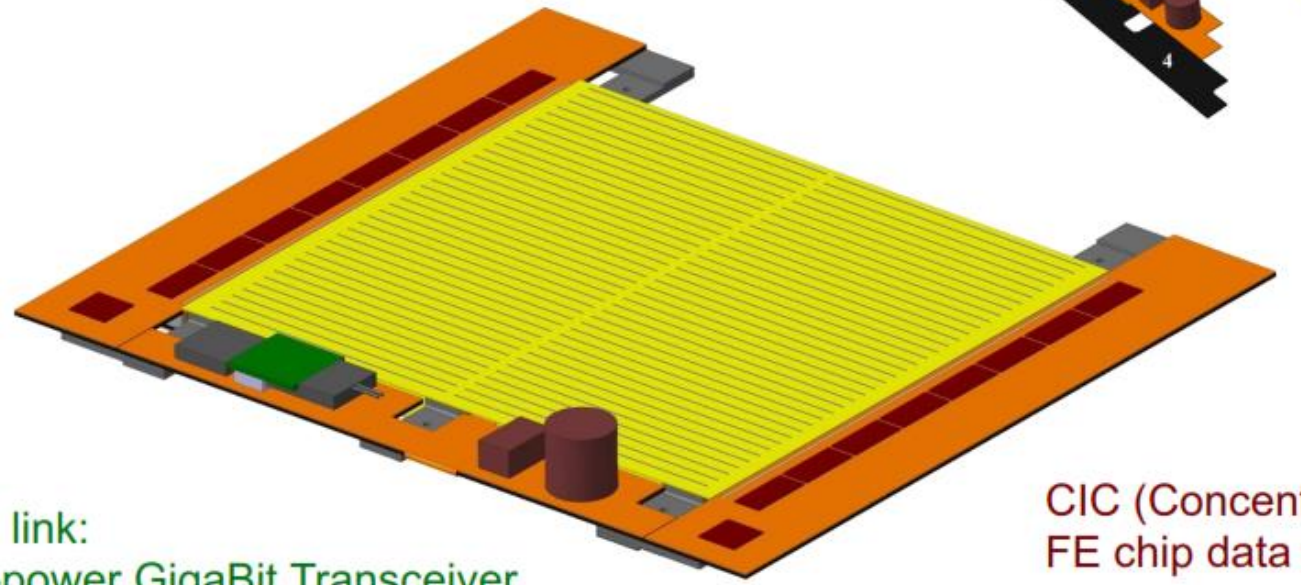
P_T Modules for track trigger: 2S

2S module

5cm×90μm AC coupled strips (both sides) pitch, ~10x10cm^2 ,P~4W

Front-end electronics (CBC chips), specialized for strips features top/bottom sensor correlation





Low-mass mechanical structures optimized for cooling

Data link:
Low-power GigaBit Transceiver (LpGBT) + laser driver currently under development

CIC (Concentrator IC):
FE chip data sparsification

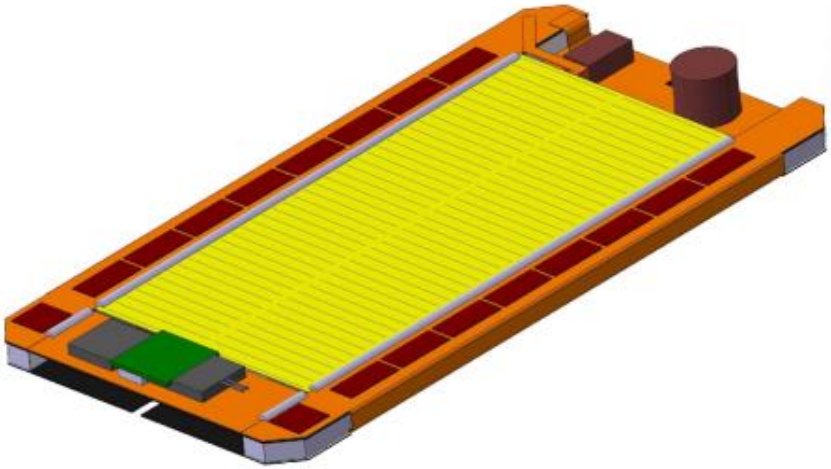
DC/DC converter (already foreseen in Phase-1 pixel project) 10-12V lines: lower current, lower material

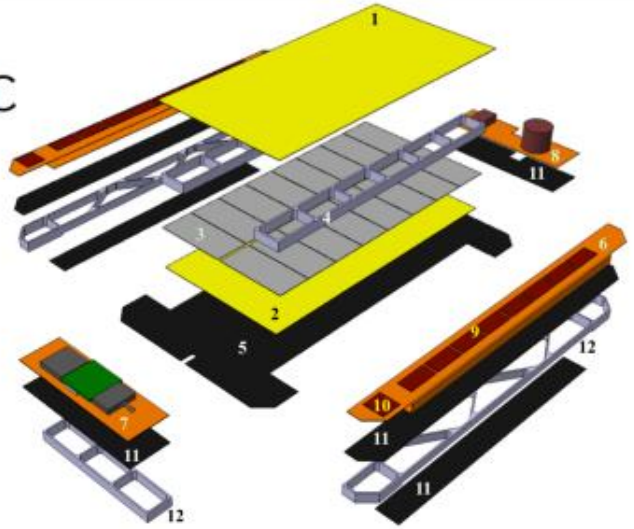
P_T Modules for track trigger: PS

PS module

2.4cm×100μm AC coupled strips + 1.5mm×100μm DC coupled macro pixels, ~5x10cm^2 ,P~6-8W

Front-end electronics, specialized for strips (SSA chip) and pixels (MPA chip), features top/bottom sensor correlation





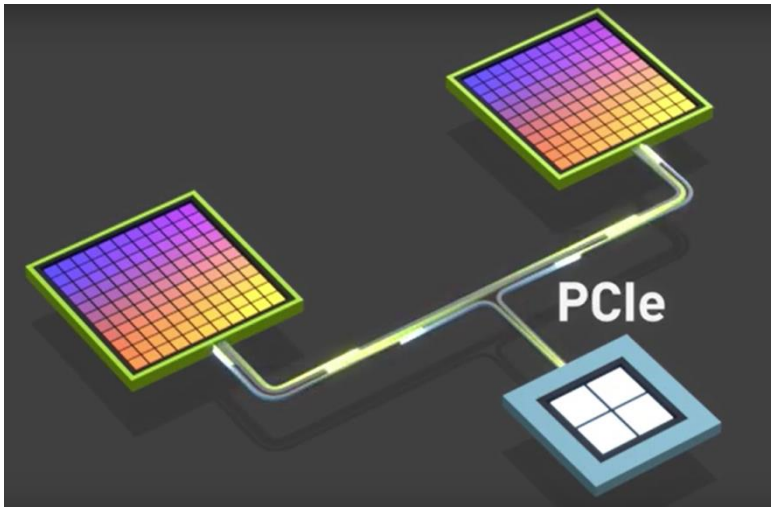
Low-mass mechanical structures optimized for cooling

CIC (Concentrator IC):
FE chip data sparsification

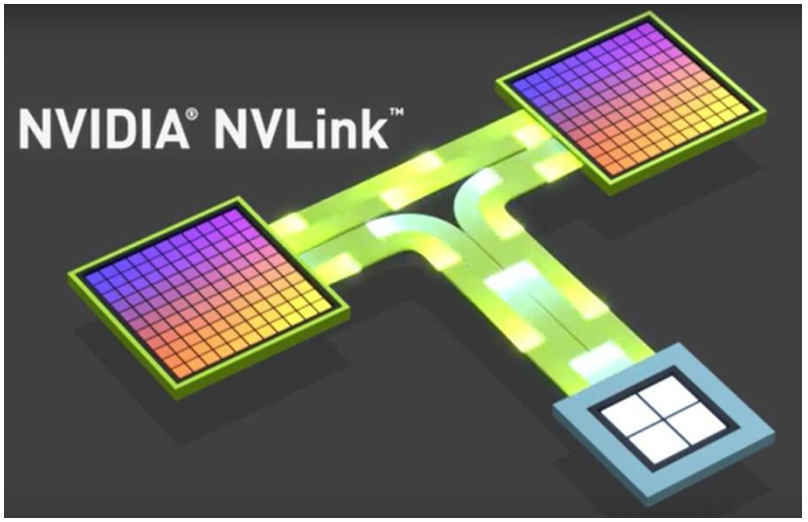
Data link:
Low-power GigaBit Transceiver (LpGBT) + laser driver currently under development

DC/DC converter (already foreseen in Phase-1 pixel project) 10-12V lines: lower current, lower material

GPUDirect vs NVLINK (NVIDIA)



GPUDirect communication

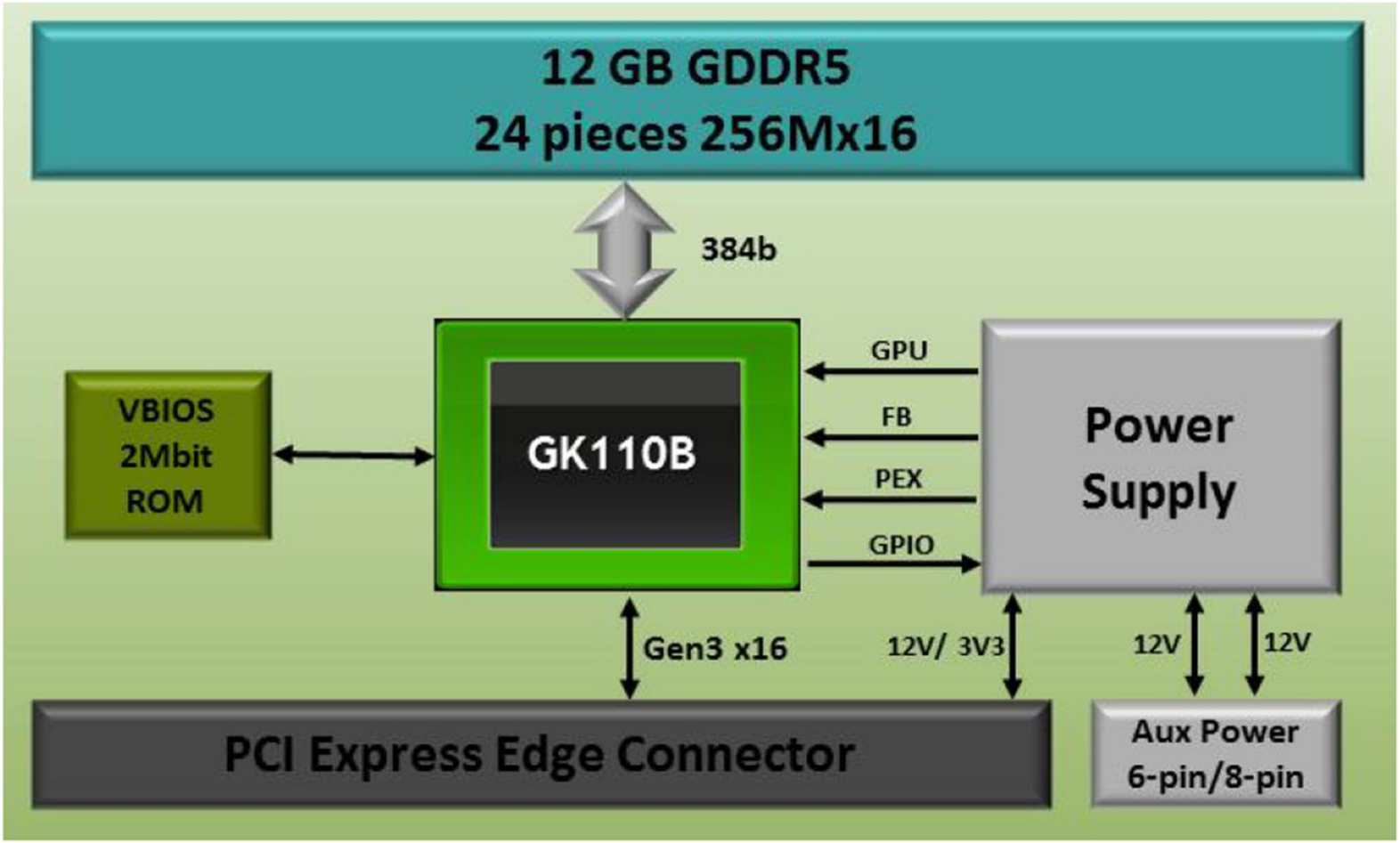


NVLINK communication

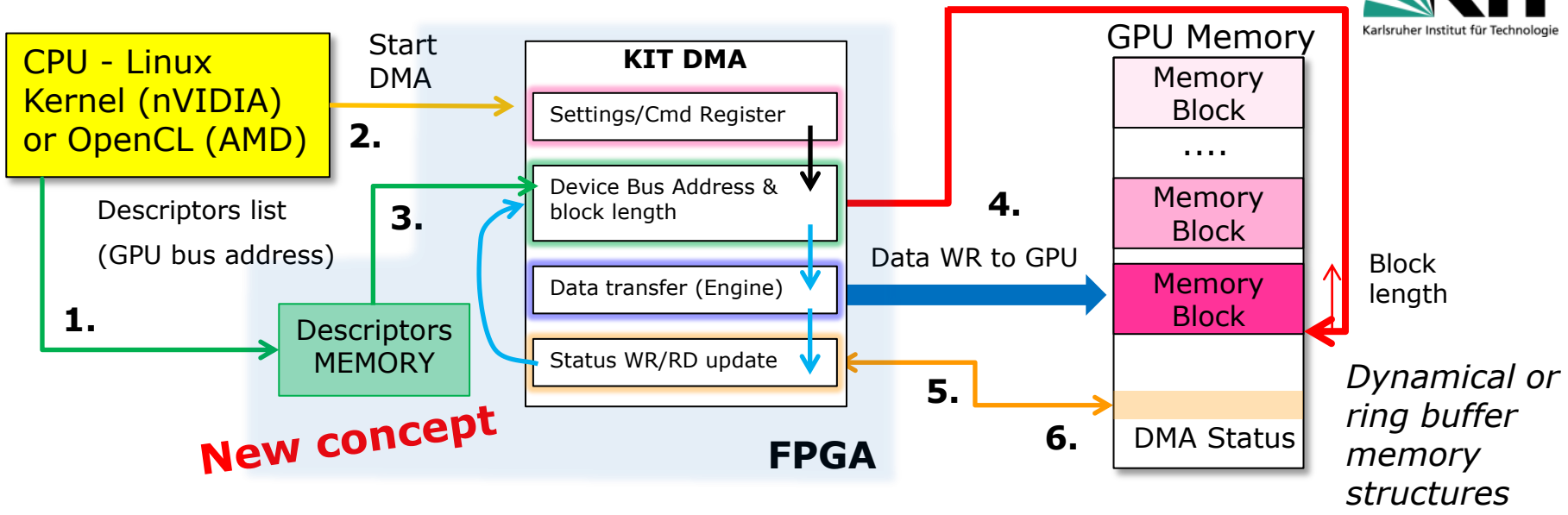
NVIDIA® NVLink™ is a high-bandwidth, energy-efficient interconnect that enables ultra-fast communication between the CPU and GPU, and between GPUs. The technology allows data sharing at rates 5 to 12 times faster than the traditional PCIe Gen3 interconnect, resulting in dramatic speed-ups in application performance and creating a new breed of high-density, flexible servers for accelerated computing

See more at: http://www.nvidia.com/object/nvlink.html#sthash.7RlpyR8X.dpu

K40 (NVIDIA)


12 GB GDDR5
24 pieces 256Mx16
384b
VBIOS
2Mbit
ROM
GK110B
GPU
FB
PEX
GPIO
Power
Supply
Gen3 x16
12V/ 3V3
12V
12V
PCI Express Edge Connector
Aux Power
6-pin/8-pin

Novel concept of DMA (KIT)



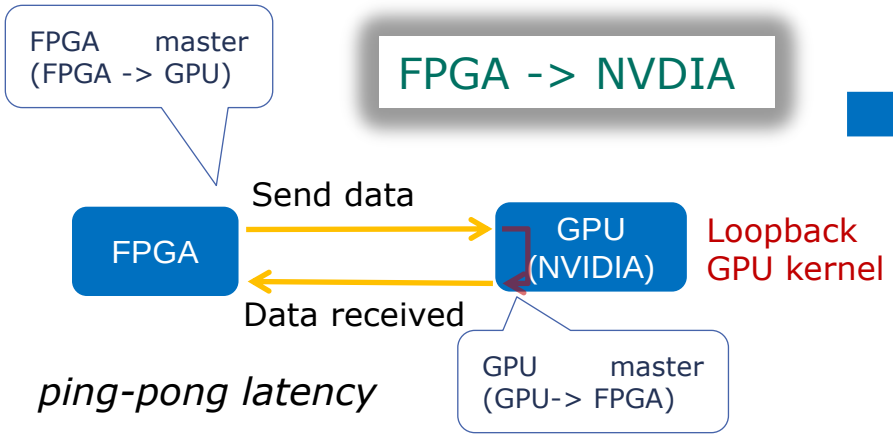
Operations:

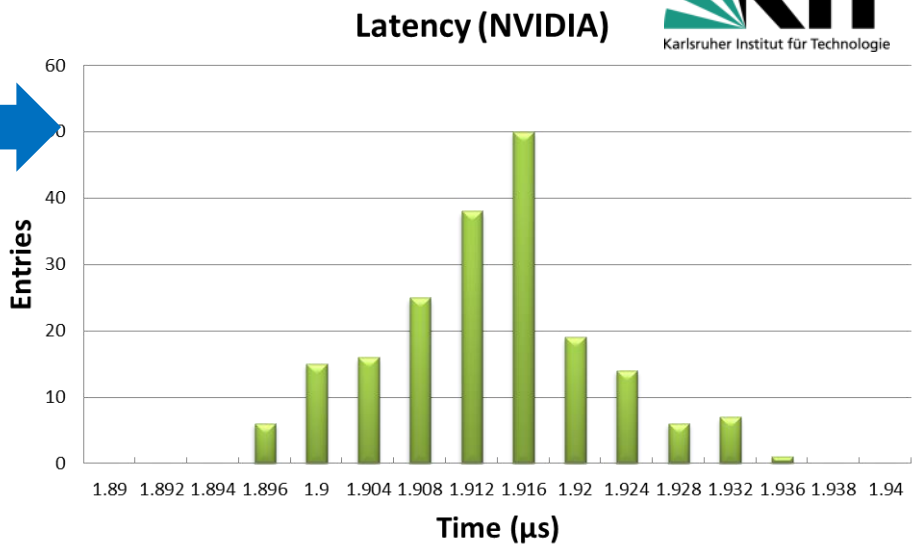
1. GPU memory allocation (**nvidia_p2p_get_pages()** or **clCreateBuffer()** OpenCL).

→ write the "Bus addresses" into FPGA descriptor memory, **addr.surface_bus_address()** OpenCL

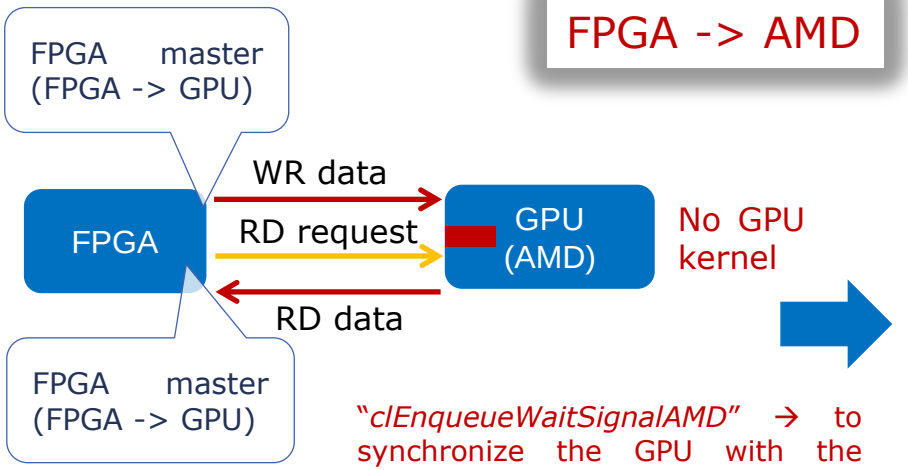
2. Start DMA data transfer
3. DMA load the descriptor from the memory and fetch the DATA
4. Data transfer from FPGA → to GPU memory block (defined by descriptor)
5. DMA Update the Status for GPU kernel → number of blocks written, current descriptor address
6. DMA receive the current descriptor read by driver and therefore free for the next block transfers

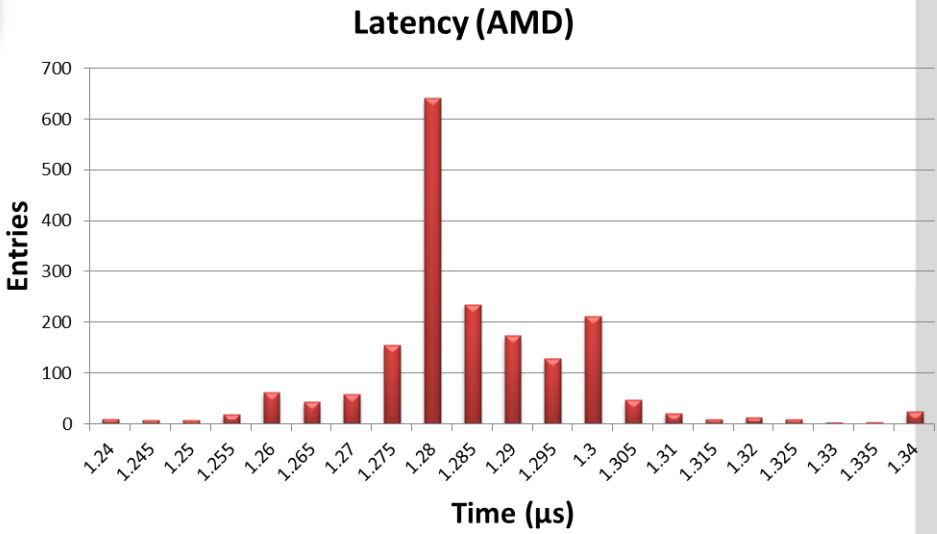
Readout system – performance / comparison





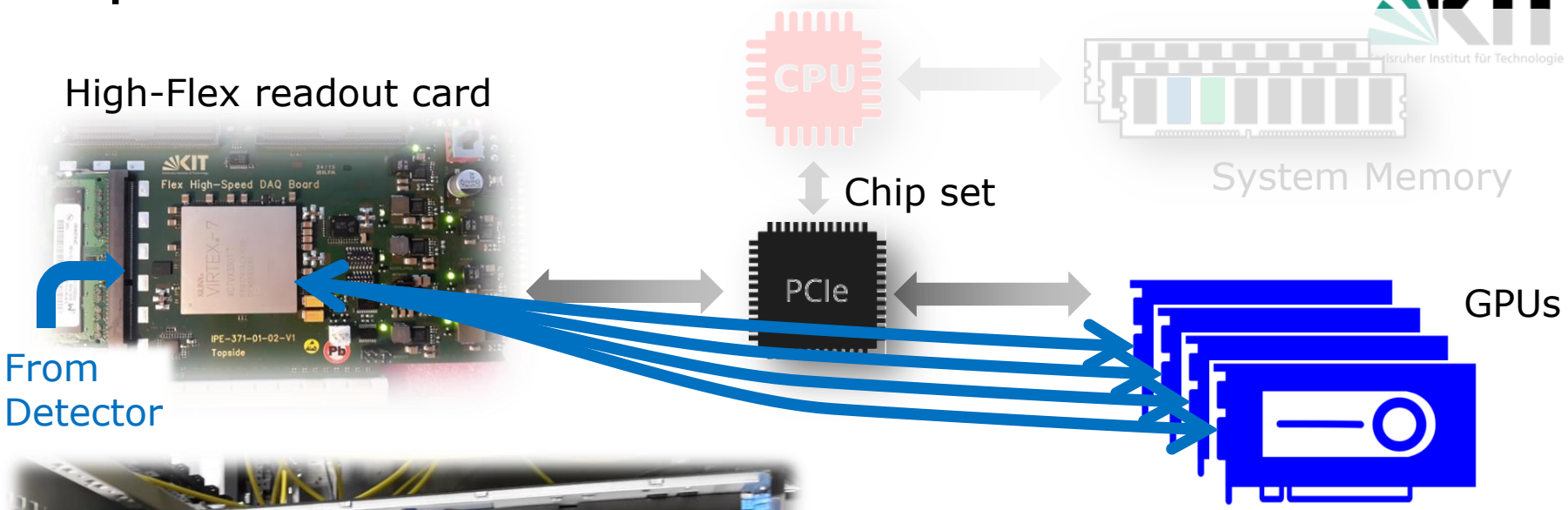
NVIDIA: Latency < 2 µs, jitter < 30 ns





Latency < 1.3 µs, three PCIe transactions

Multiple “GPU-Direct” architecture



Only 20 ns are necessary to switch from GPU_1 to GPU_N

Ref: Ultra-fast computer tomography real-time 3D reconstruction (data rate 50Gb/s). DOI: 10.1109/TNS.2015.2425911. *Presented to Real-time 2014 -Nara*

Ref: Streaming Camera Platform for Scientific Applications. DOI: 10.1109/TNS.2013.2252528. *Presented to Real-time 2012 – Lawrence Berkeley Laboratory*