GPU-based online track reconstruction for the MuPix-telescope

Carsten Grzesik
for the Mu3e collaboration

February 29, 2016

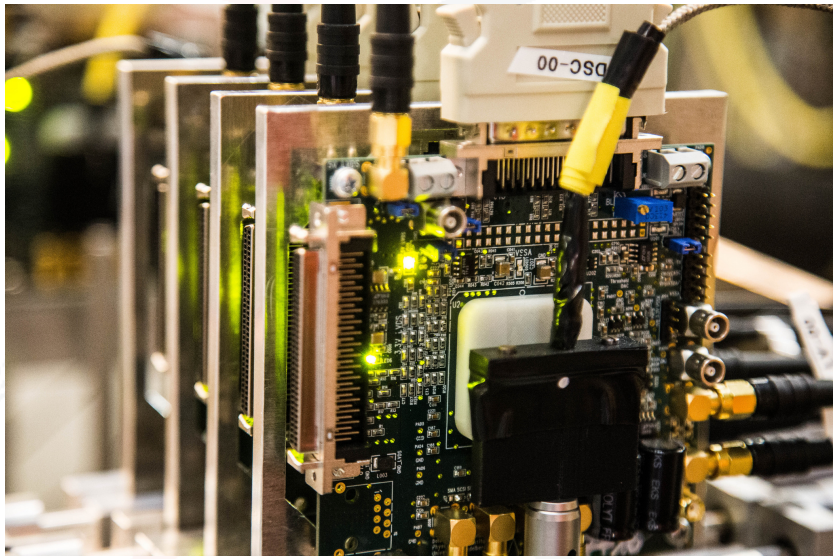
Motivation

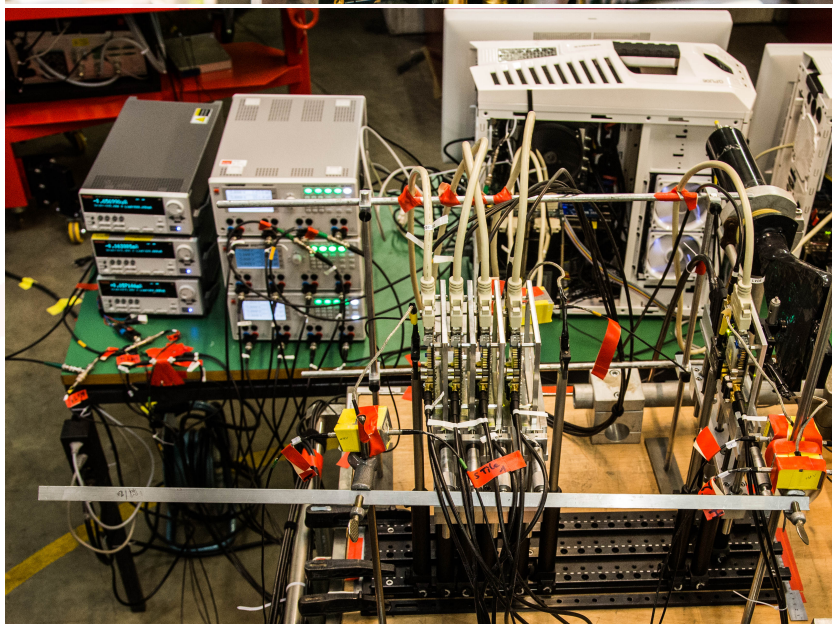
Mu3e experiment

- high data rate: $\sim 1\,\mathrm{Tbit\,s^{-1}}$
- online track reconstruction
- reduction factor: ~ 1000

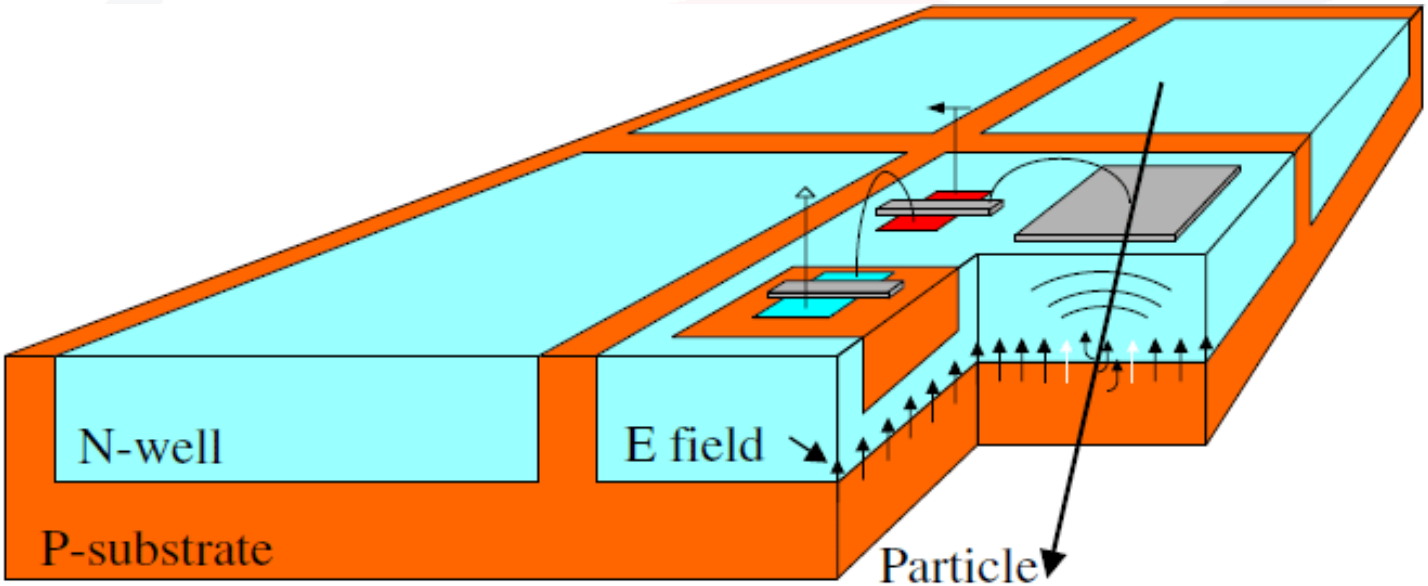
MuPix telescope

- test setup: pixel sensors, readout and online reconstruction
- high beam rates: $\mathcal{O}(1\,\mathrm{MHz})$
- max. output rate: $4 \times 1.25\,\mathrm{Gbit\,s^{-1}}$





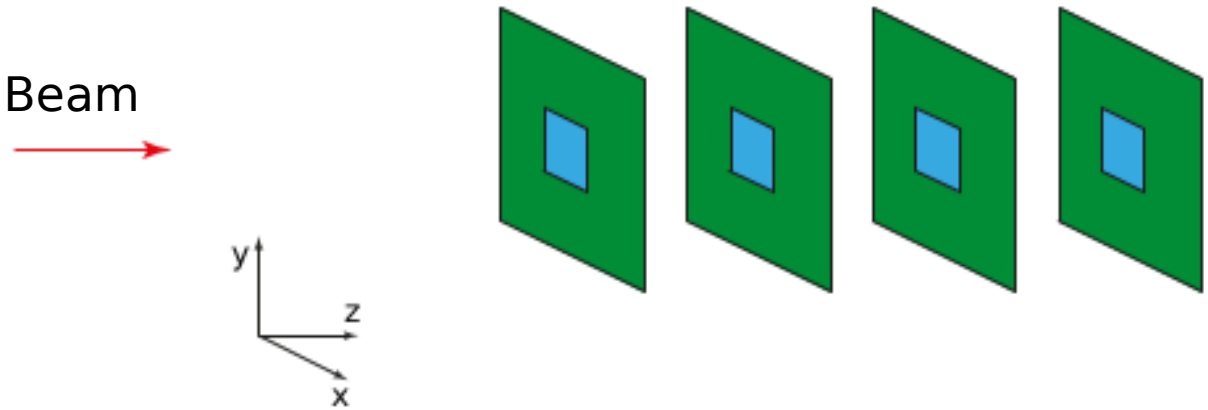
High Voltage Monolithic Active Pixel Sensor

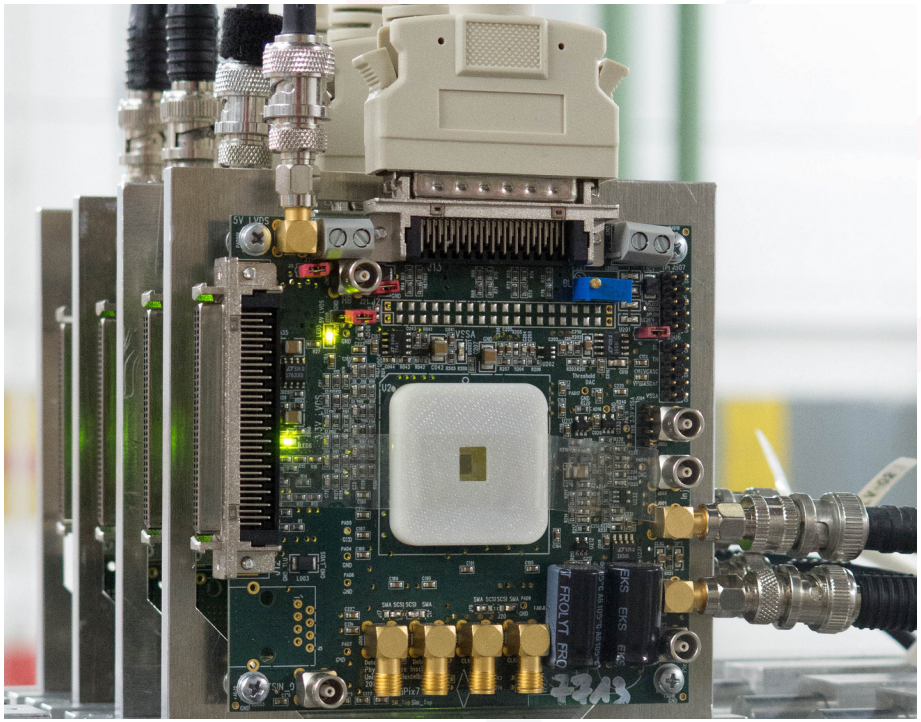


- 180 nm HV-CMOS technology
- reverse biased up to 90 V
- thin depletion region
- thinning to 50 µm
- readout logic directly on chip
- zero suppressed, serial data output 1.25 Gbit s^{-1}
- details in T72.1/2/3

I.Perić,Nucl.Instrum.Meth., 2007, A582, 876

Setup

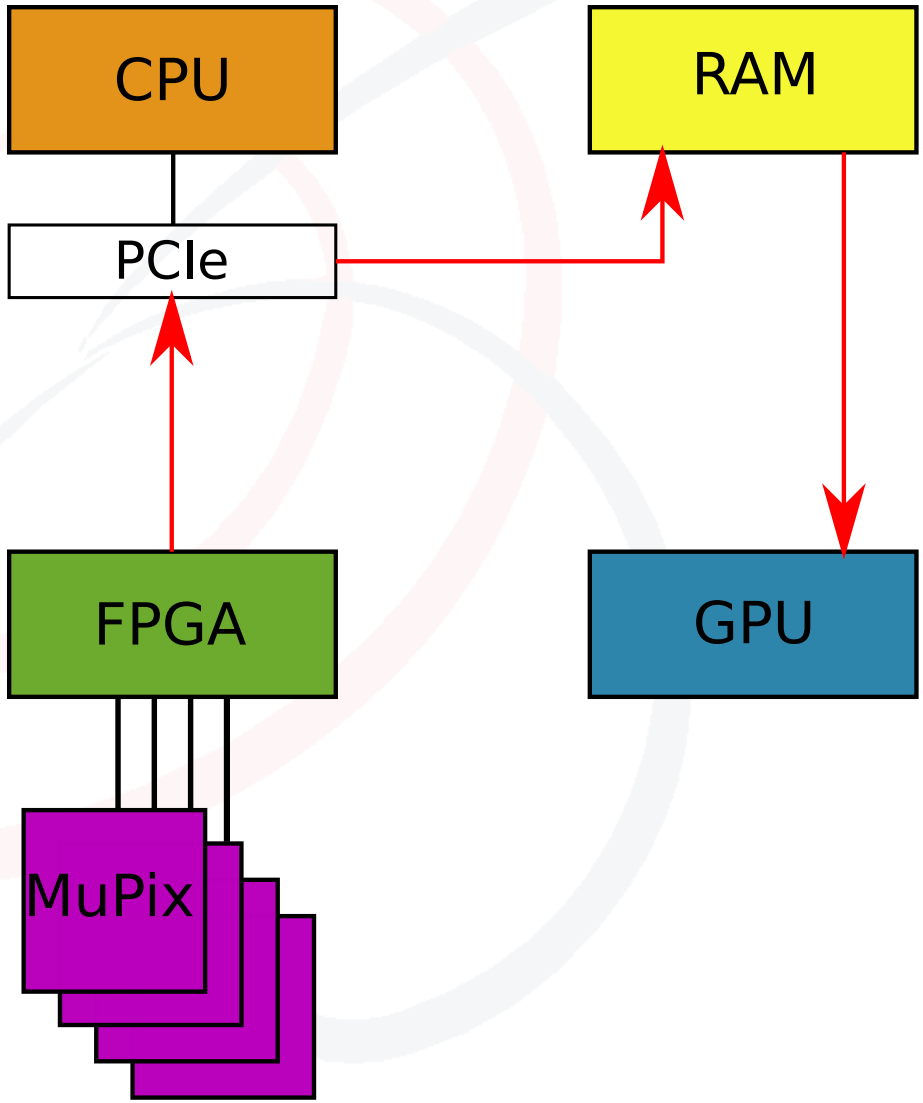




- MuPix7
- sensor size: $3.2 \times 3.2\,\mathrm{mm}^2$
- serial data output via LVDS

Data Transmission - DMA

- serial data output from sensor planes
- merge and sort on FPGA
- PCIe connection to PC
- Direct Memory Access to GPU not available
- DMA via main memory
- data rate: $\leq 1.5\,\mathrm{GB\,s^{-1}}$

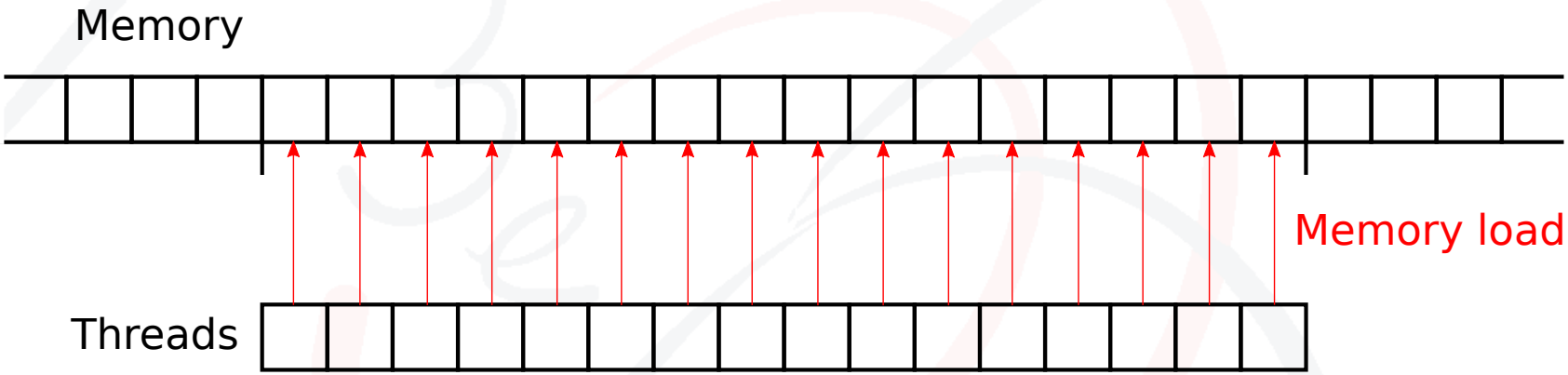


Graphics Processing Unit

- programming: CUDA API
- commercial gaming GPUs
- GTX 980: 2048 cores @ 1.3 GHz

- straight track model $\rightarrow$ few calculation steps
- combinatorics of hits $\rightarrow$ lots of memory loads
- memory bound algorithm $\rightarrow$ need high memory throughput

Memory Coalescing



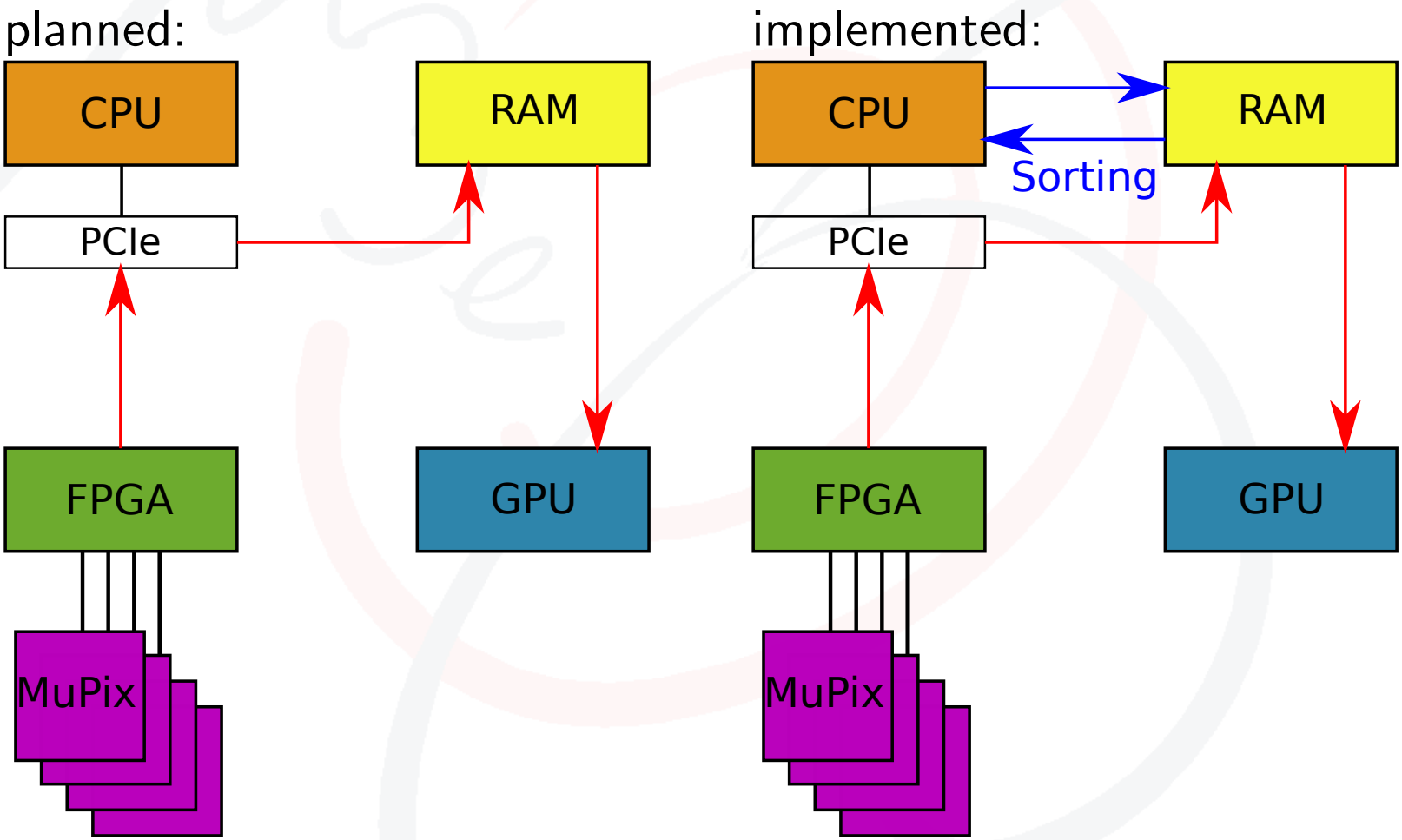
- example for 16 threads/SM
- 16 threads perform same operation at same time (e.g. memory load)
- for consecutive data in memory $\rightarrow$ grouped in 1 load operation

GPU implementation

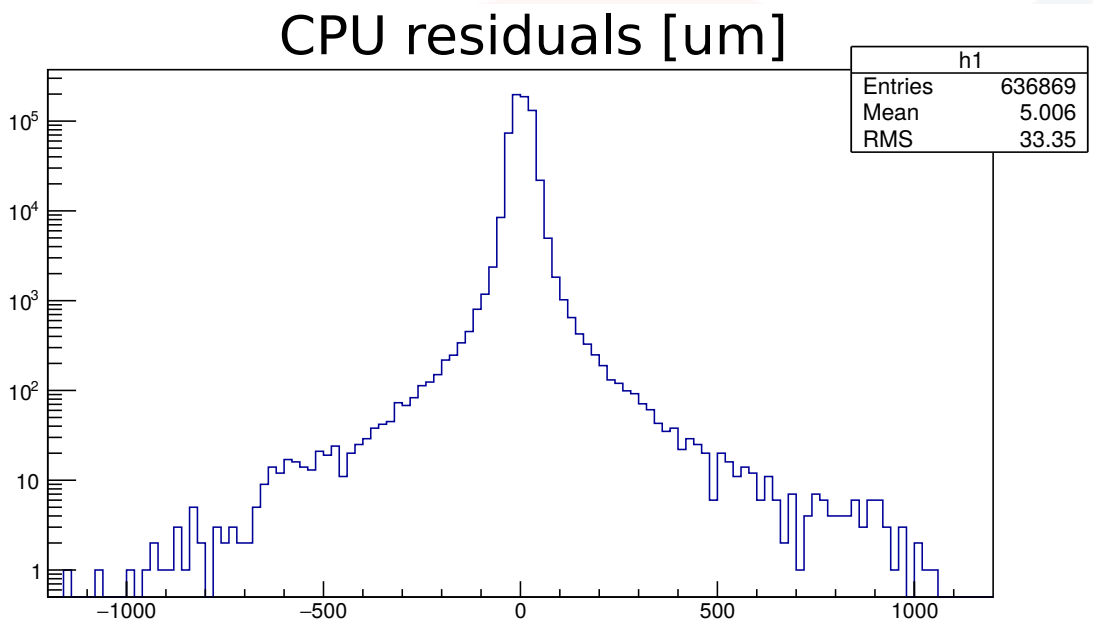
- parallelization: one timeframe per thread $\rightarrow$ no communication required across thread boundaries
- hits from consecutive frames next to each other (coalesced memory access)
- need to sort the data by plane and time

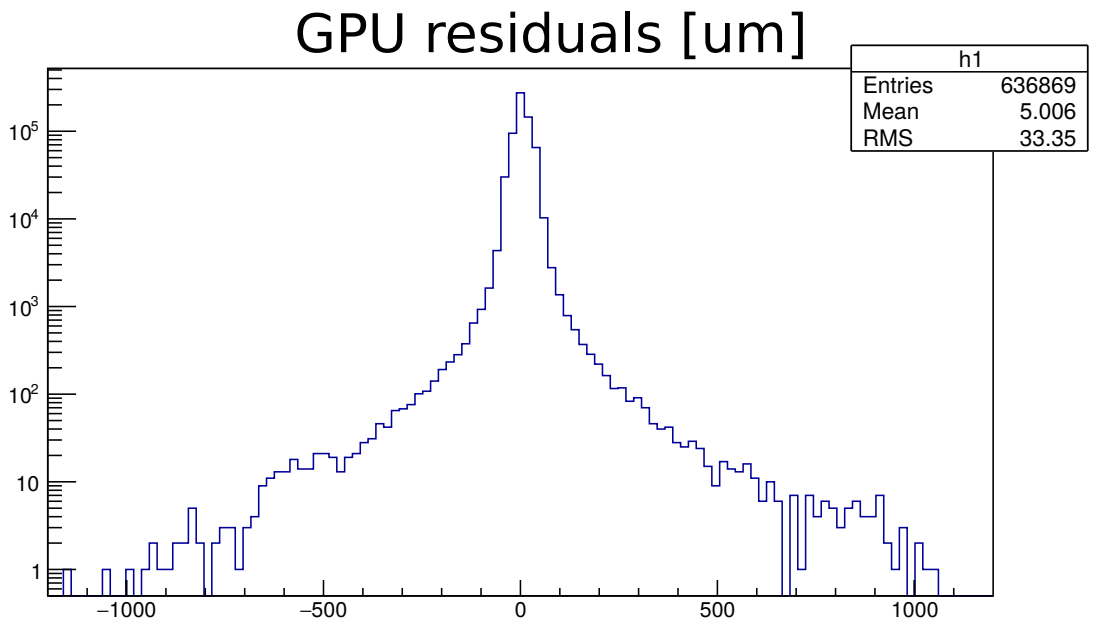
memory pos.	0	1	2	...	31
hit.plane.frame	0.0.0	0.0.1	0.0.2	...	0.0.31
memory pos.	32	33	...	...	63
hit.plane.frame	1.0.0	1.0.1	...	...	1.0.31
...					
memory pos.	256	257	...		
hit.plane.frame	0.1.0	0.1.1	...		

Setup - DESY Testbeam

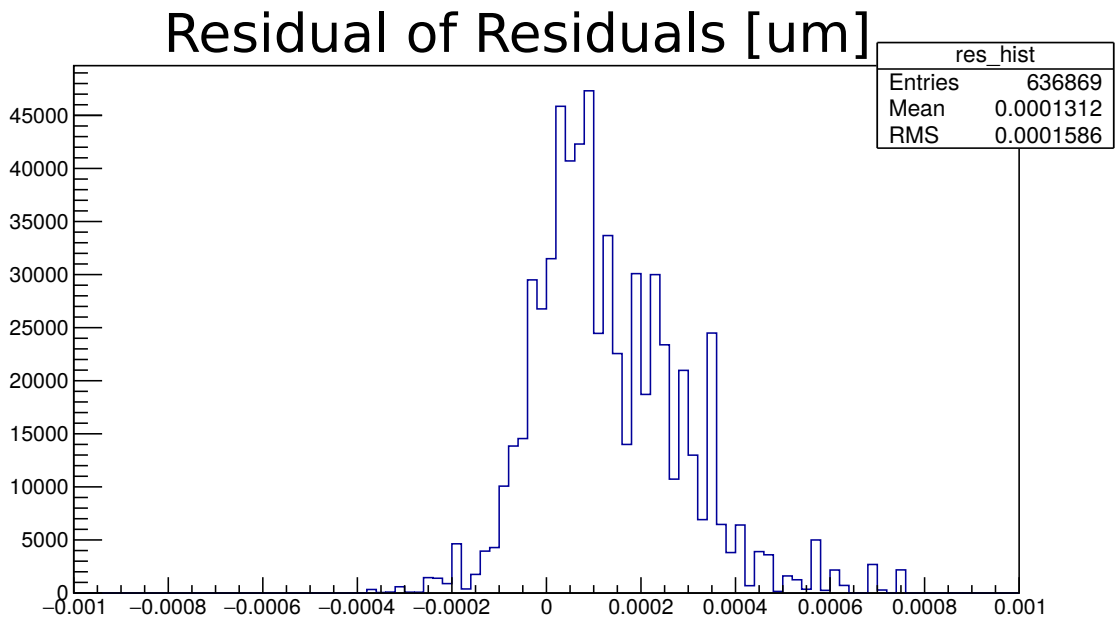


Results - DESY Testbeam





Results - DESY Testbeam



- deviation $< 1\,\mathrm{nm}$
- bias to bigger CPU values
- execution differences CPU/GPU (e.g. floating point precision)

Summary and Outlook

- GPU tracking implemented
- DMA working up to $1.5\,\mathrm{GB\,s^{-1}}$
- offline tracking on GPU gives reasonable results

to do:

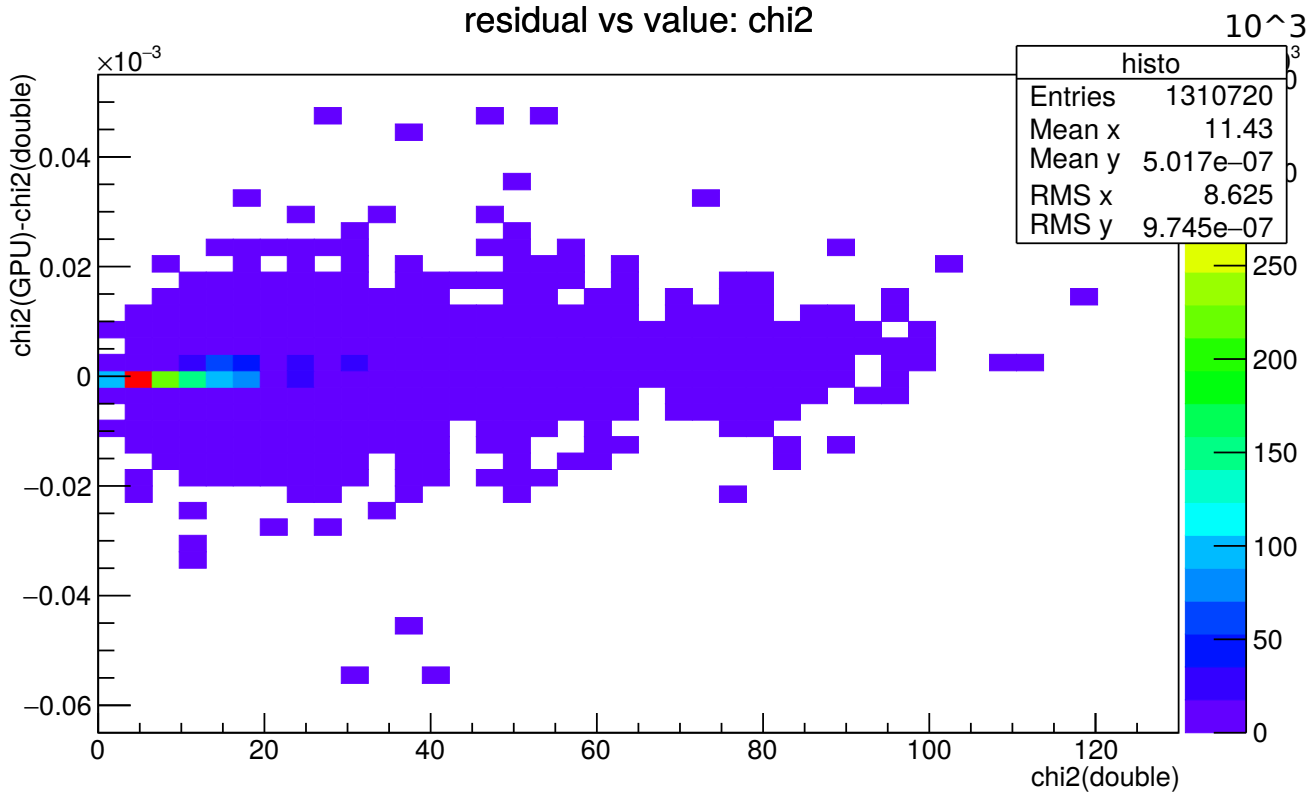
- finally test FPGA firmware
- use GPU tracking online
- optimization of GPU code

Acknowledgments

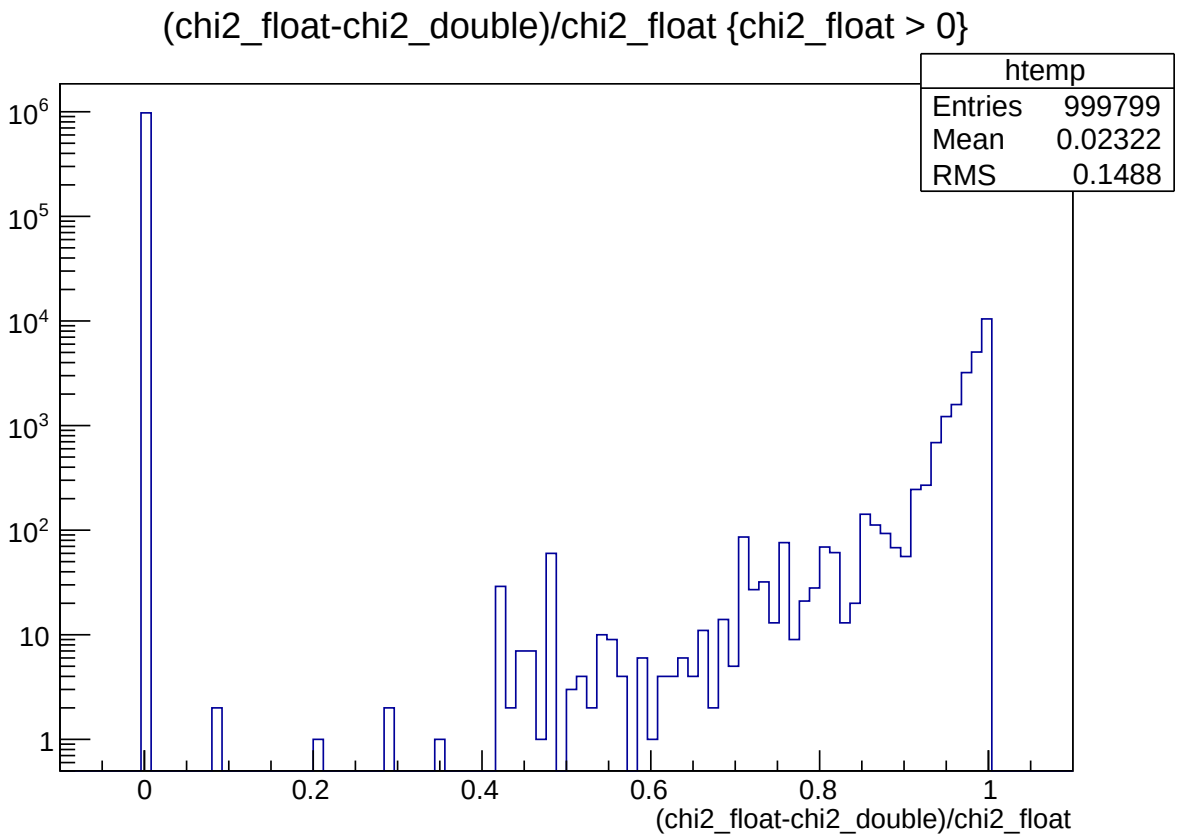
The measurements leading to these results have been performed at the Test Beam Facility at DESY Hamburg (Germany), a member of the Helmholtz Association (HGF)

Backup

- memory bound GPU kernels $\rightarrow$ 32 bit floating point
- IEEE 754 floating point arithmetic
- GPU uses Fused Multiply Add (FMA)

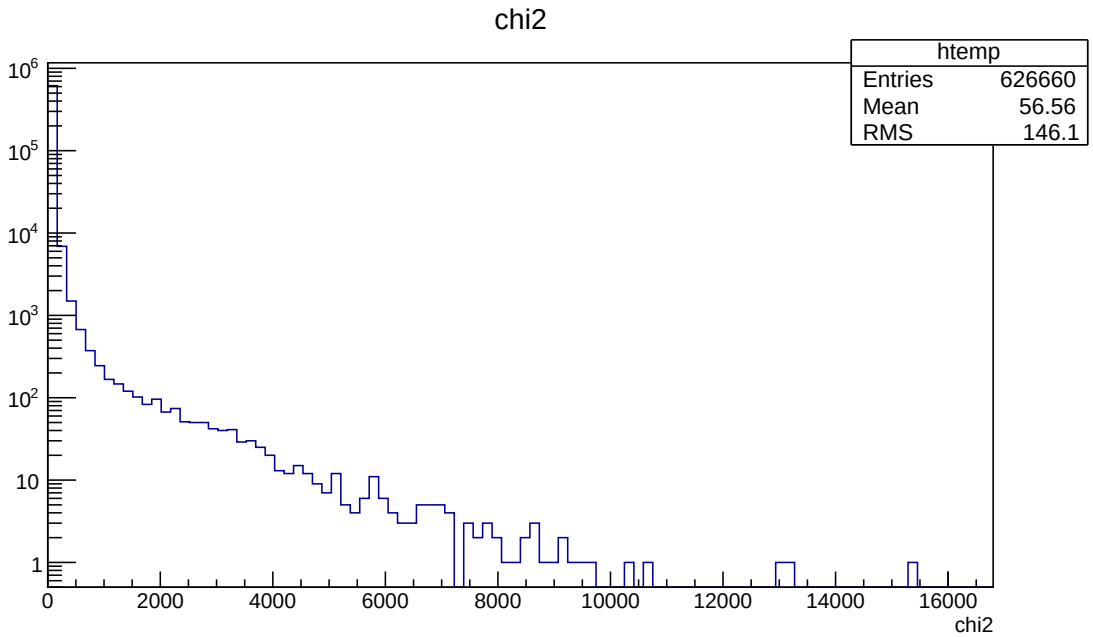


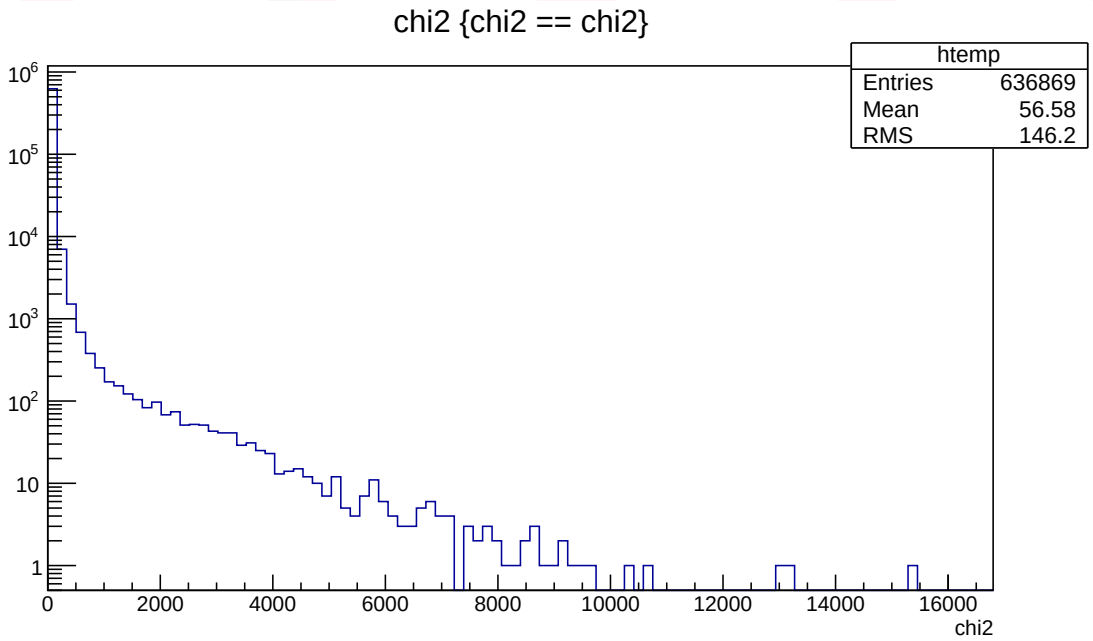
Backup



- IEEE 754:
- float ULP: 10^{-7}
- double ULP: 10^{-16}

Backup


chi2
htemp
Entries 626660
Mean 56.56
RMS 146.1
10^6
10^5
10^4
10^3
10^2
10
1
0 2000 4000 6000 8000 10000 12000 14000 16000
chi2


chi2 {chi2 == chi2}
htemp
Entries 636869
Mean 56.58
RMS 146.2
10^6
10^5
10^4
10^3
10^2
10
1
0 2000 4000 6000 8000 10000 12000 14000 16000
chi2