WP4 updates

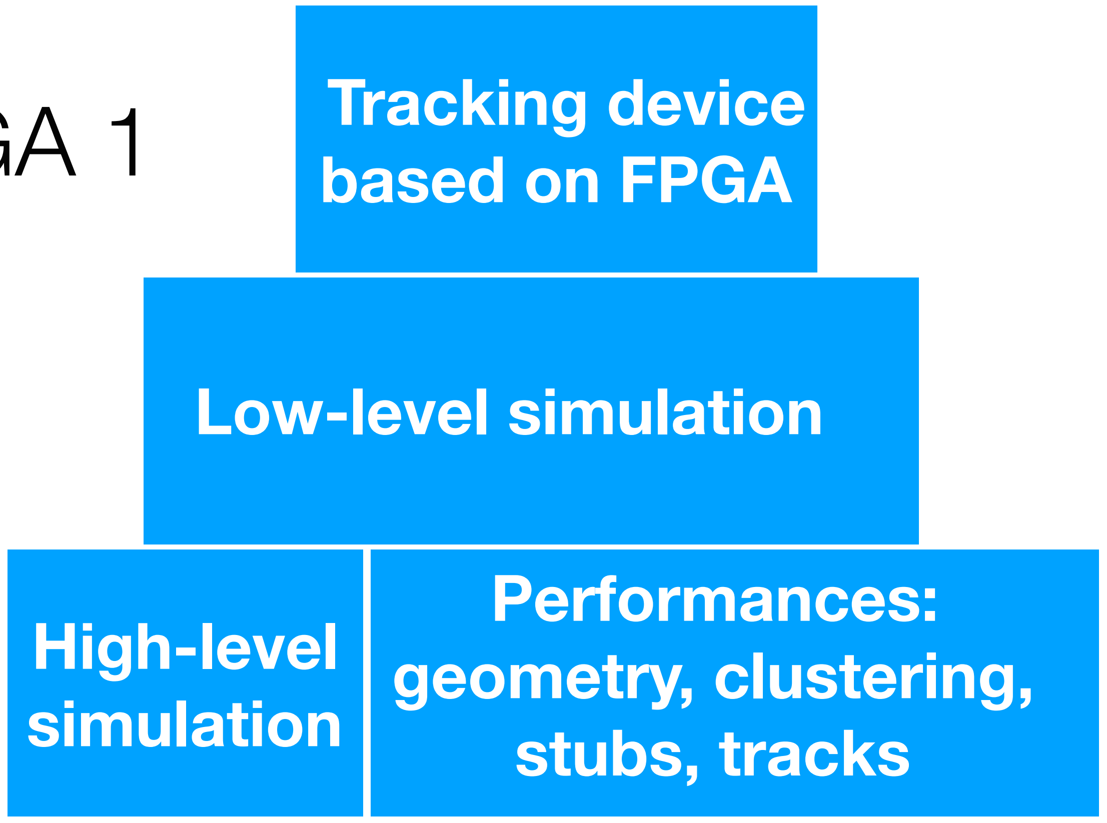
A. Carbone, **S. Maccolini**, N. Neri, M. Petruzzo, B. Siddi, S. Vecchi

TIMESPOT meeting

16/17 December 2021

WP4 activities

- TARGET: **Fast tracking**
  - clustering, stub maker on FPGA 1
  - tracking on FPGA 2


Tracking device based on FPGA
Low-level simulation
High-level simulation
Performances: geometry, clustering, stubs, tracks

- Simulations:
  - performance of 4D tracking using the TIMESPOT sensor and FPGA
  - optimization of the architecture
  - optimization of the geometry for stub based tracking

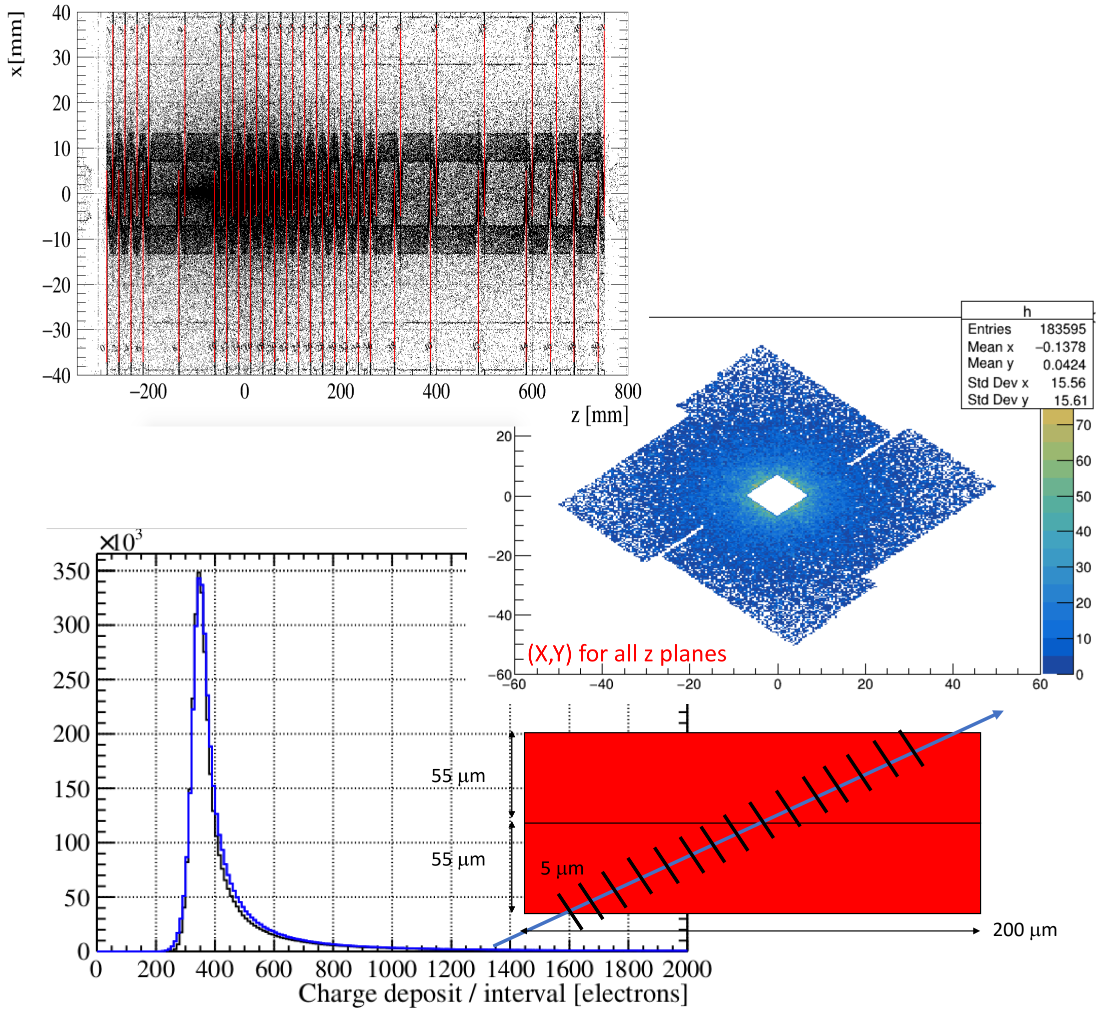
Performance studies using the TIMESPOT sensor

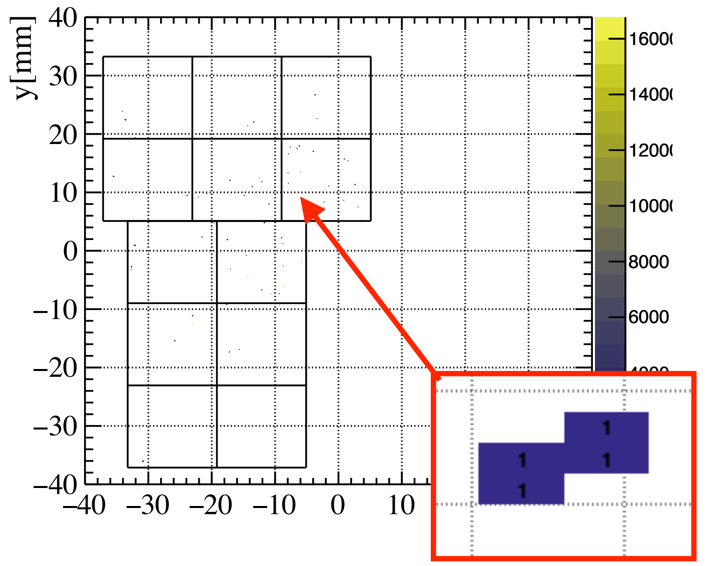
Primary and secondary particles from Pythia8+Gauss as input

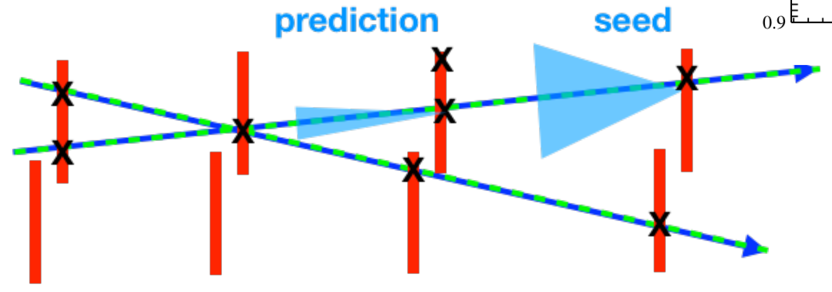
VeloPixel Fast simulation

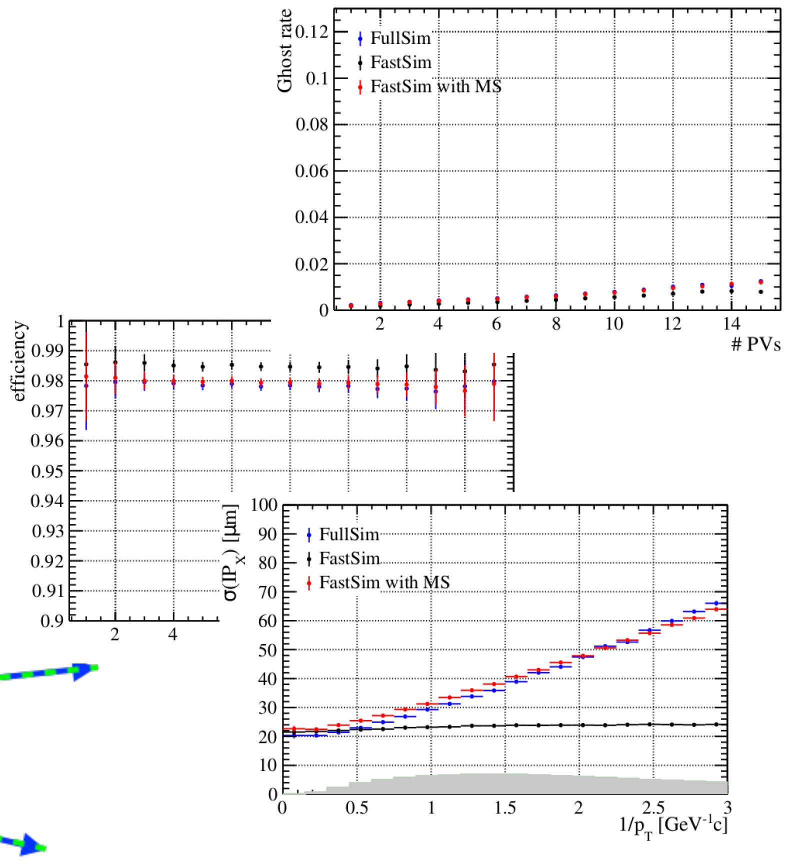
- The workflow:

Detector geometry + sensor response → Hit clustering + Track reconstruction → Performance studies

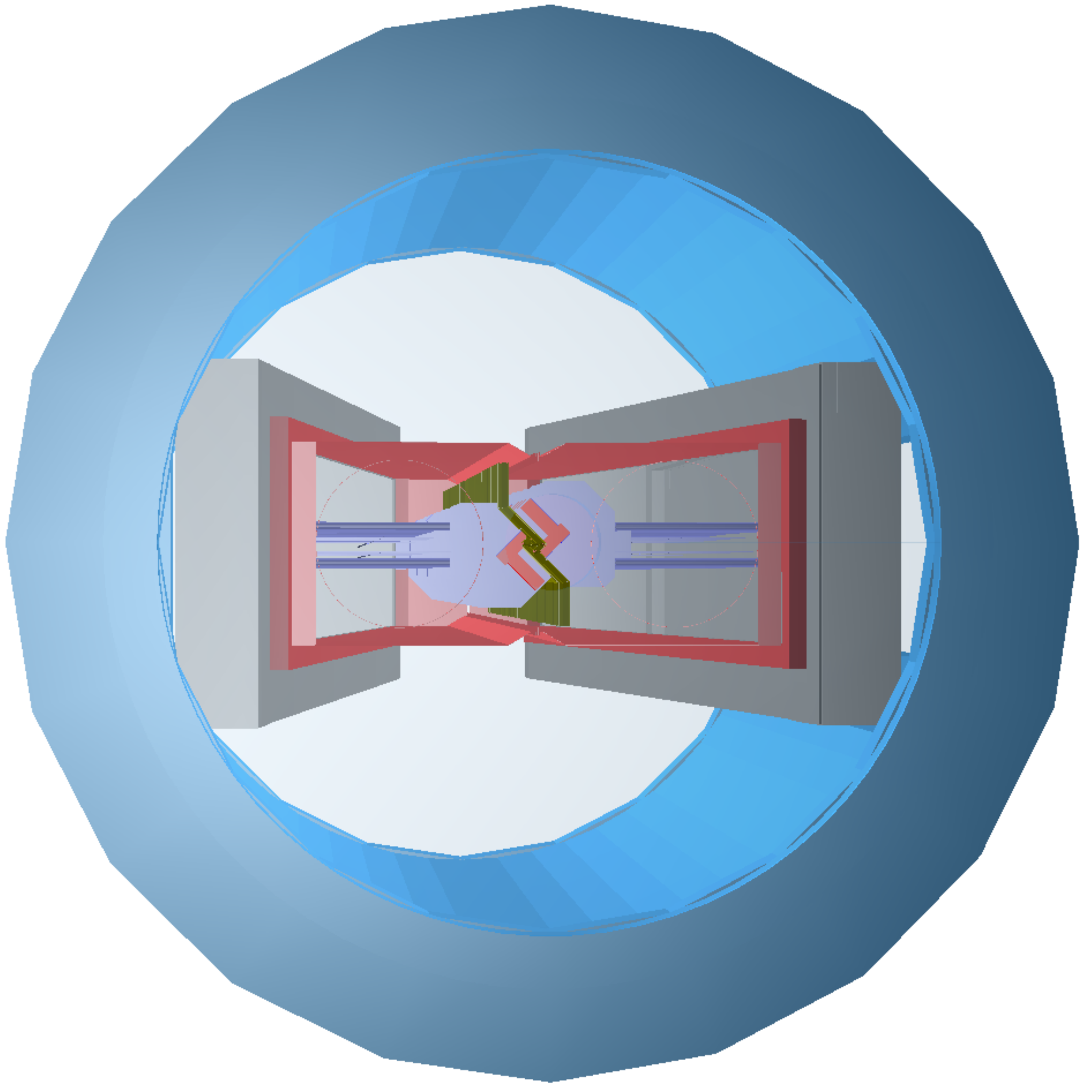


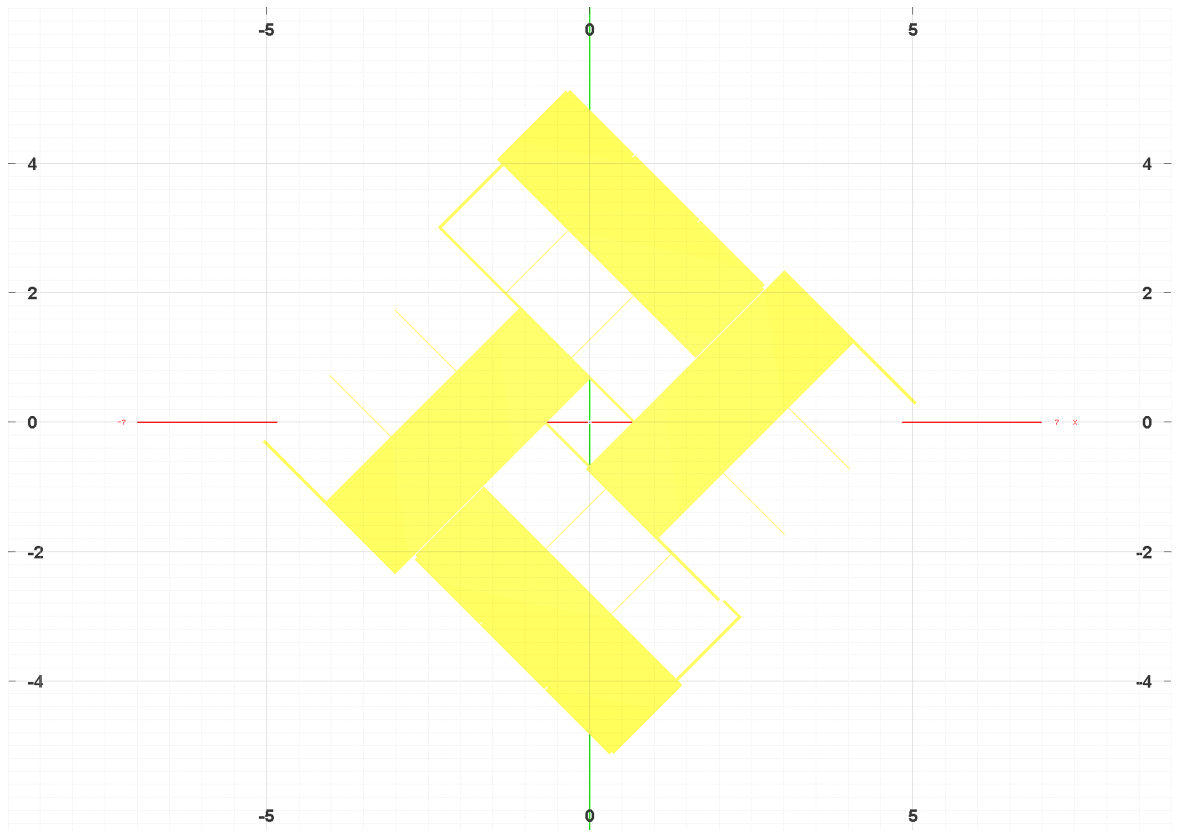


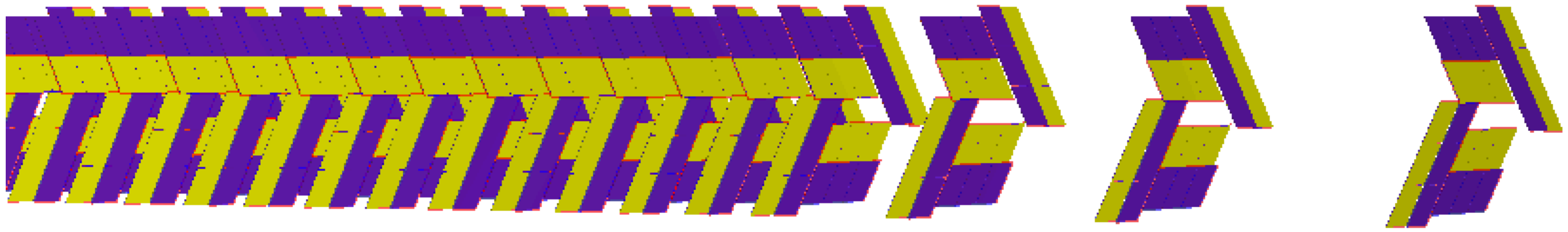




Moving to DD4Hep

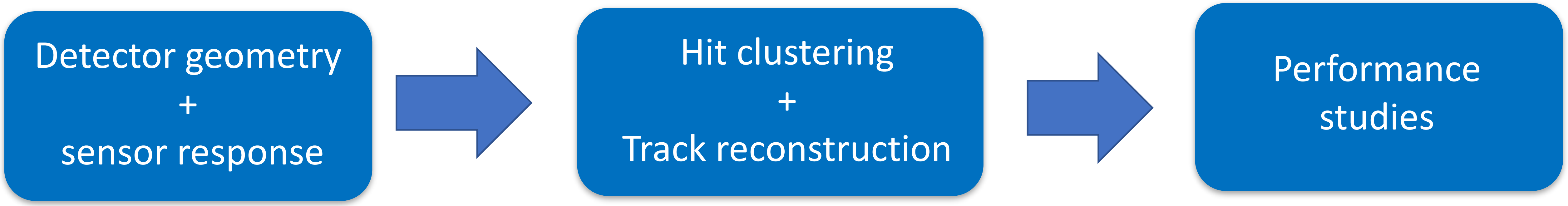




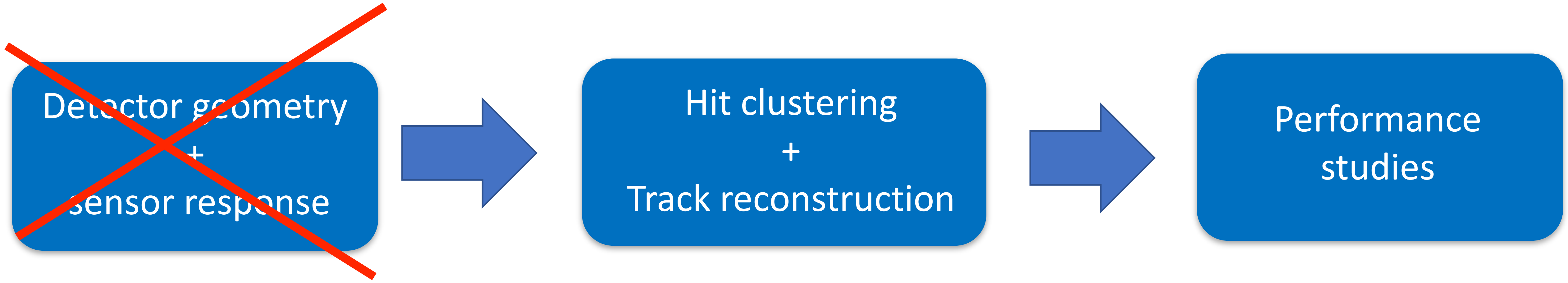


From MC particles to MC hits

- **MC particles** (primaries and secondaries) as input:

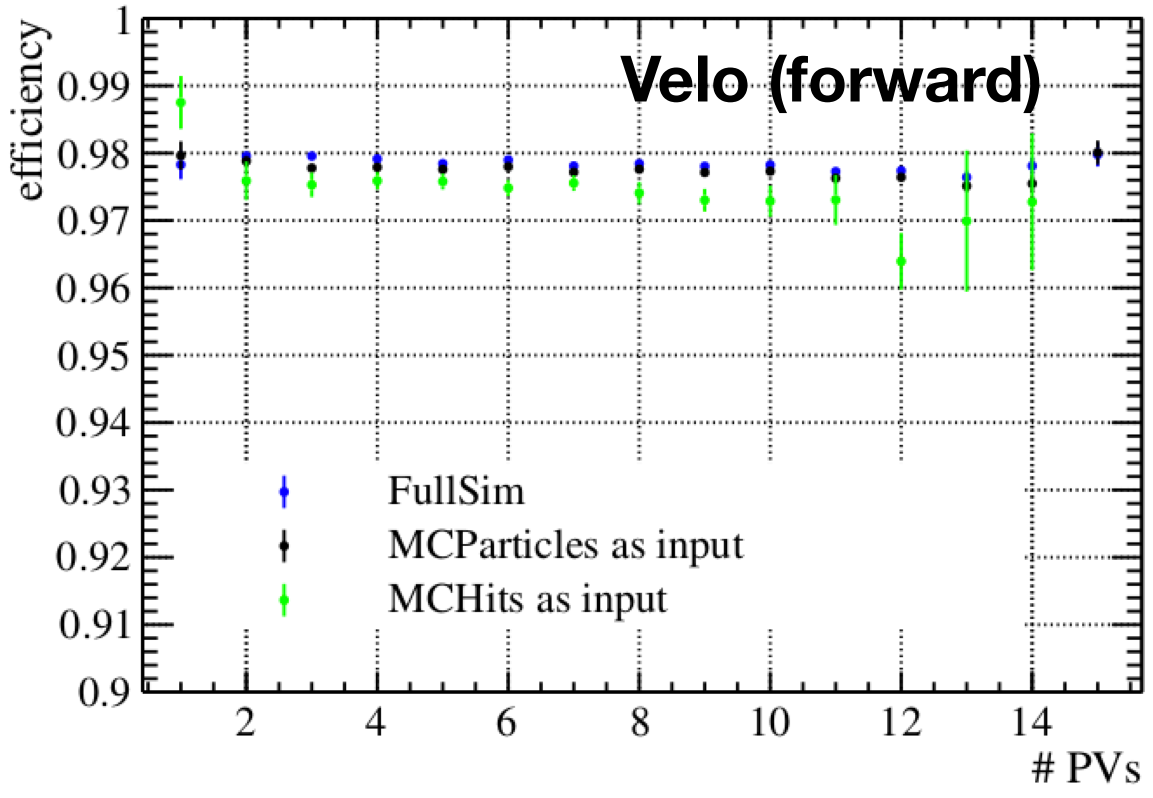

Detector geometry + sensor response
Hit clustering + Track reconstruction
Performance studies

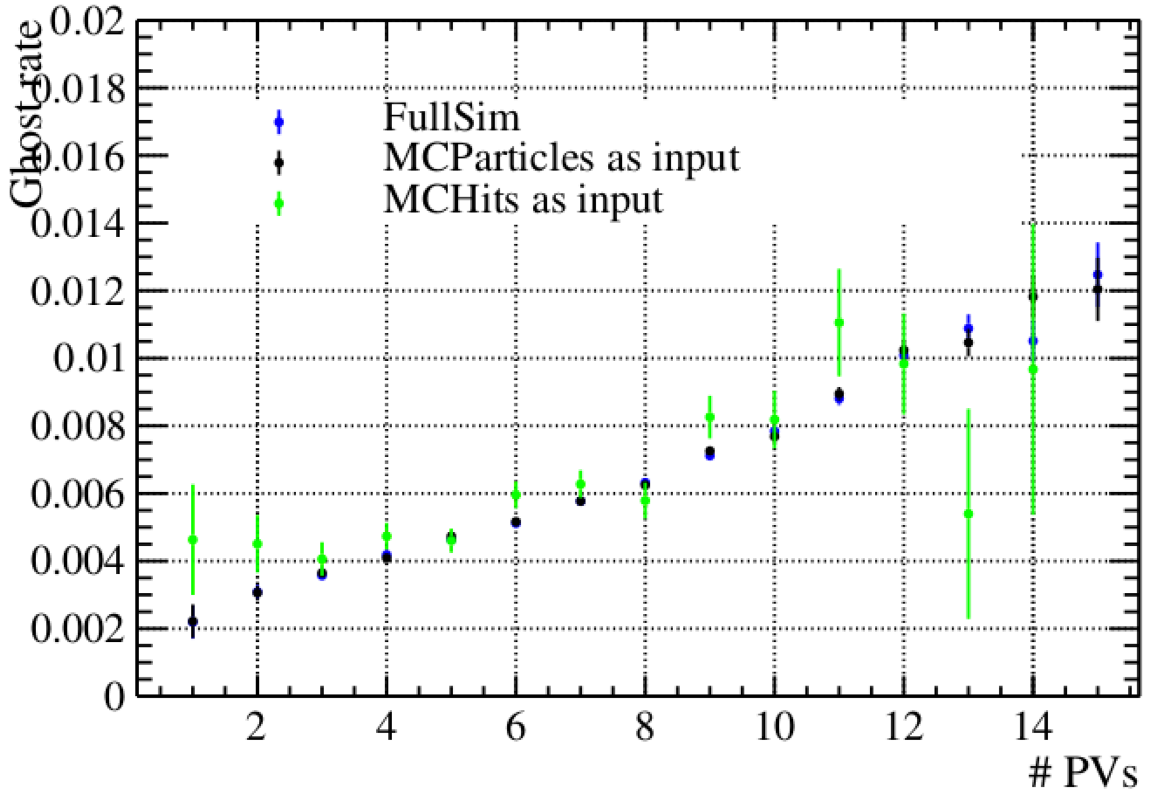
- **MC hits** from DD4Hep as input


~~Detector geometry + sensor response~~
Hit clustering + Track reconstruction
Performance studies

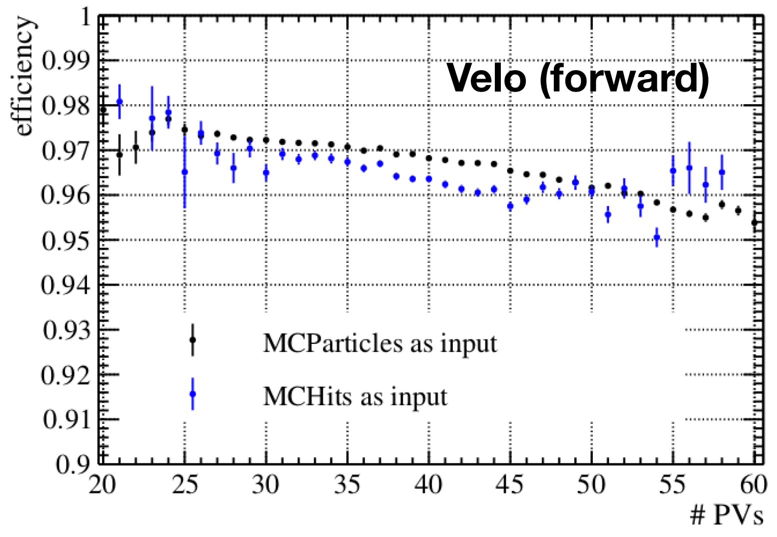
MC hits as input - validation

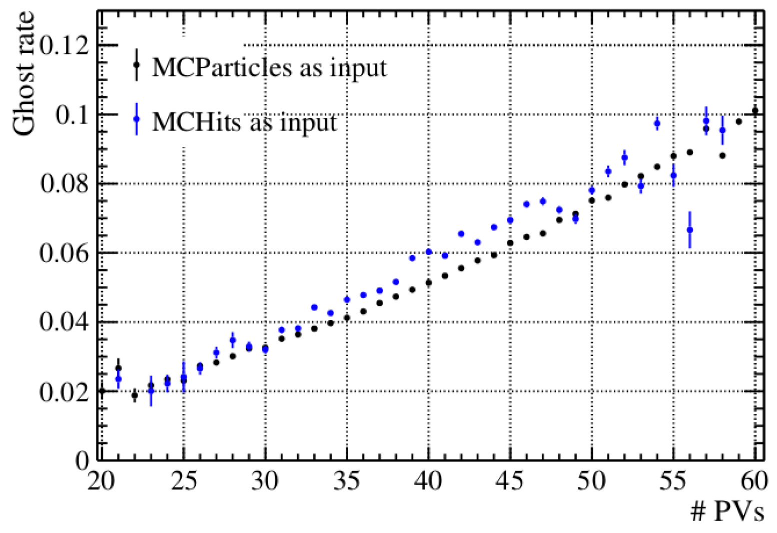
- Efficiency and ghost probability in U1





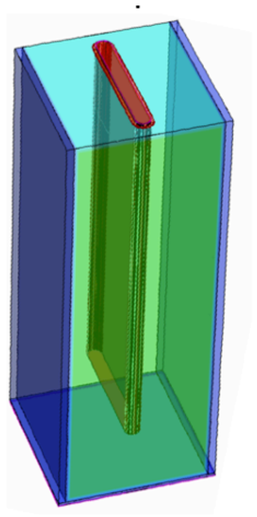
- U2 scenario in agreement between the two methods





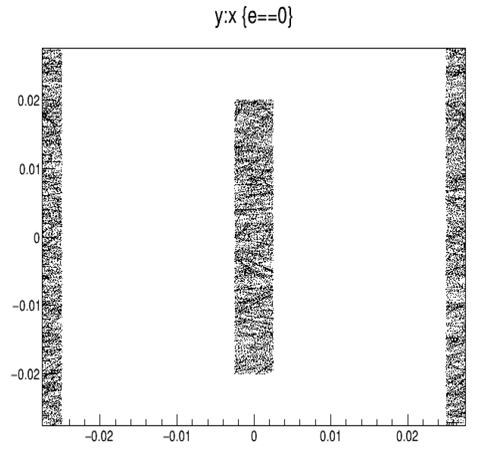
TIMESPOT sensor (TS) - simulation

- Deposited charge taken from MCHit. Rescaled considering the smaller thickness of TS w.r.t. VP sensor.
  (Multiple scattering effects are then overestimated by the simulation)



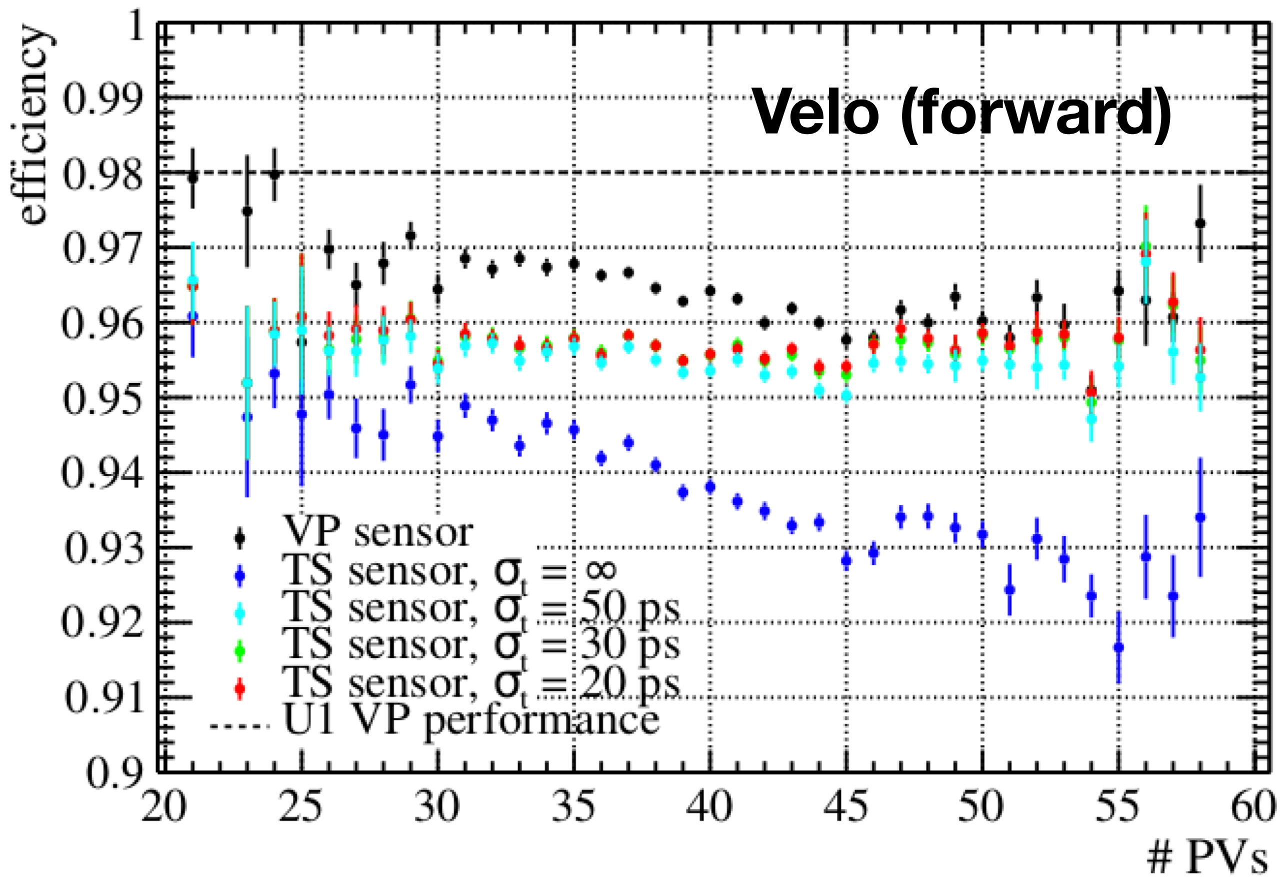
- Charge distributed on the sensor pixels and digitized.

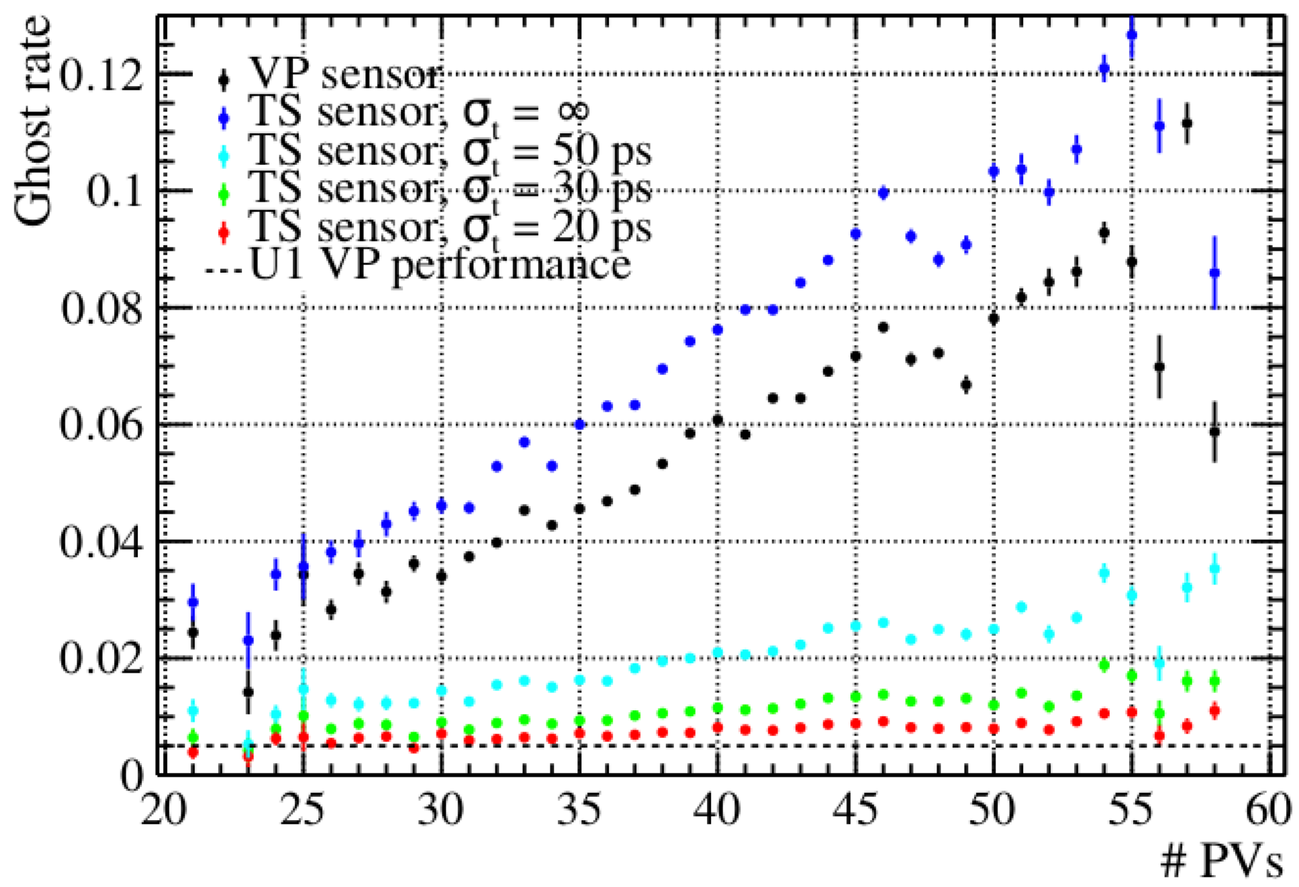
- Sensor parameters:
  - trench = 5x40 + 5x55 mum2 in XY (vs none in VP)
  - depth = 150 mum (vs 200 mum in VP)
  - noise = (guess)300 e- (vs 130e- in VP)
  - threshold = (guess)1500 e- (vs 1000e- in VP)
  - No diffusion in XY (vs diffusion in Z in VP)
  - time resolution = 20,30,50 ps (vs no time info in VP)



Upgrade I	εVELO(%)	PGHOST(%)
VP No timing	98.0	0.5

Performances

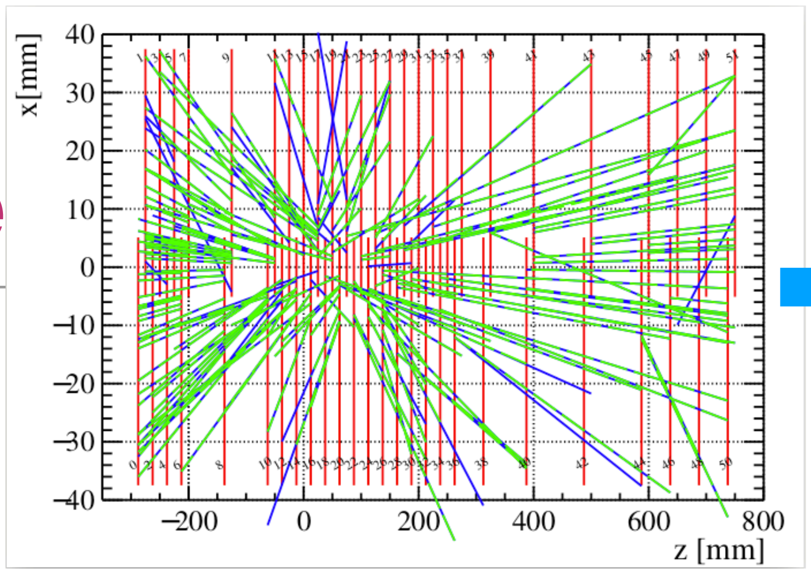


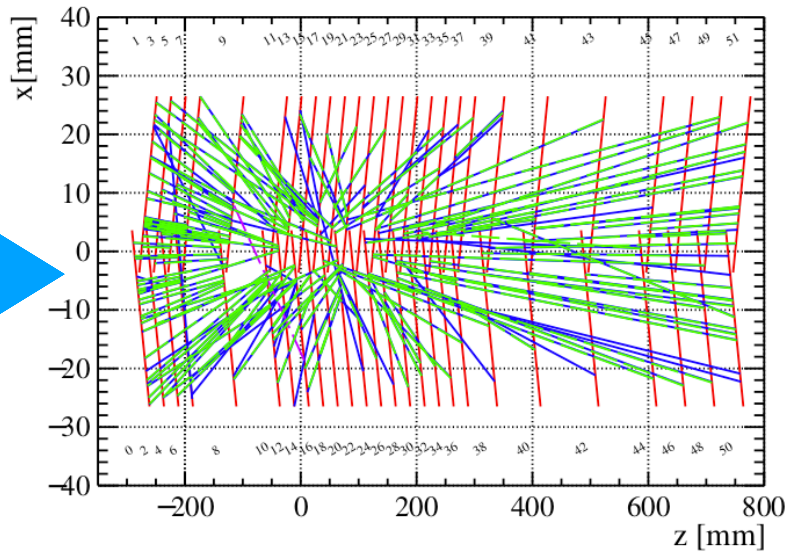


- Targeting Upgrade I VP performances
  Efficiency lower than U1
  Ghostrate comparable with U1
  → exploring different tilting angles
  to improve efficiency

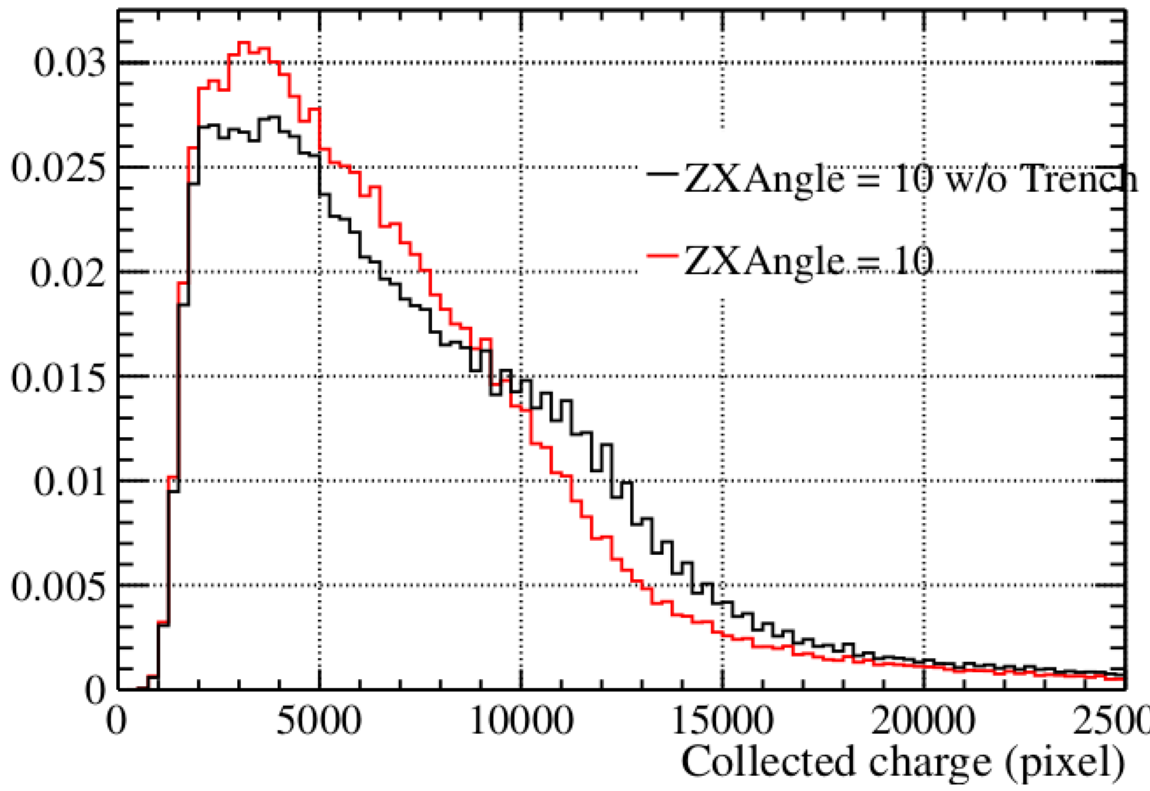
Upgrade II	εVELO(%)	PGHOST(%)
TIMESPOT σ_t = 20 ps	95.6	0.7
TIMESPOT σ_t = 30 ps	95.6	1.1
TIMESPOT σ_t = 50 ps	95.4	2.0
VP No Timing	96.4	5.6

The tilting angle

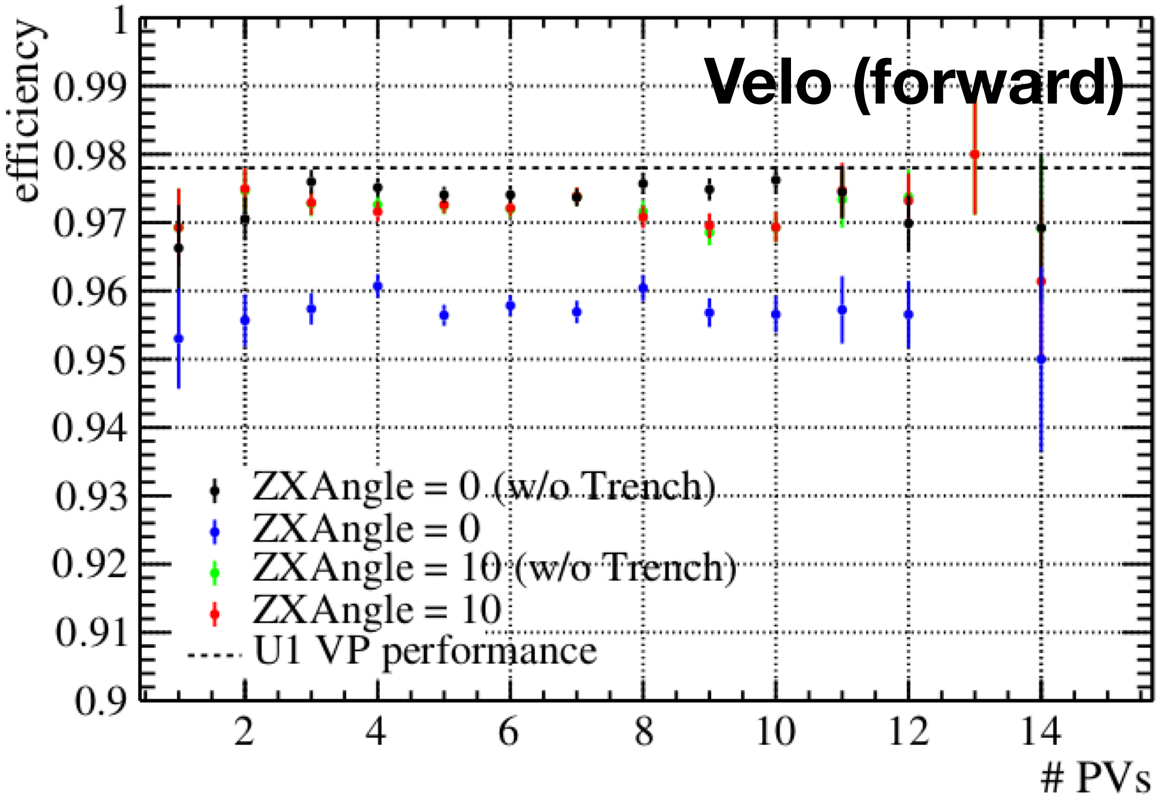




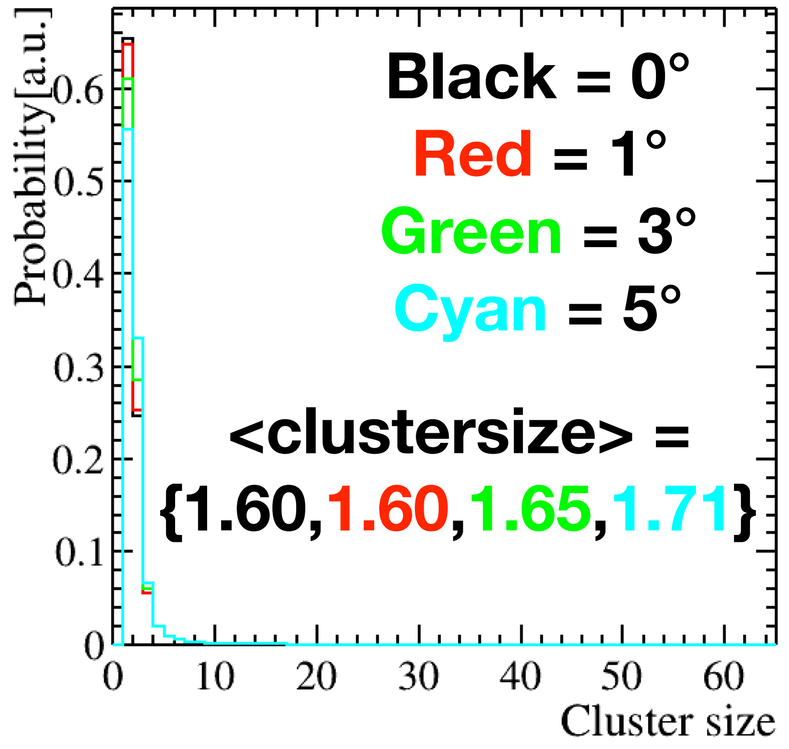
- The tilting angle is exploited to recover the loss in efficiency arising from the presence of the trench. Eg. at 10° in U1 scenario this loss is completely recovered.



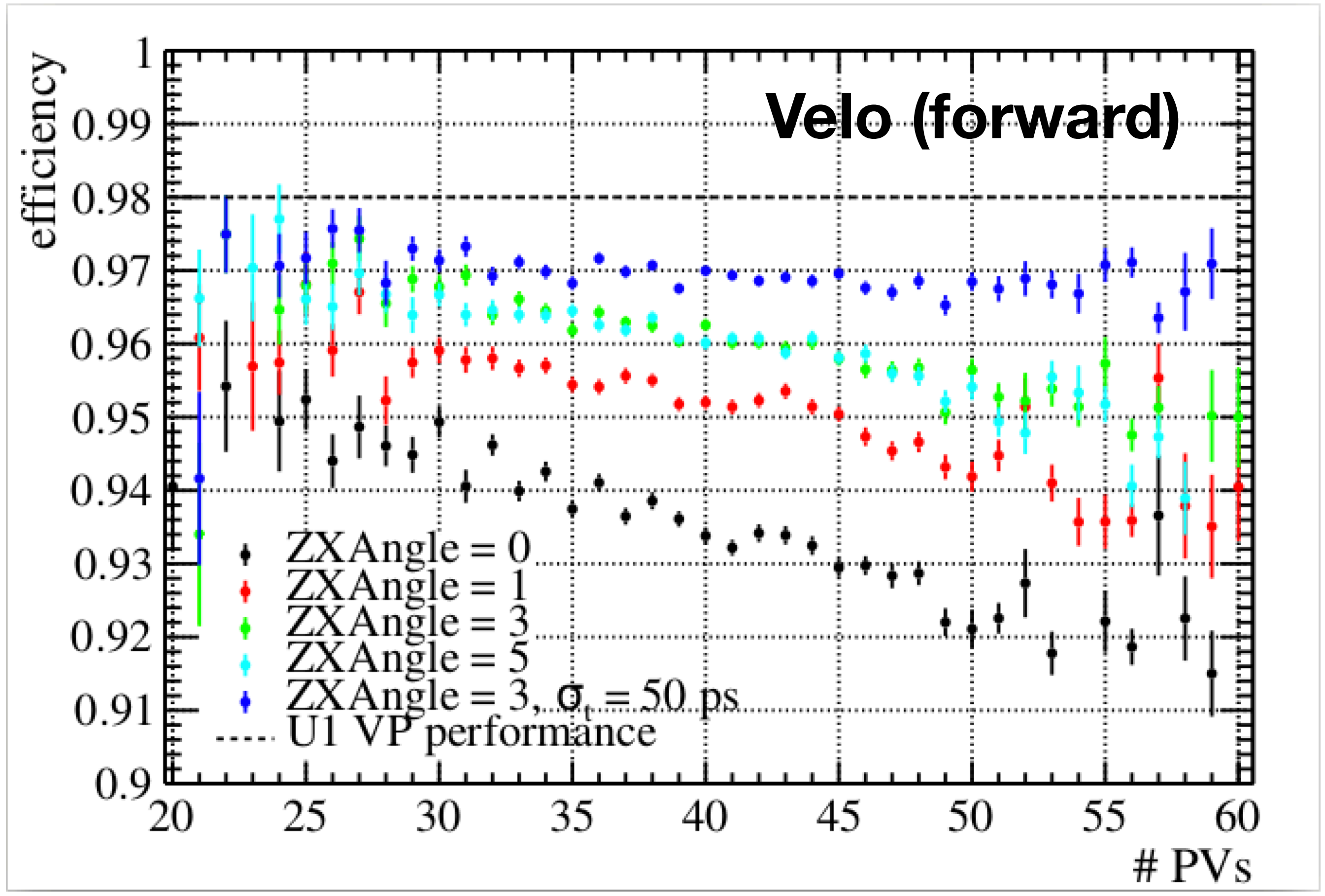
In U1, TS sensor is less efficient than VP

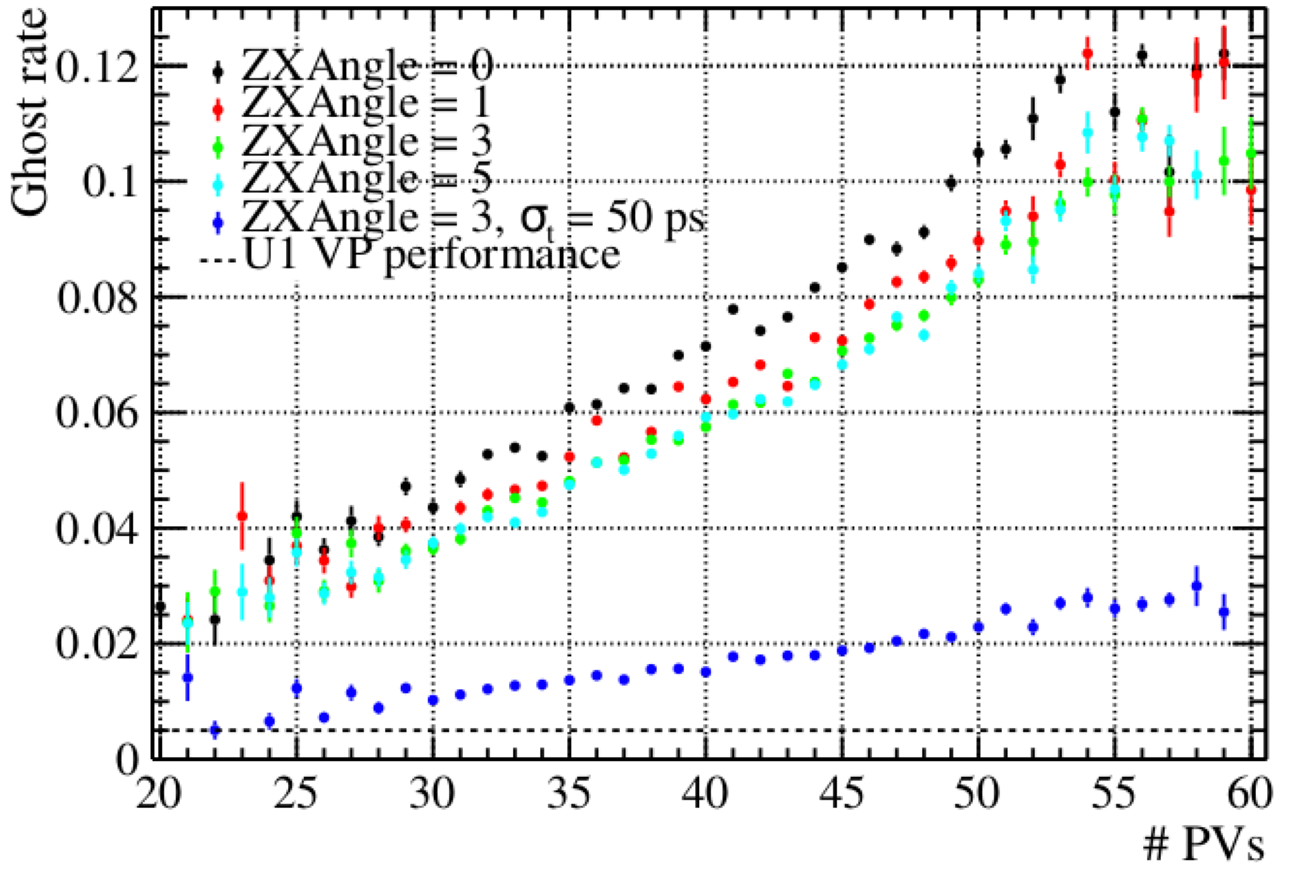


Exploring different tilting angles in Upgrade 2



- Efficiency and ghost probability for 1°, 3° and 5° and considering a realistic pixel resolution of 50 ps.





- Huge improvements in efficiency. However U1 performances are not completely recovered in U2 both for efficiency and ghost probability.

Moving to a stub based algorithm

- Marco stuff…

Conclusions

- We studied the performances for a future LHCb VELO using the TIMESPOT sensor.

- As a *preliminary* study, a new detector using the TS sensor in the same geometry and tracking algorithm of VP with a realistic time resolution of 50 ps and a tilting angle of 3° could bring to performances comparable with the current one (expected for U1).

- Space for improvements:
  - improve tilting angle
  - change tracking algorithm (*stubs* based)
  - test new geometries with *doublets of modules*

Thanks!

Serena Maccolini

User guide reference

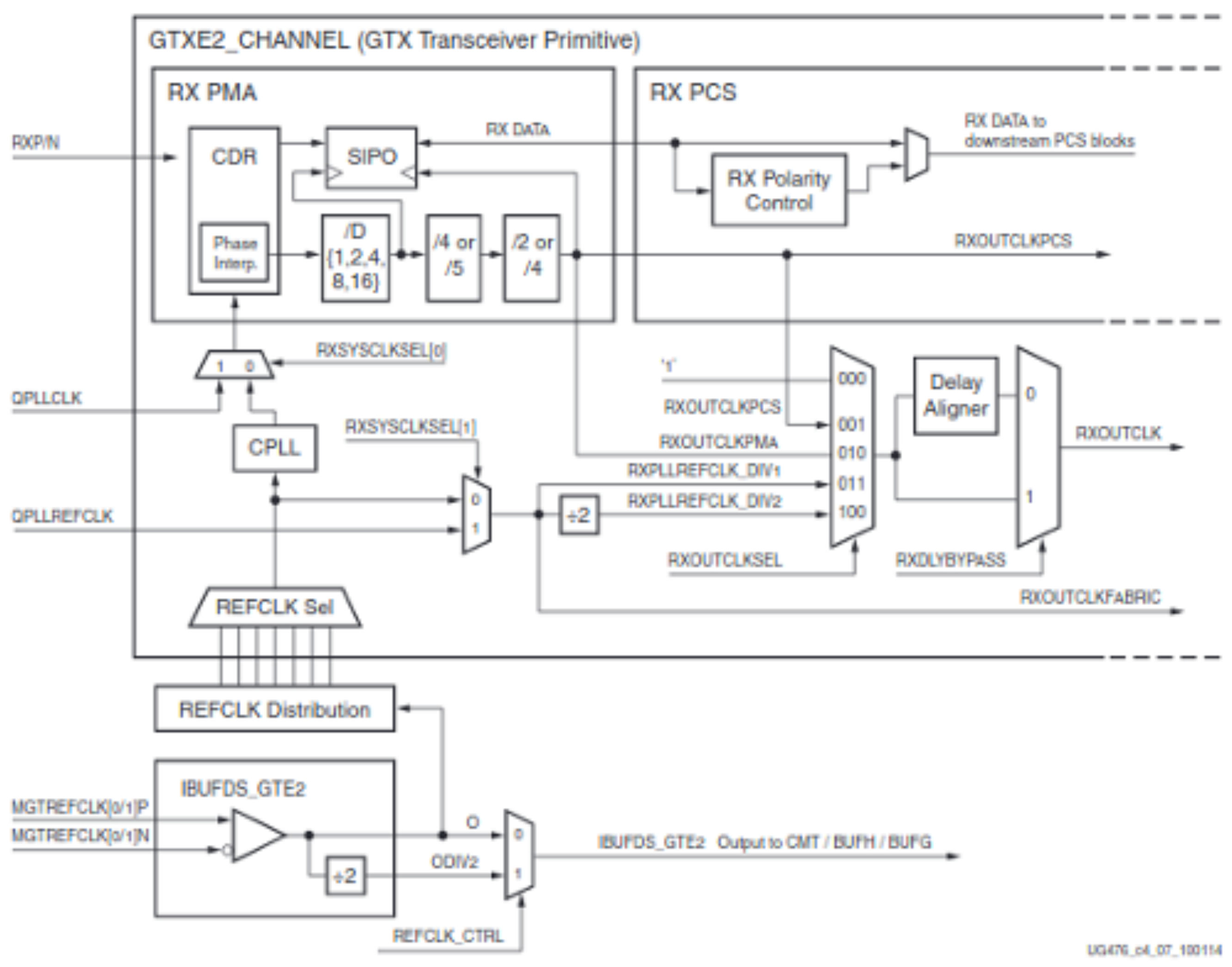


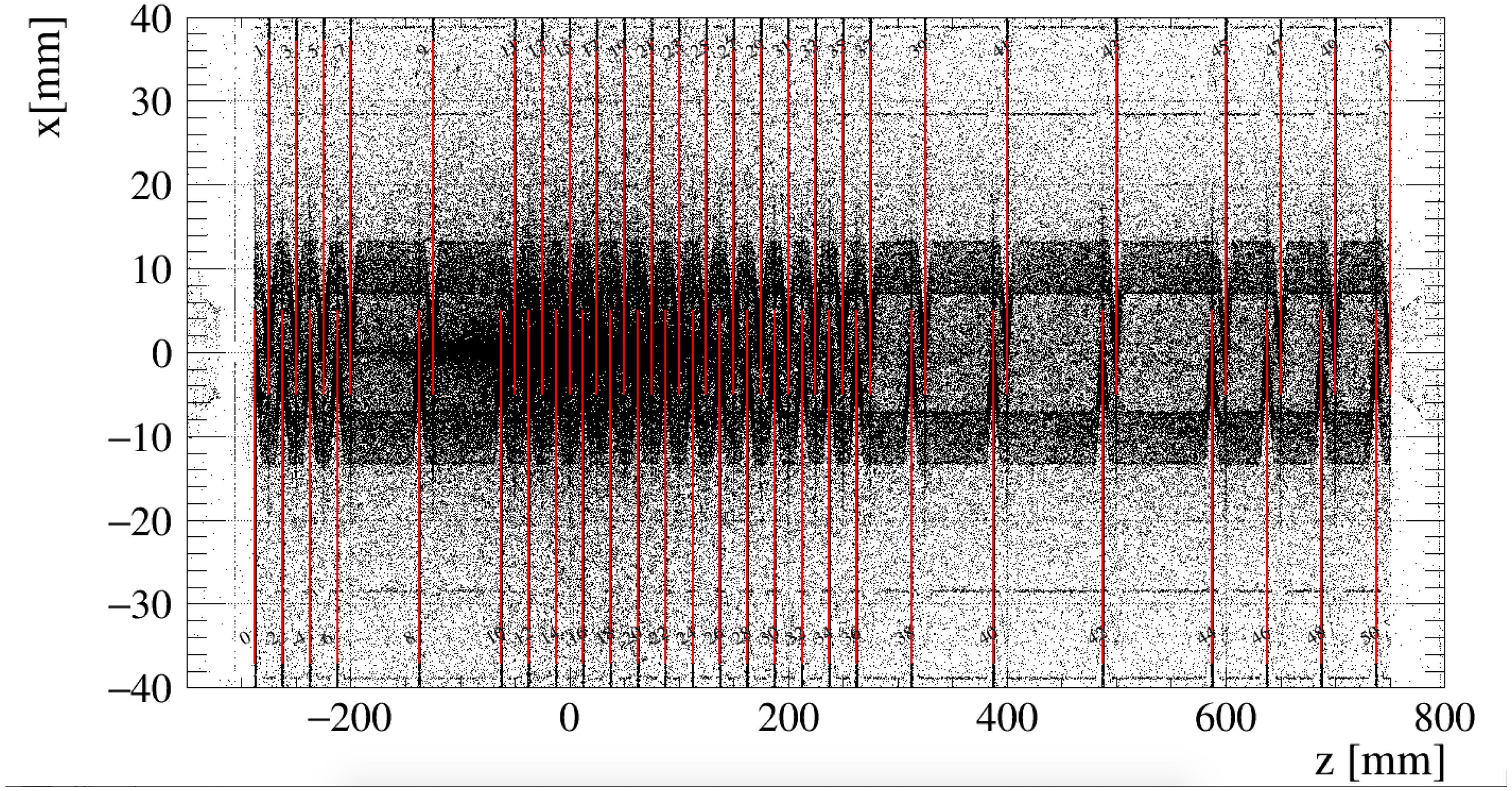
Figure 4-23: **RX Serial and Parallel Clock Divider**

- From Xilinx User Guide 476 , pag 210

How the FastSim works?

Check geometry consistency full vs. fast (z)

- *Fast* simulation design with *full* simulation radiography (VP geometry seen by secondary collisions)



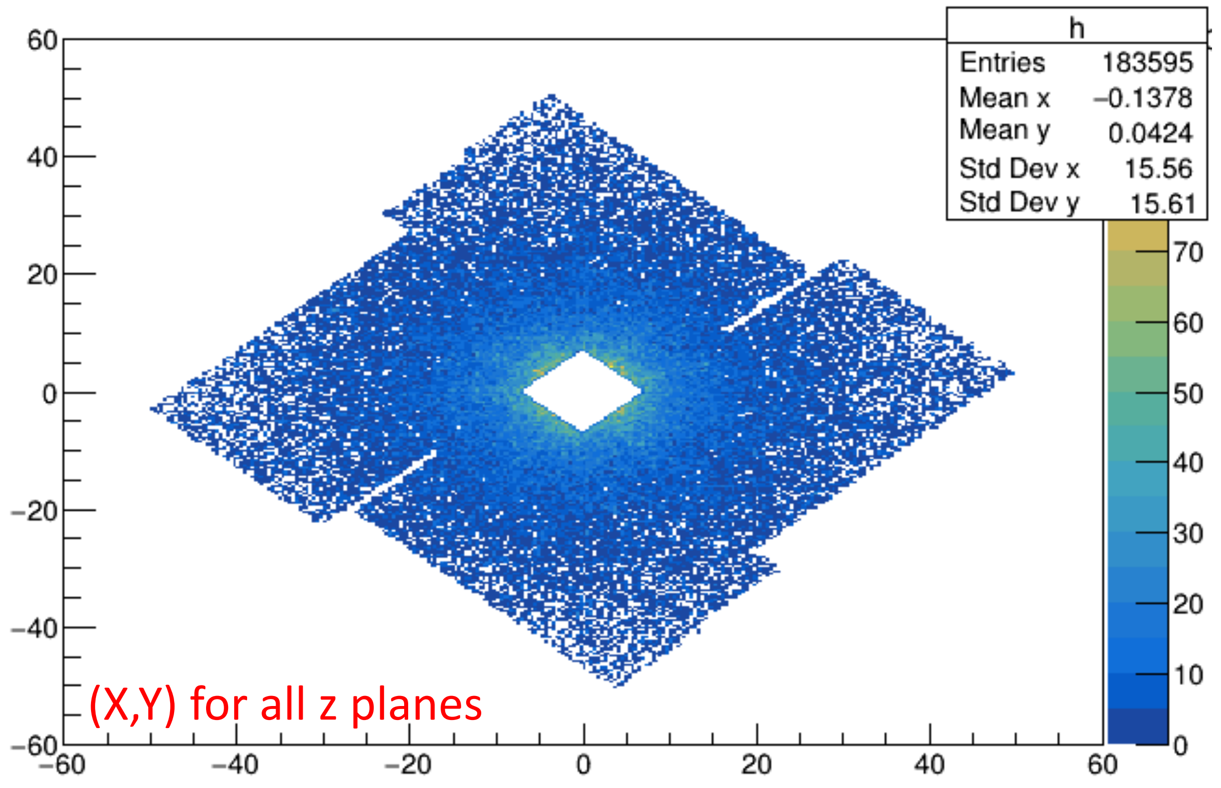
Z station in fast simulation overlaps with full simulation

- Z geometry is well reproduced.
  No z-shifts between sensors on same module implemented.

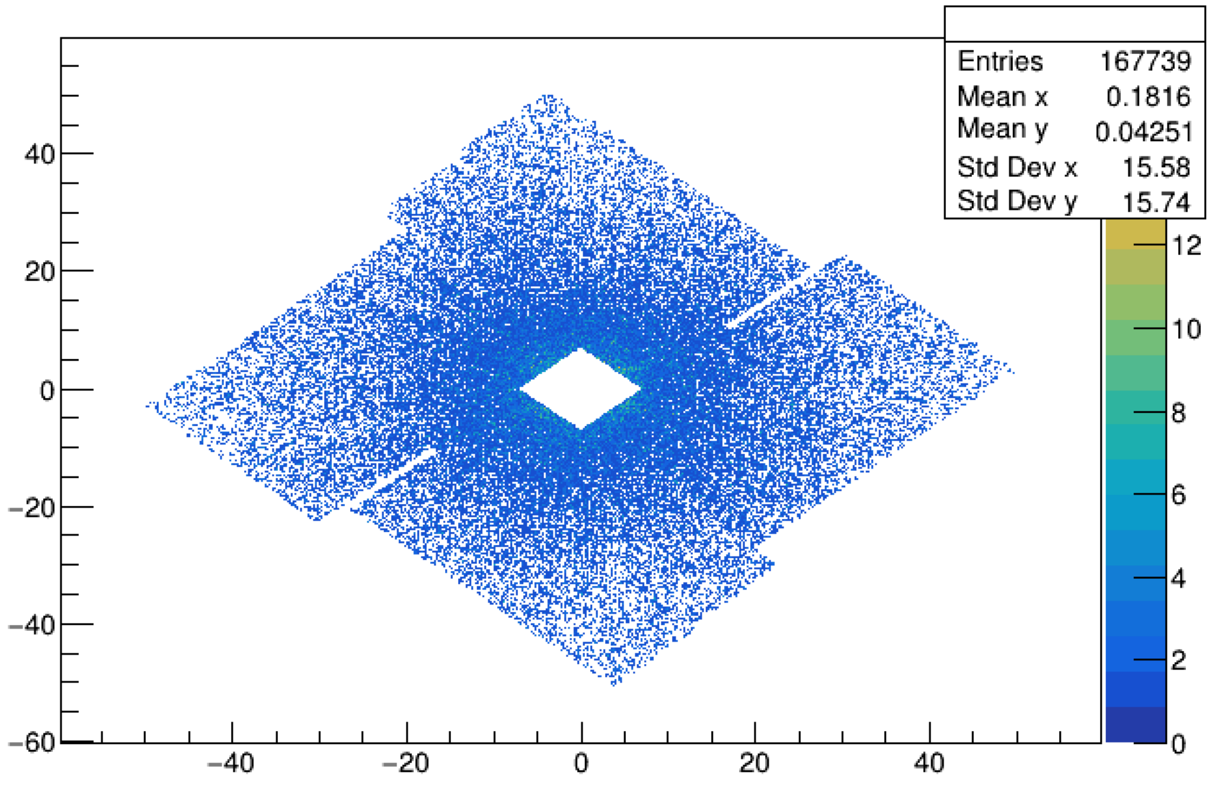
Check geometry consistency full vs. fast (x,y)

- Active pixels

Full simulation



Fast simulation

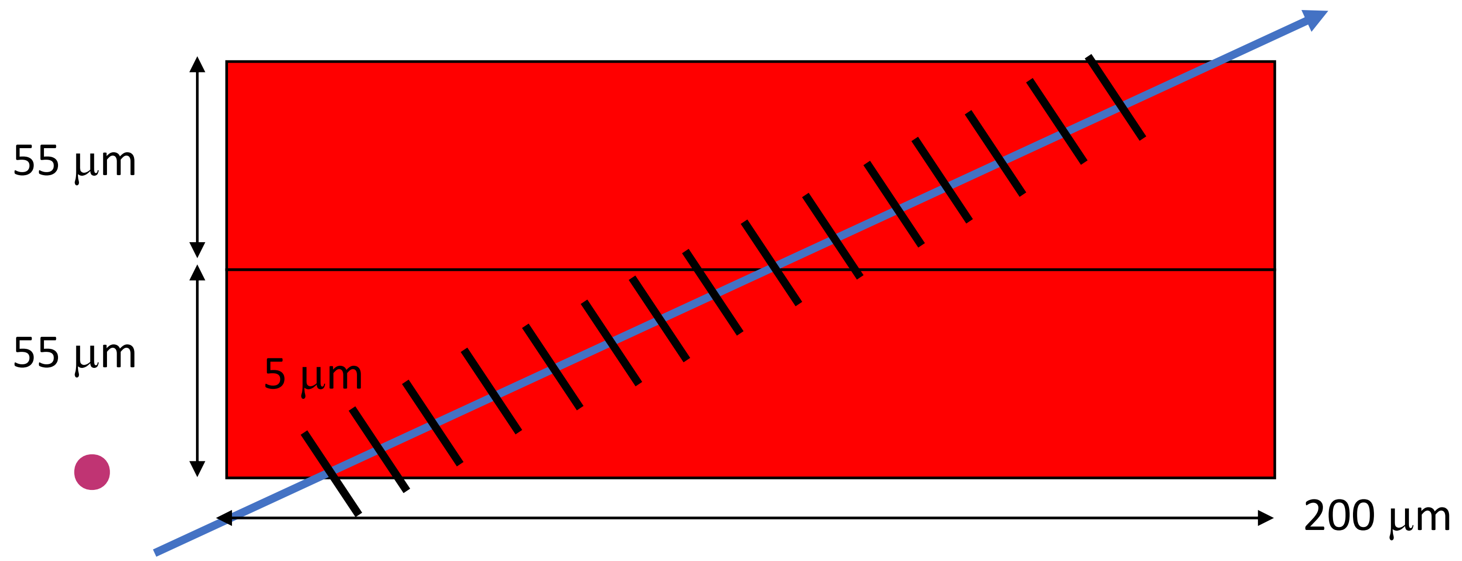


- (x,y) geometry is well reproduced. Visible space between the sensors implemented. Variable pitch size of the pixels between the chips not implemented.

Charge deposit in the fast simulation

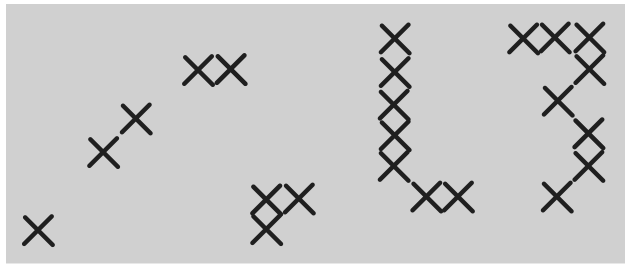
Similar to what is implemented in LHCb software (*VPDepositCreator*):

- The path of each particle in silicon is divided in *intervals* of 5 μm
- For each interval the number of *electrons* released is randomly extracted from a gaussian distribution (mean 350) + 1/n^2 tail
- The total number of electron is forced to be equal to **deposited energy**/3.6 eV
- *Dispersion* of charge in neighbors pixels is also simulated
-

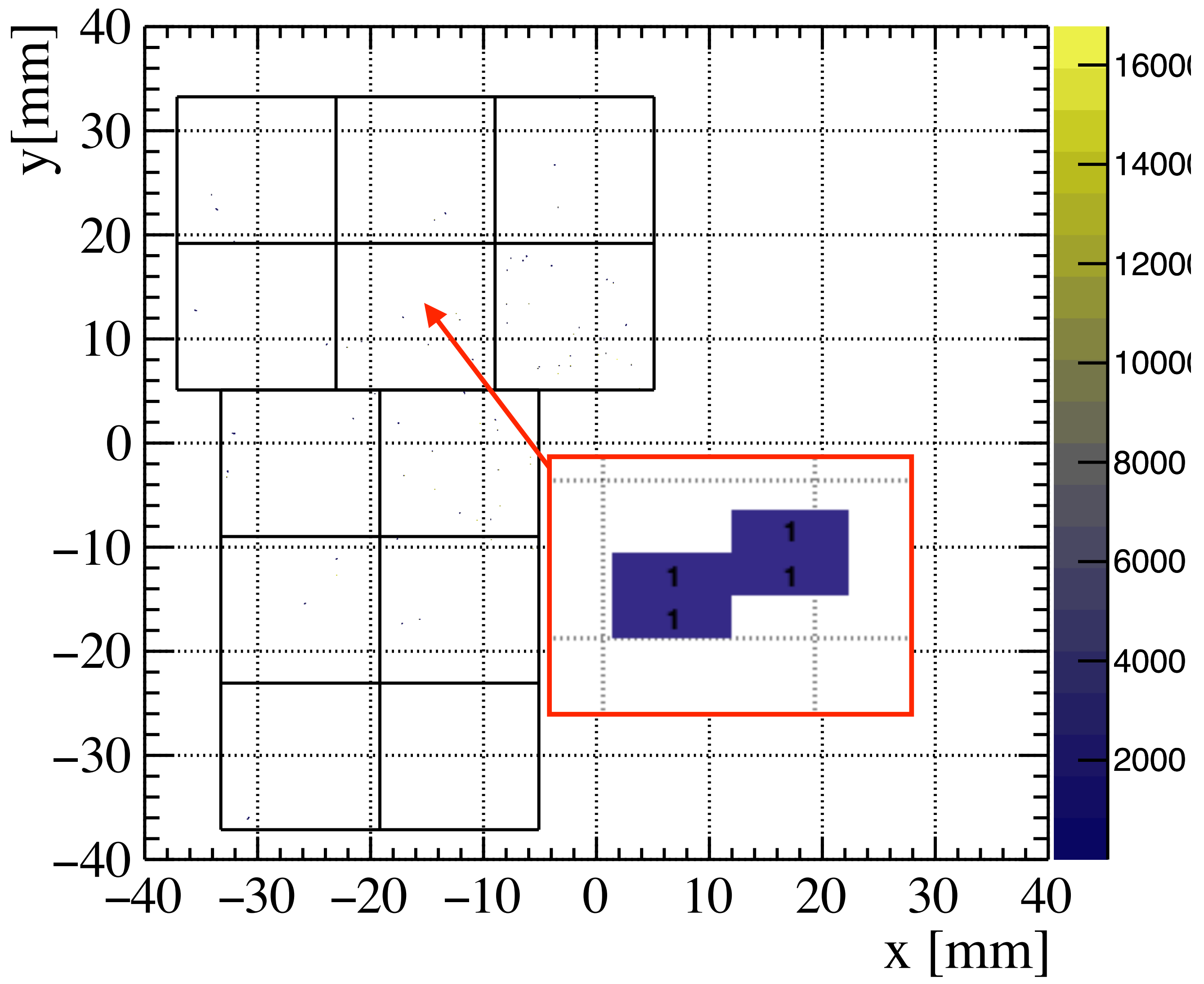


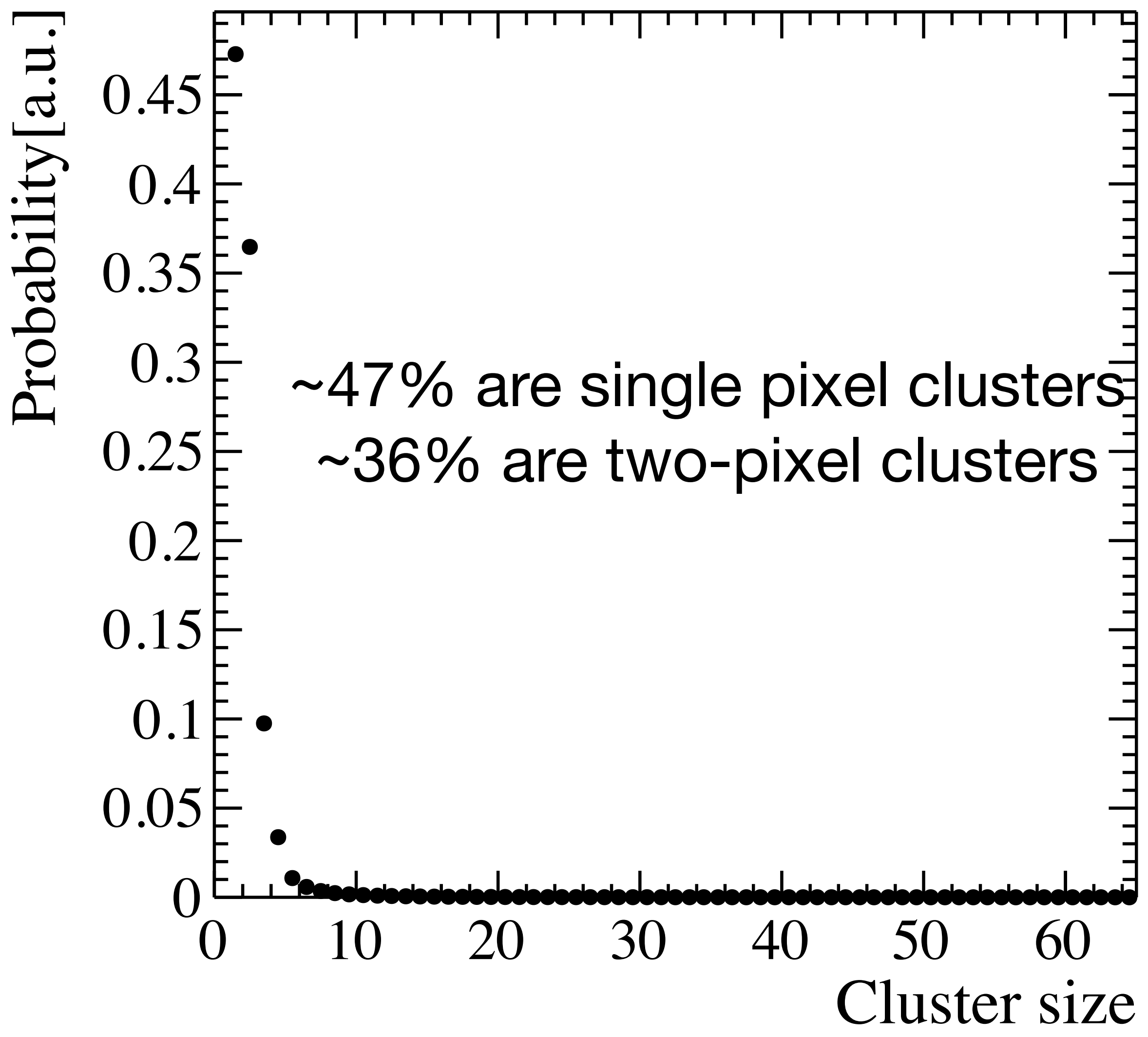
Deposited energy: randomly obtained from Landau distribution (CERNLIB/RANLAN algorithm) using as input the most probable value Δ_p and the width parameter ξ given by PDG2018/33.12 formula

Clustering



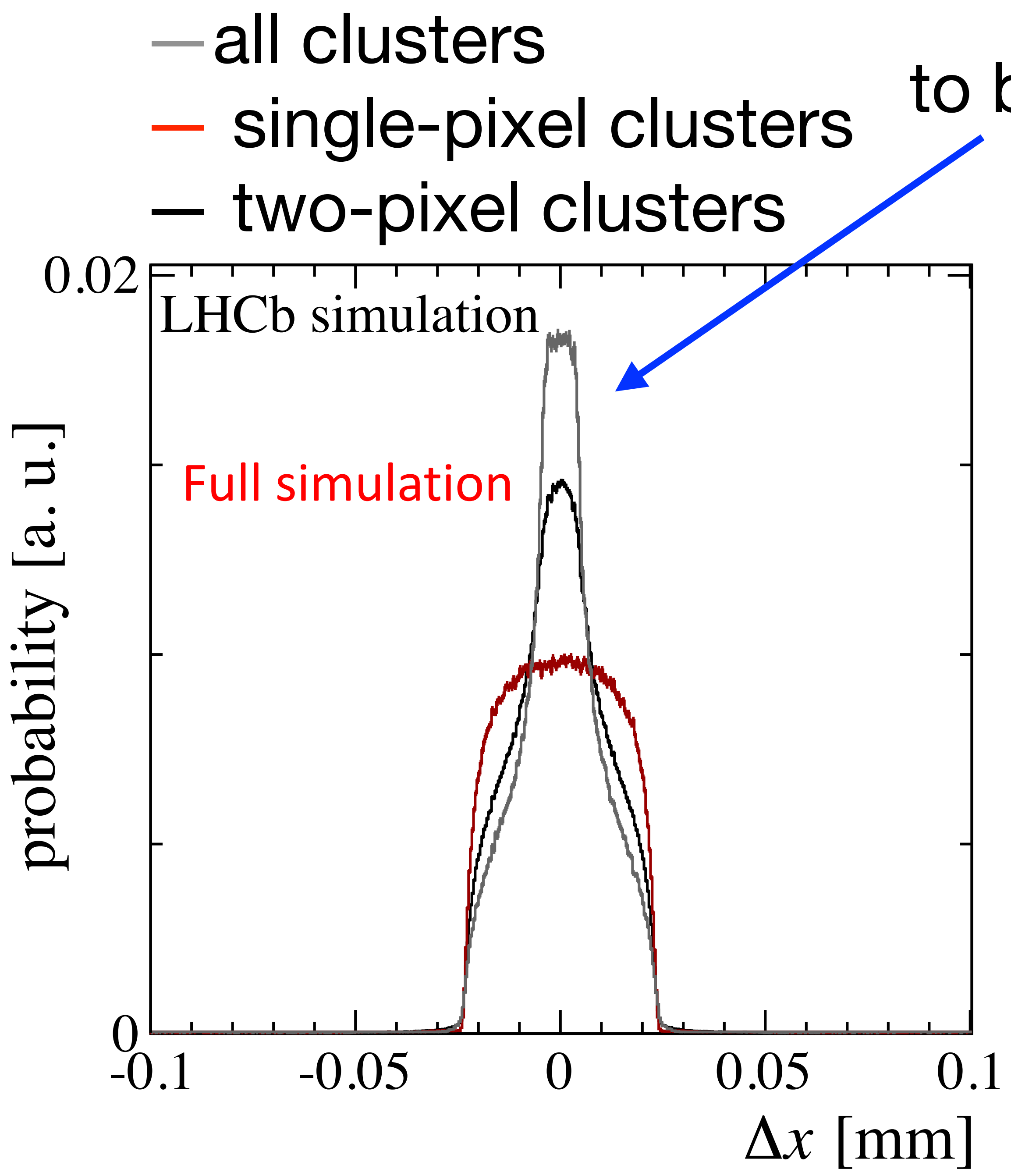
- A cluster is defined as an isolated group of *eight-way connected* active pixels

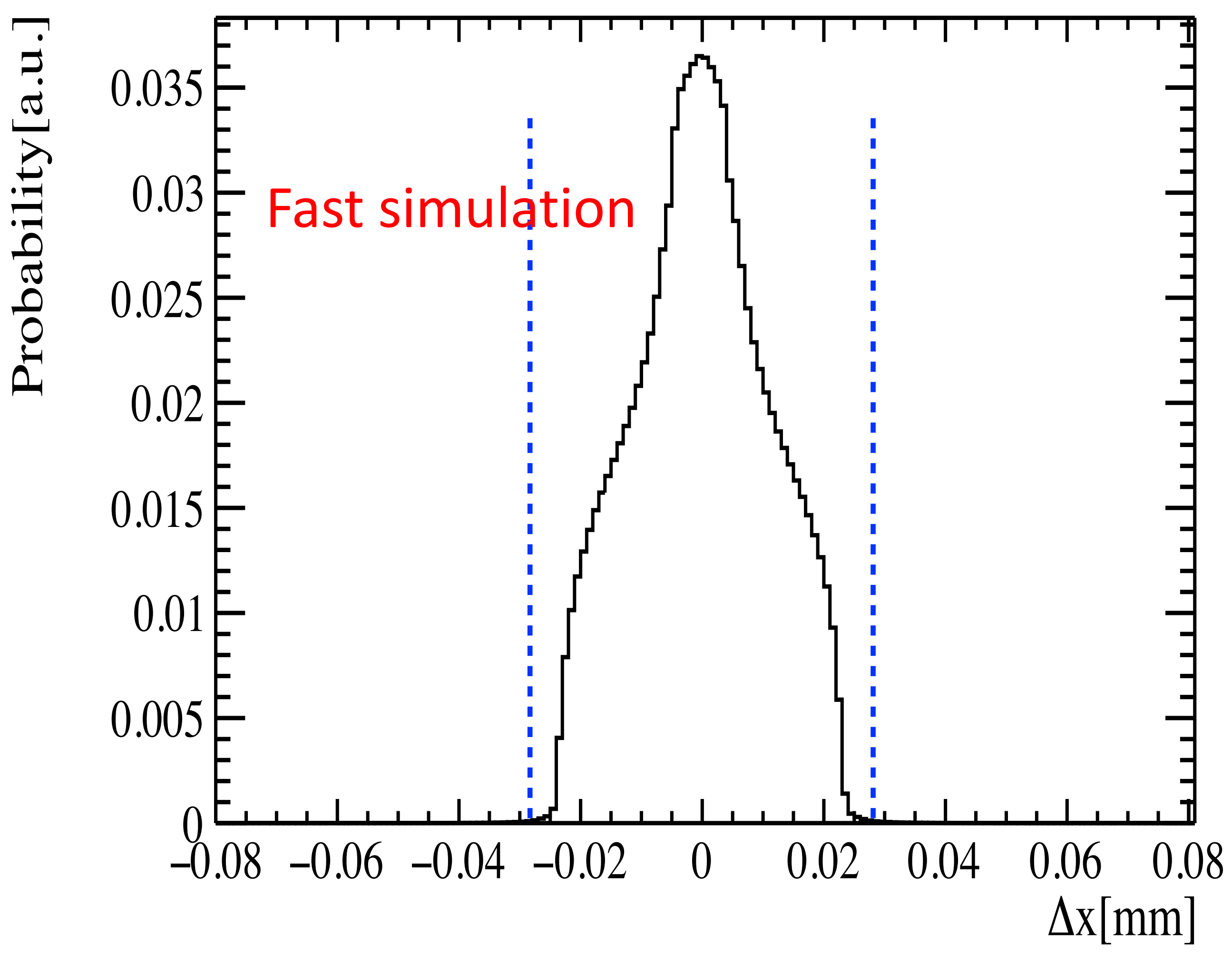




Hit resolution

- Clustering improves resolution





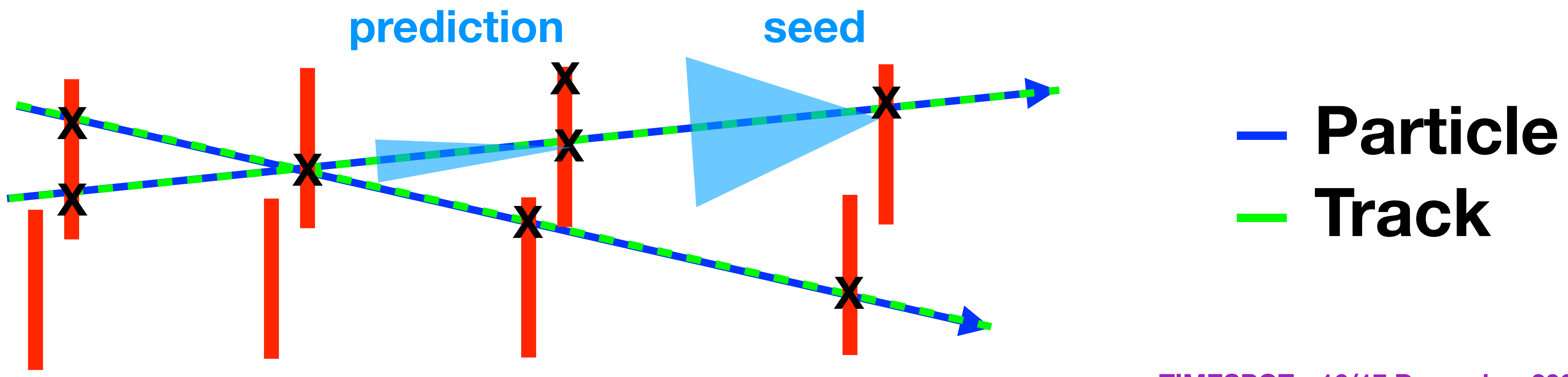
Equivalent to LHCb::Pr::Velo::PixelTracking/PrPixelTrackingFast

Reconstruction strategy

- Create a **seed** of track:
  take a pair of hits belonging to same-side (left or right) modules with $|dx/dz| < 0.4$ and $|dy/dz| < 0.4$

- **Predict** the position of the hit of the track in the next same-side module. Look in a window around the predicted hit, take the hit closest to the prediction and add it to the seed to form a track.
  If no hit is found go search to the next opposite-side module.

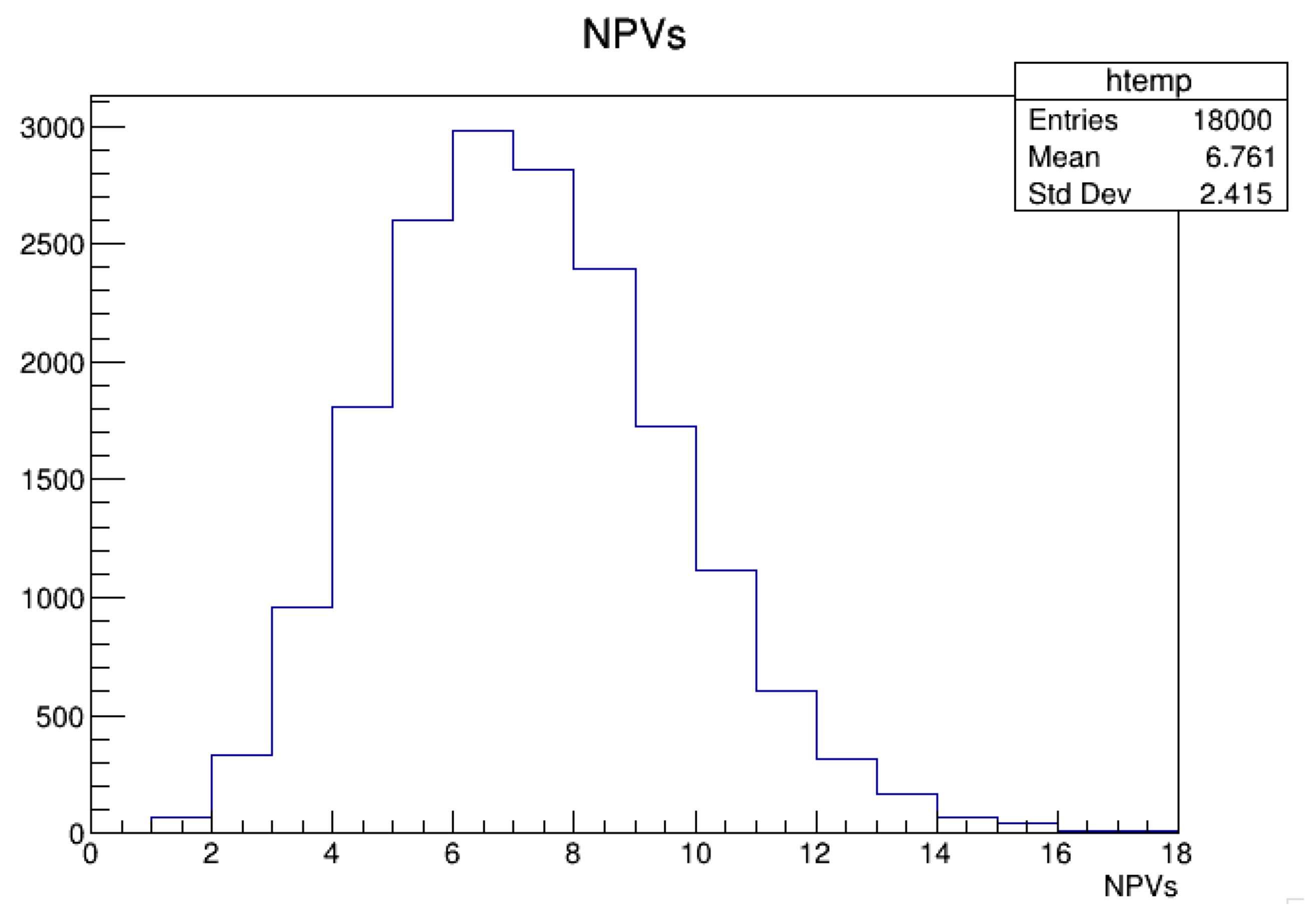
- The hits are *exclusive*: once a track is found, its hits cannot be used for other tracks

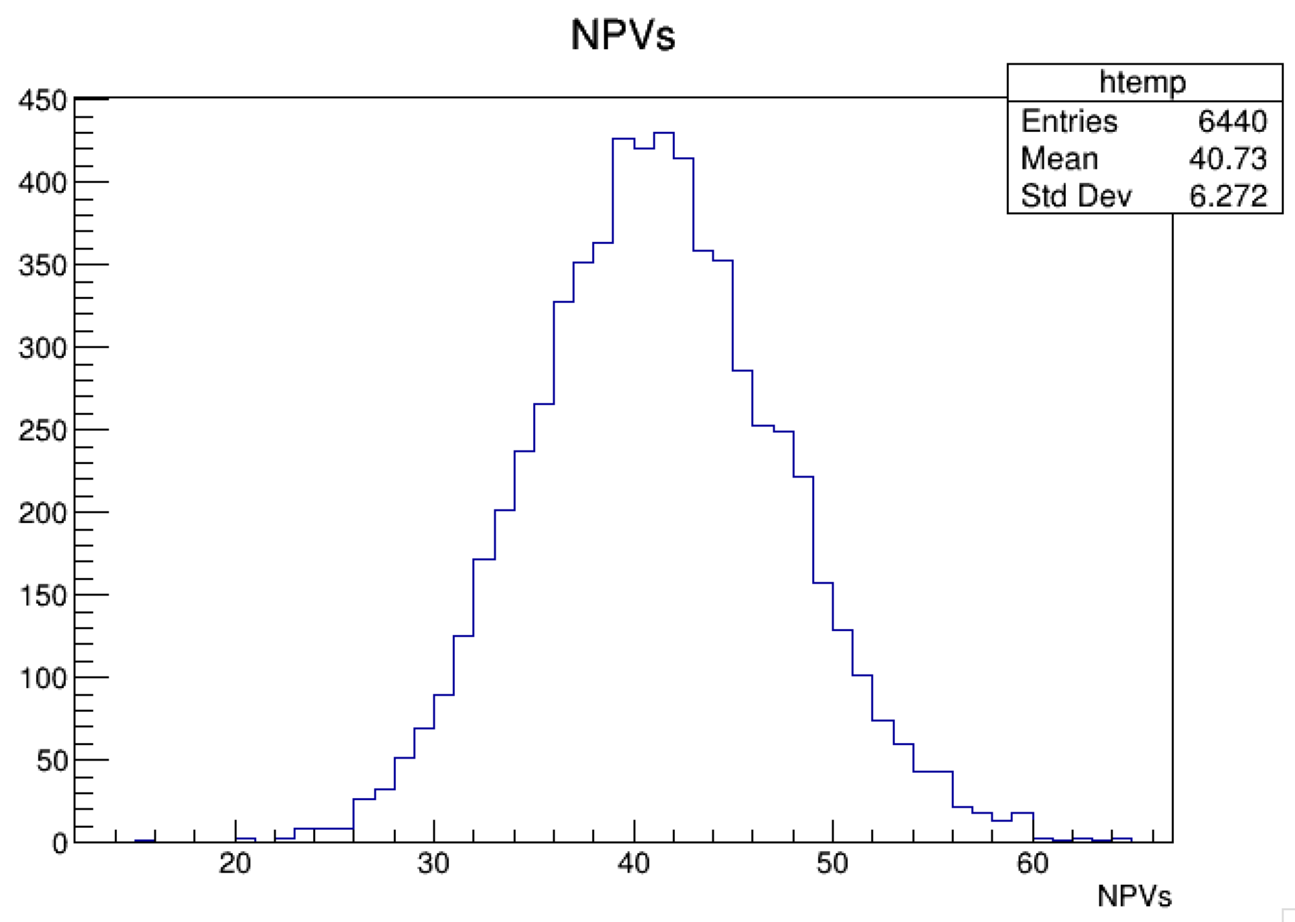


Upgrade-II

From U1 to U2

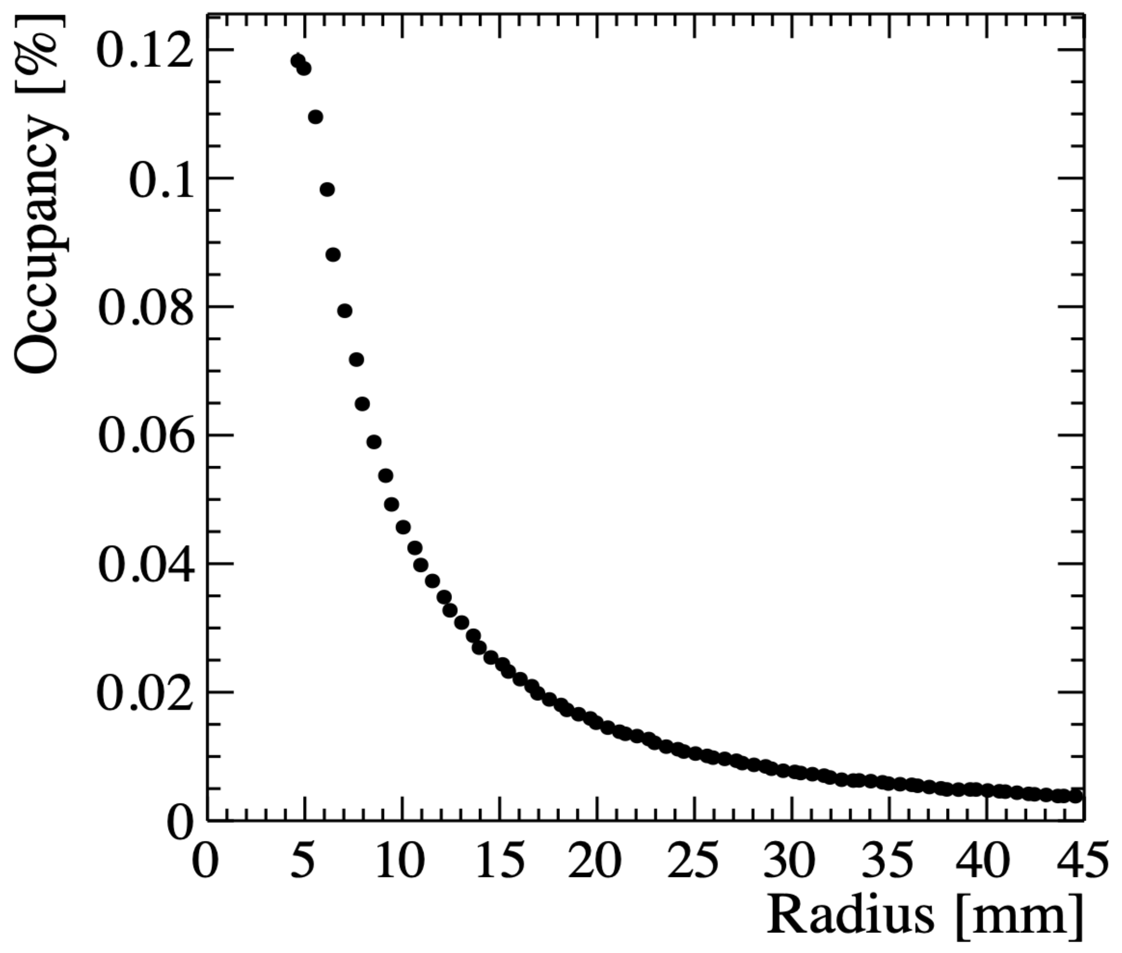
- From an average of 7 visible PVs per event to 41 PVs

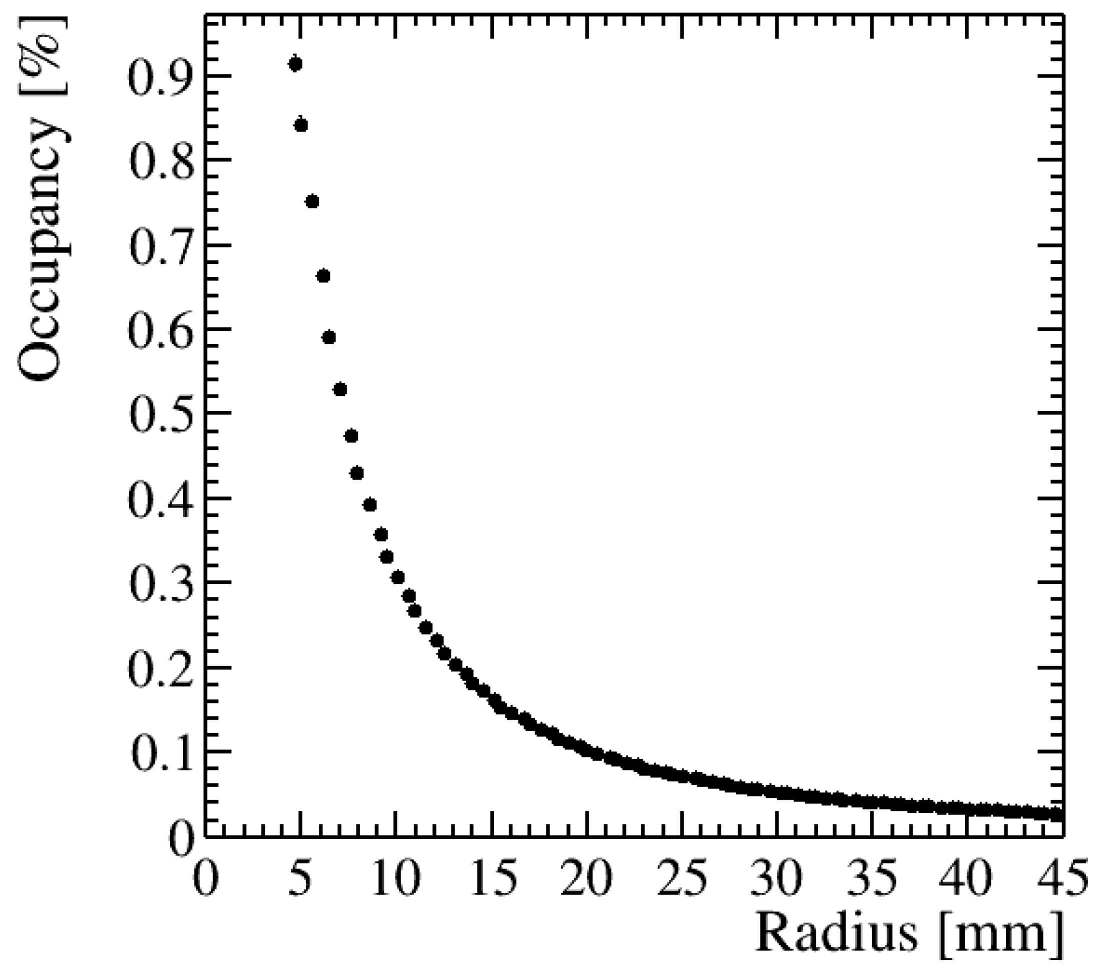




From U1 to U2

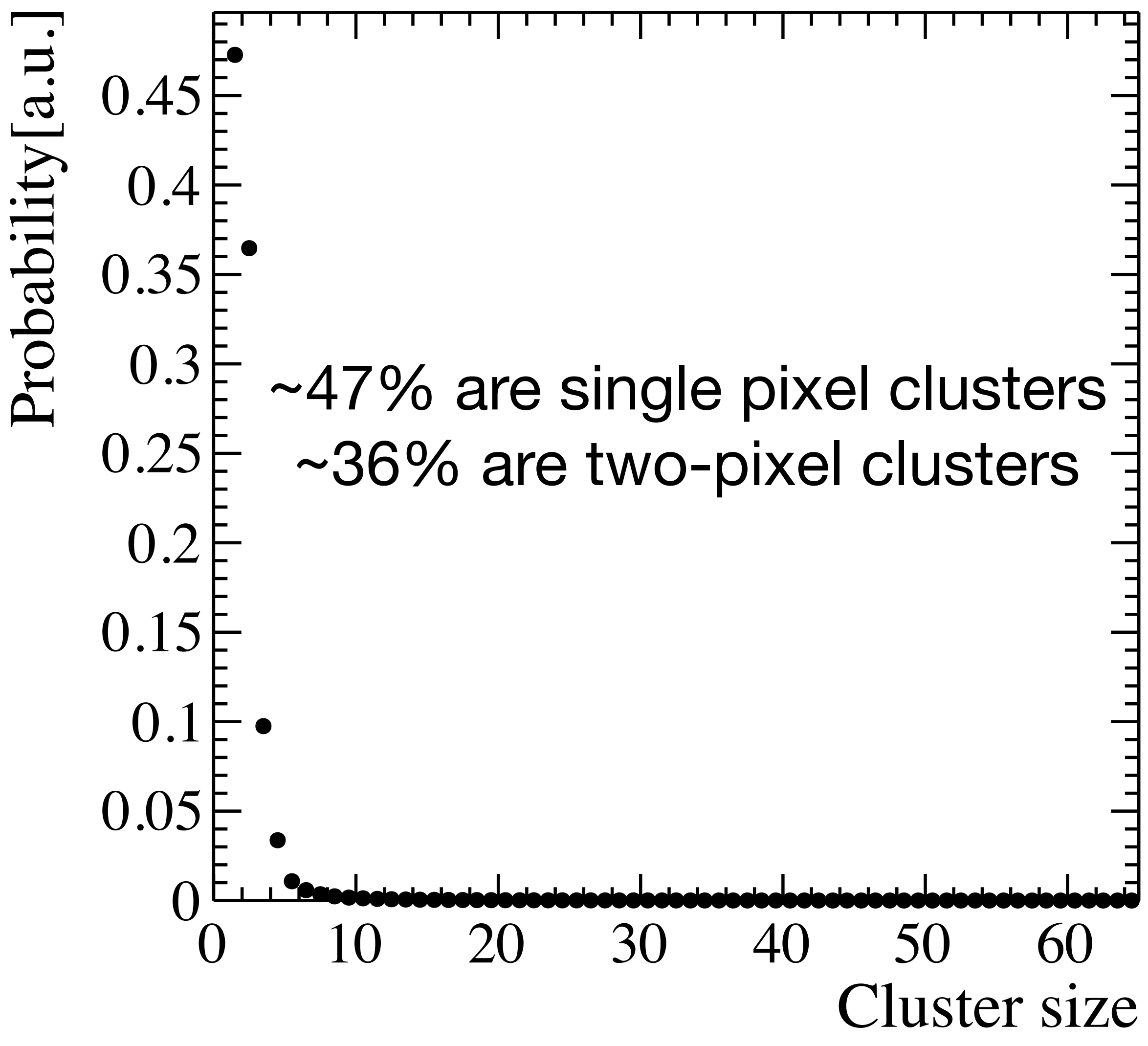
- Occupancy increased of a factor ~10



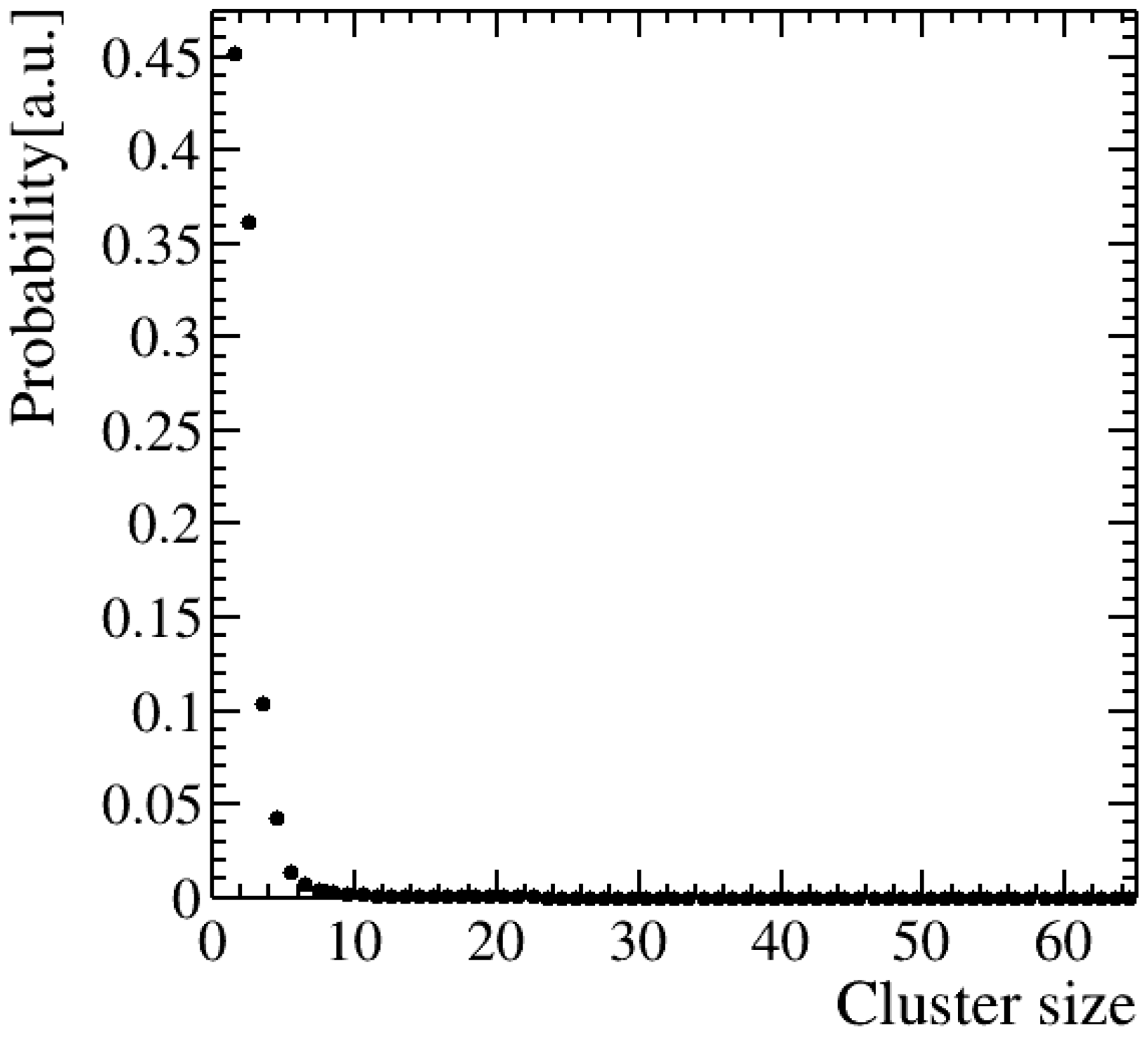


From U1 to U2

- Cluster size essentially unchanged



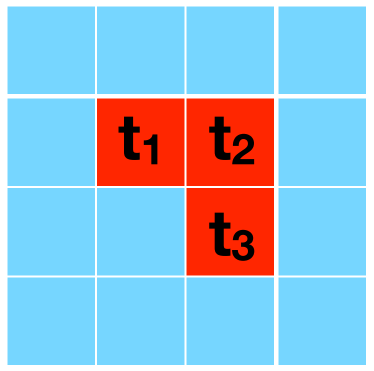
~47% are single pixel clusters
~36% are two-pixel clusters



Add timing

Introducing time

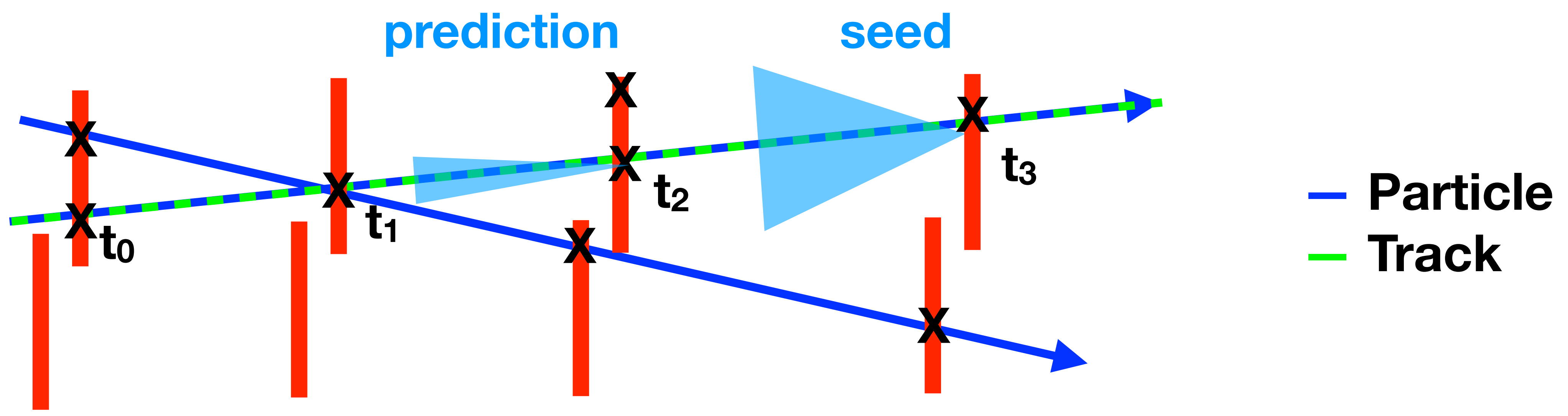
- The timing of Primary Vertices is non simulated in Gauss, i.e. $t_{PV} = 0$
- t_{PV} extracted from a Gaussian of sigma = 212 *ps*
- OriginVertices time is given by Geant4 and corrected by t_{PV}
- The true time t_{true} of a particle is propagated along his path trought the detector modules
- The *time* of each charge deposit is smeared considering certain resolution $\boldsymbol{\sigma_t}$ ={0,10-60} ps
- Approximation: in case more than one particle hits the pixel (<3% of the active pixels), its time is the one assigned by the first particle



$t_i \sim G(t_{true}, \sigma_t)$

Timing in reconstruction

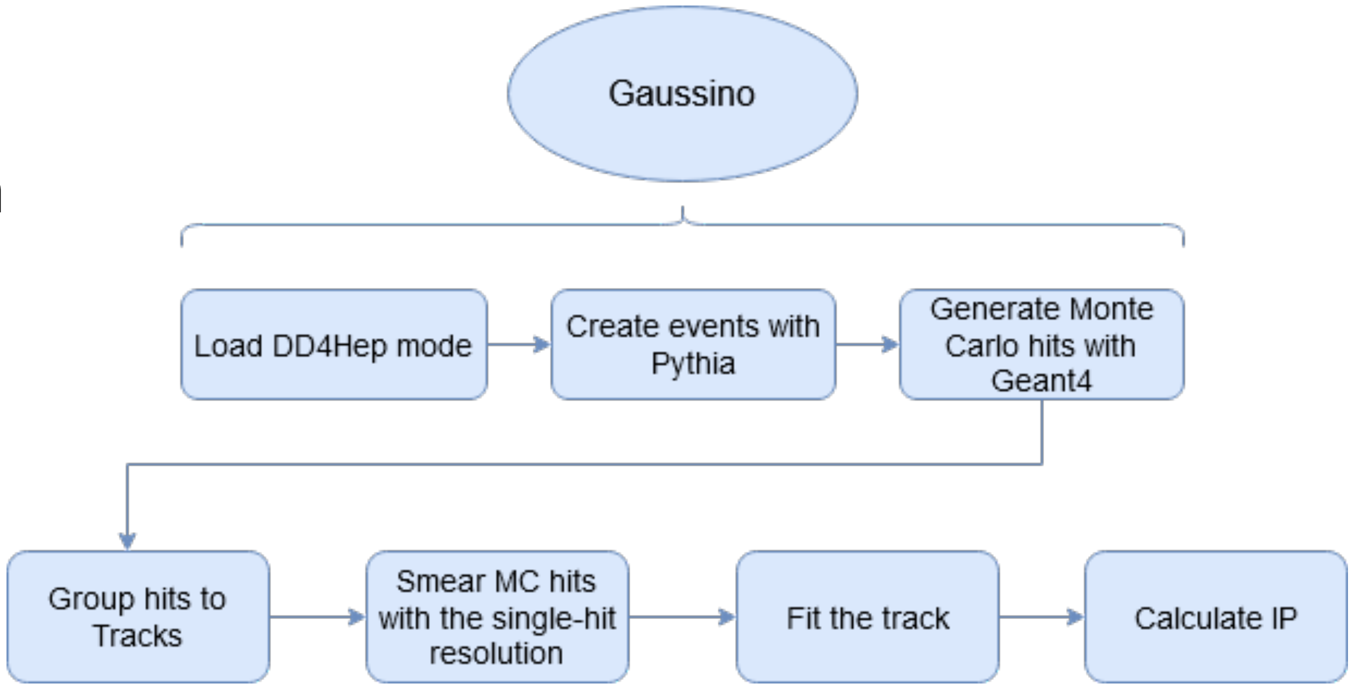
- **Clustering:**
  A cluster is identified with a group of neighboring pixels with recorded times within $3\sigma_t$ between each others.
  The time of the hit is the average of the pixel times.
- **Tracking:**
  Considering v = c for the particles.
  Assuming a certain direction of the particle (forward/backward) the time of the hit in the next module is predicted.
  Only hit with $|t_{pred} - t_{reco}| < 3\sigma_t$ are considered in track building.

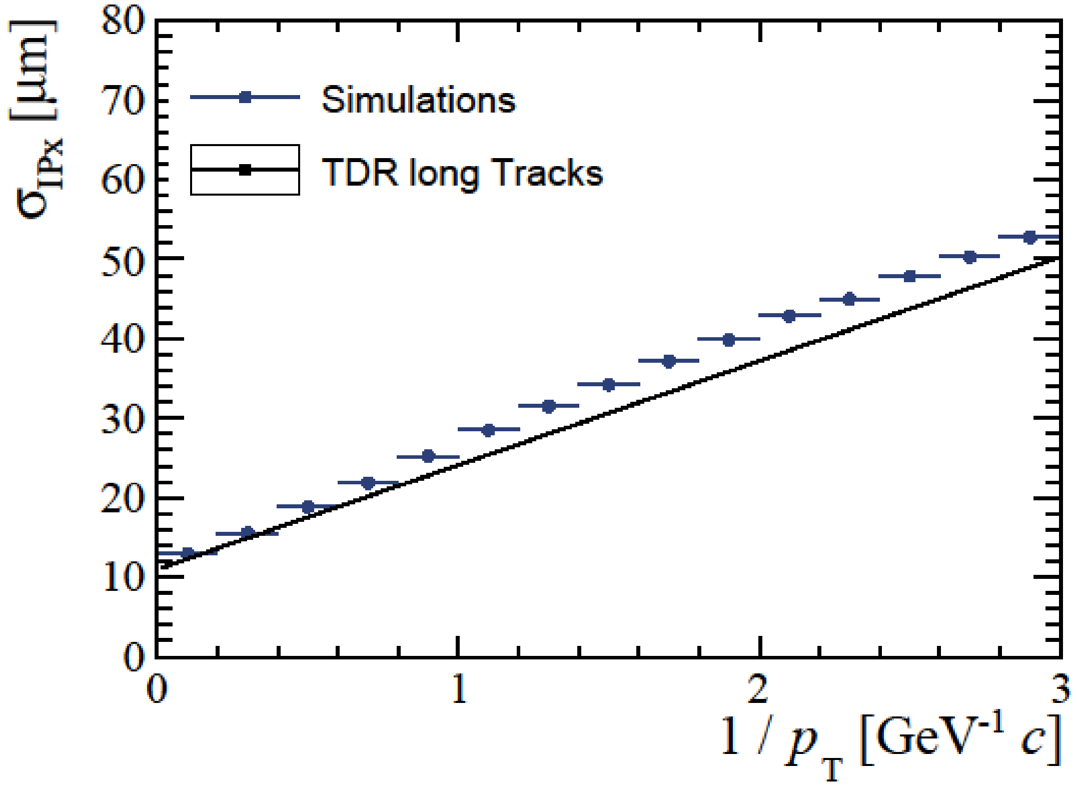


Moving to DD4Hep

CERN Fast simulation workflow

- Using DD4Hep to create the detector
- Use of Gaussino to create MC hits with Geant4 out of Pythia-generated events
- Charge sharing + clustering emulated by single hit gaussian resolution
- Python script for cheated pattern recognition and Hlt1-like Kalman filter (including DD4Hep geometry to evaluate IP)
- Validated with IP resolution of long tracks from the Upgrade 1 TDR
- Can also include digitization and pattern reconstruction, if merged with Bologna's fast simulation


Gaussino
Load DD4Hep mode
Create events with Pythia
Generate Monte Carlo hits with Geant4
Group hits to Tracks
Smear MC hits with the single-hit resolution
Fit the track
Calculate IP

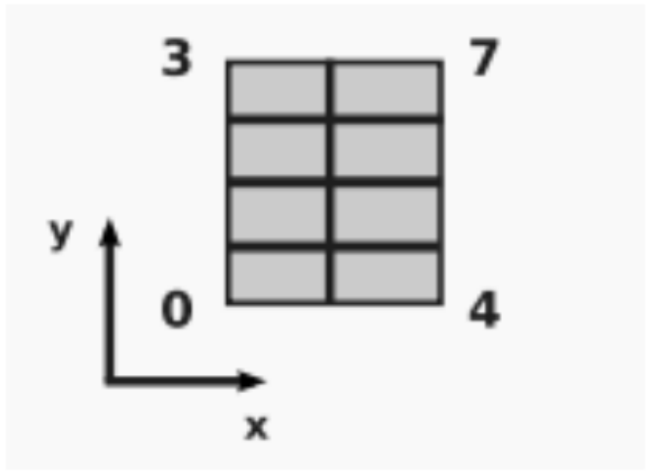


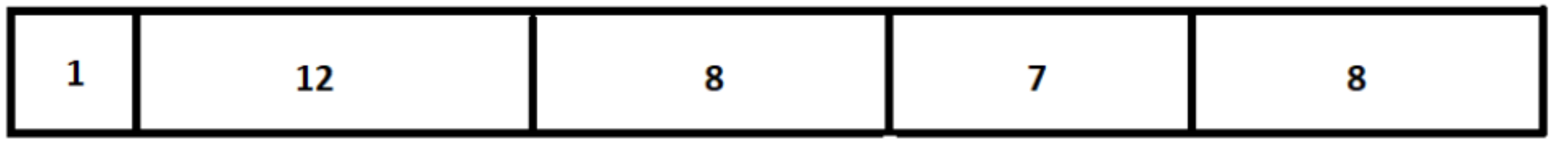
Clustering with FPGA

The Super Pixel (SP)

- In simulation pixels are organized reproducing the LHCb-VeloPixel format.

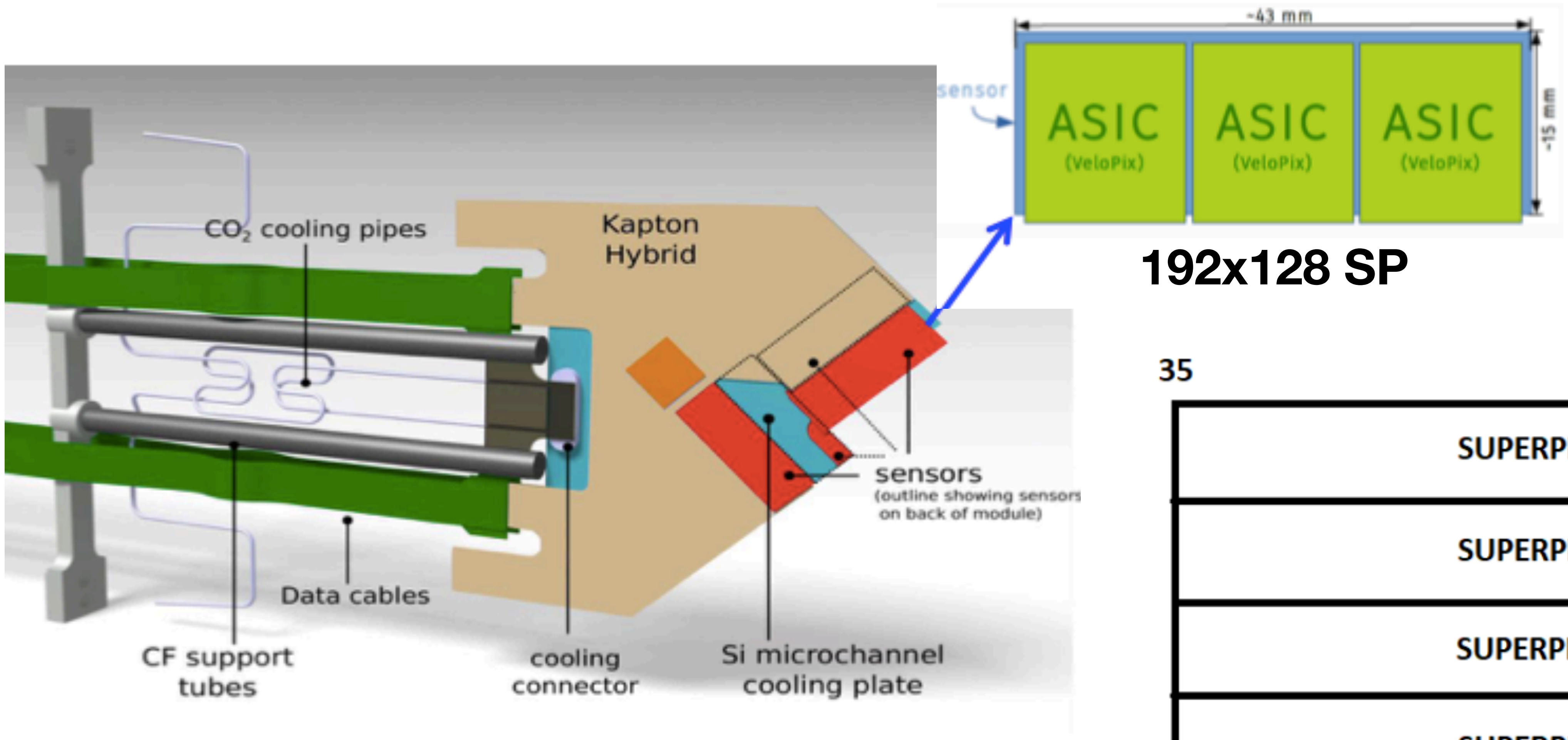
- Pixels are grouped by 8 in the so-called Super Pixel (SP) described with a word of 36 bit:
  - 1 bits for the ”hint” (isolated or not)
  - 12 bits for the SP time information (25ns sampling, 6ps time resolution)
  - 8 and 7 bits for SP spatial position
  - 8 bits to identify active pixels inside SP



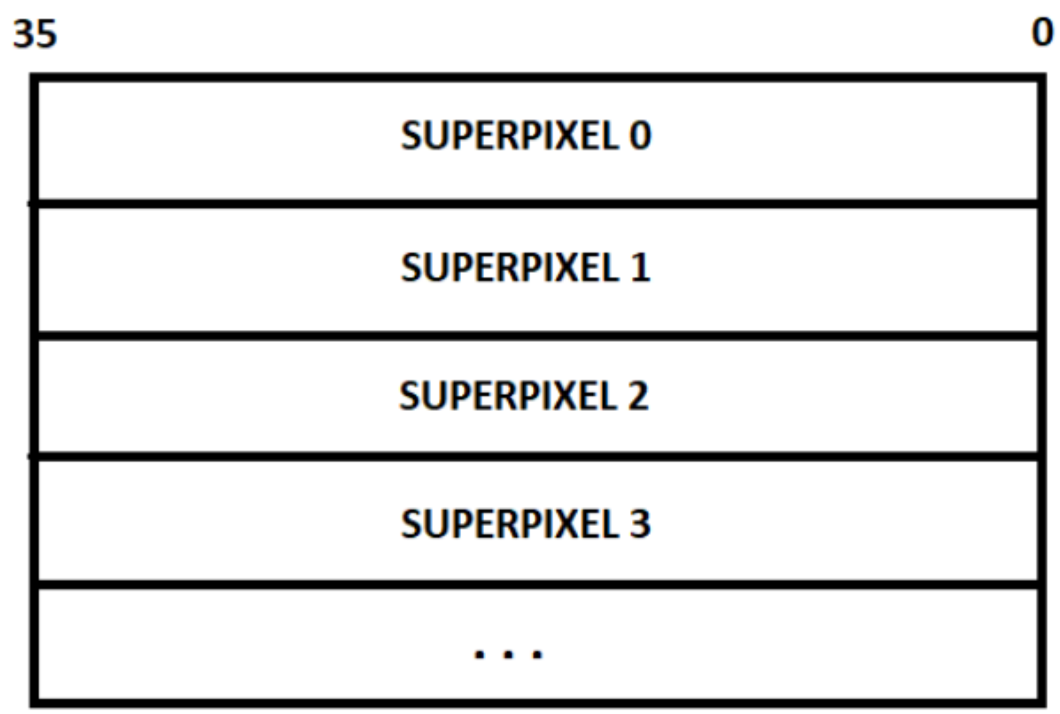


The raw bank format

- A raw bank contains information relative to each hit SuperPixel of a Sensor (zero suppression)



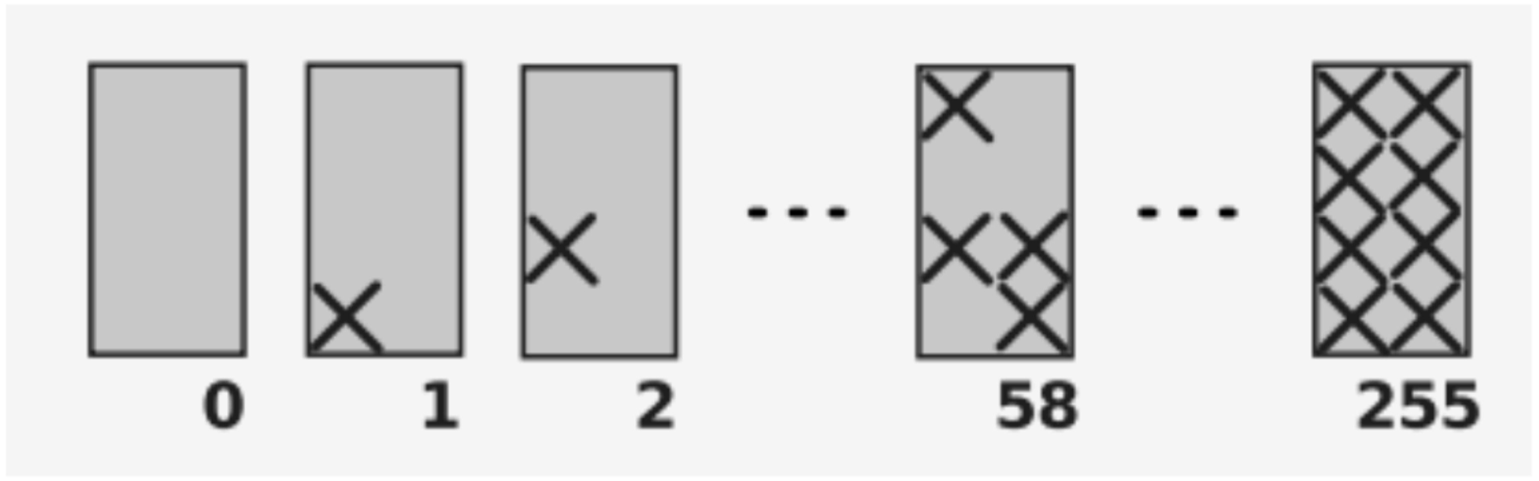
192x128 SP

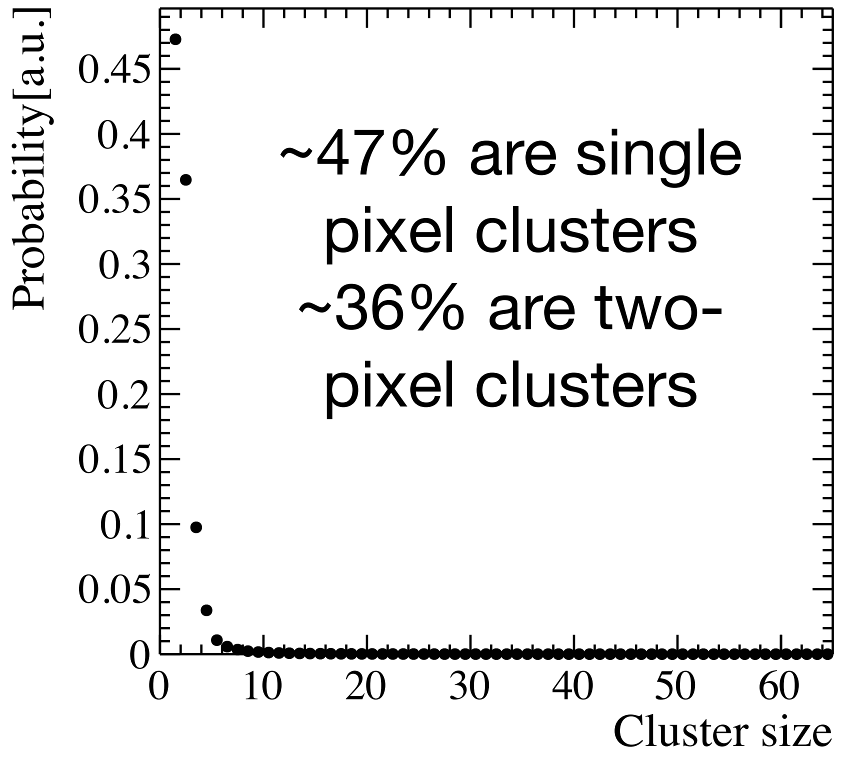


Clustering with FPGA

- **Why?**

  Large amount of clusters (2/3) are inside an isolated SP, i.e. it is possible to use a look-up table with the 256 cluster configurations (works in parallel)





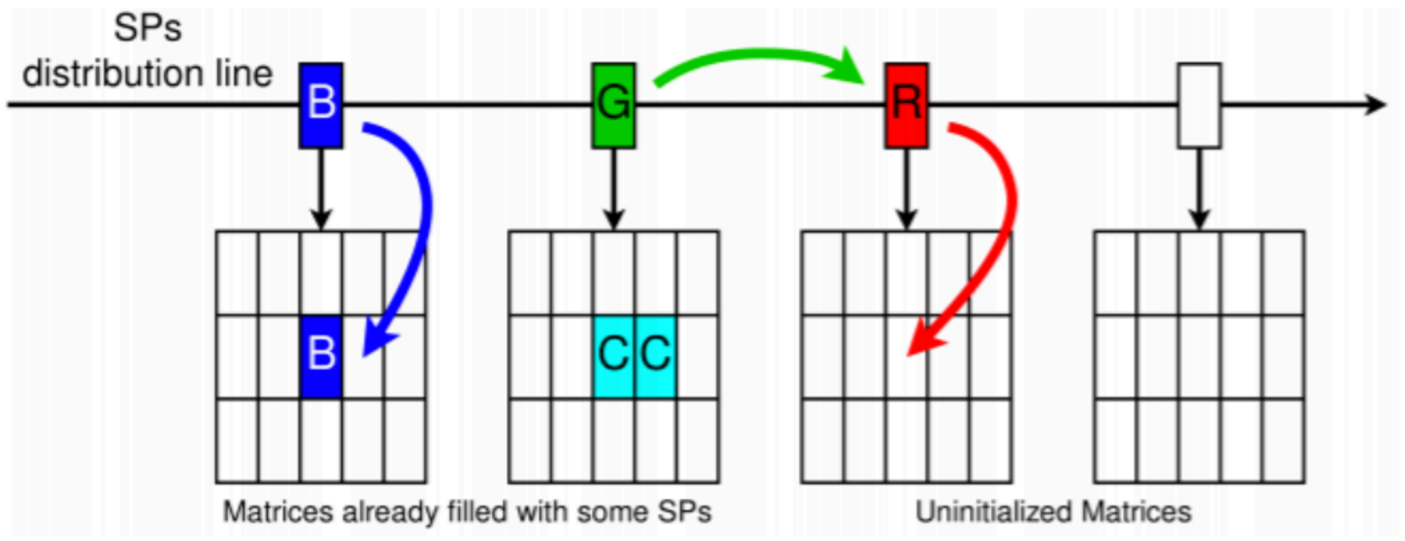
- from simulation: time info not essential for clustering

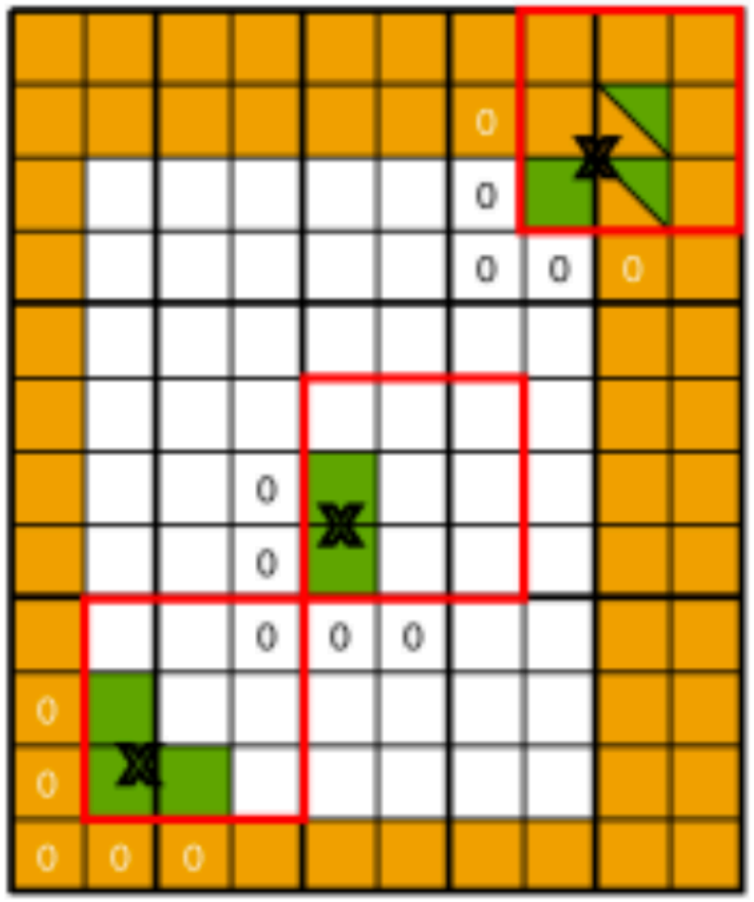
- Solution for 2D VELO clustering in FPGA has been already proposed with a parallel cluster finder for neighboring active SP

Real-time cluster finding for LHCb silicon pixel VELO detector using FPGA

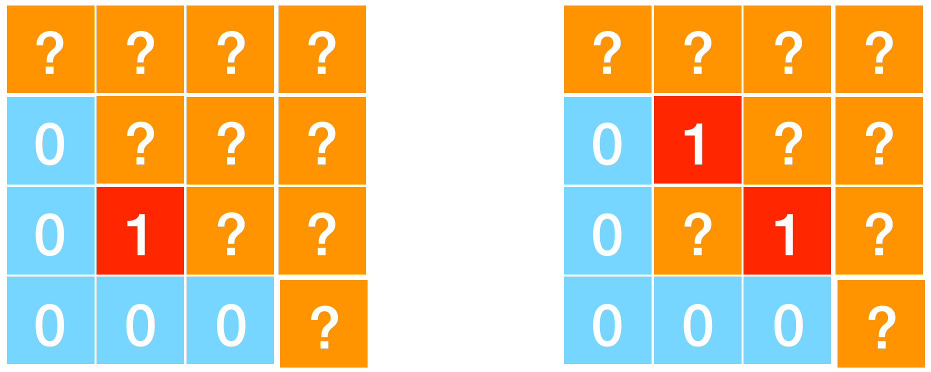
How it works?

- **Matrices** with dimension 5×3 SPs (10×12 pixels) at every clock cycle change SP input





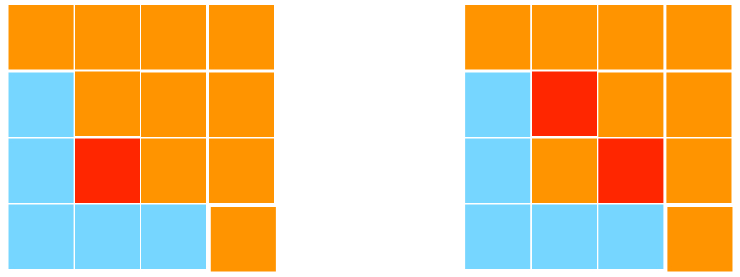
- Construction of cluster candidate from the **seed** pixel



?	?	?	?
0	?	?	?
0	1	?	?
0	0	0	?

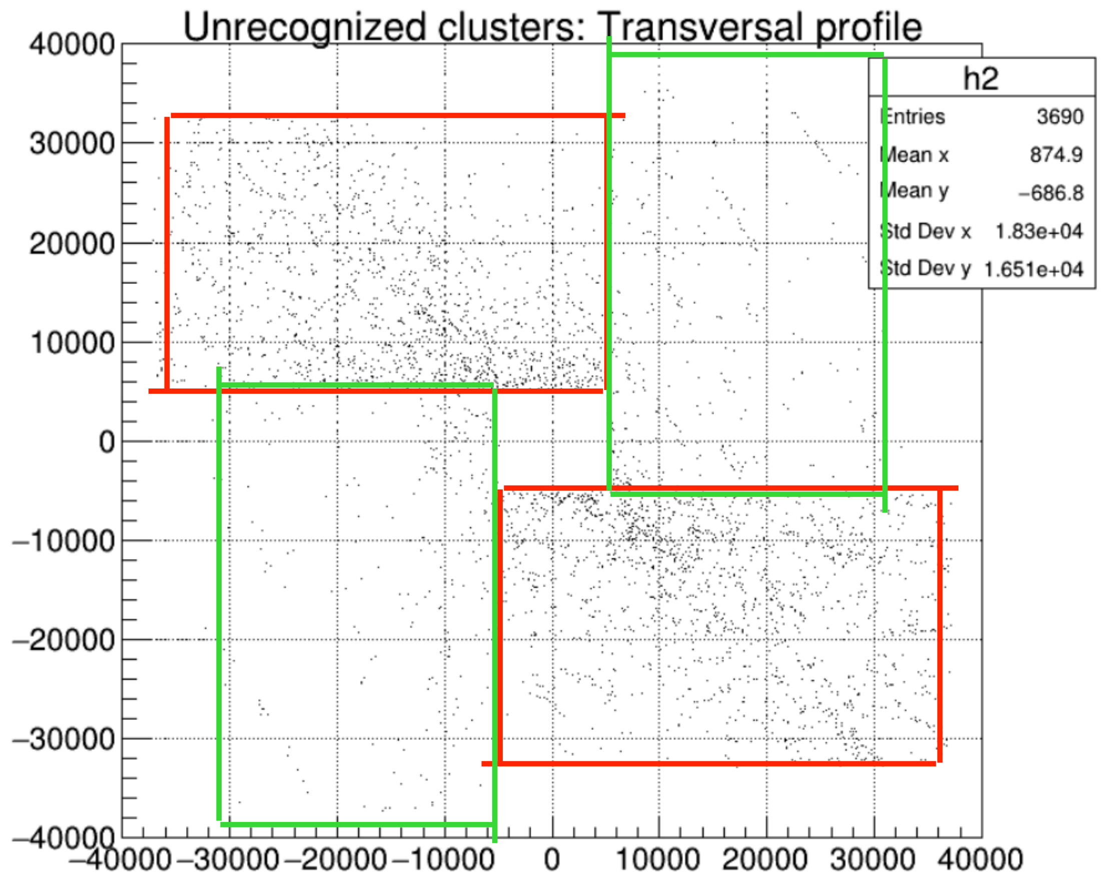
?	?	?	?
0	1	?	?
0	?	1	?
0	0	0	?

Inefficiencies



- Studies with a C++ code that emulates the behavior of a FPGA are in line with what expected when comparing to previous results

	Inefficiency (%)
CPU	0 by definition
FPGA-like	0.5
Tab.1 of LHCb-INT-2020-010	0.87



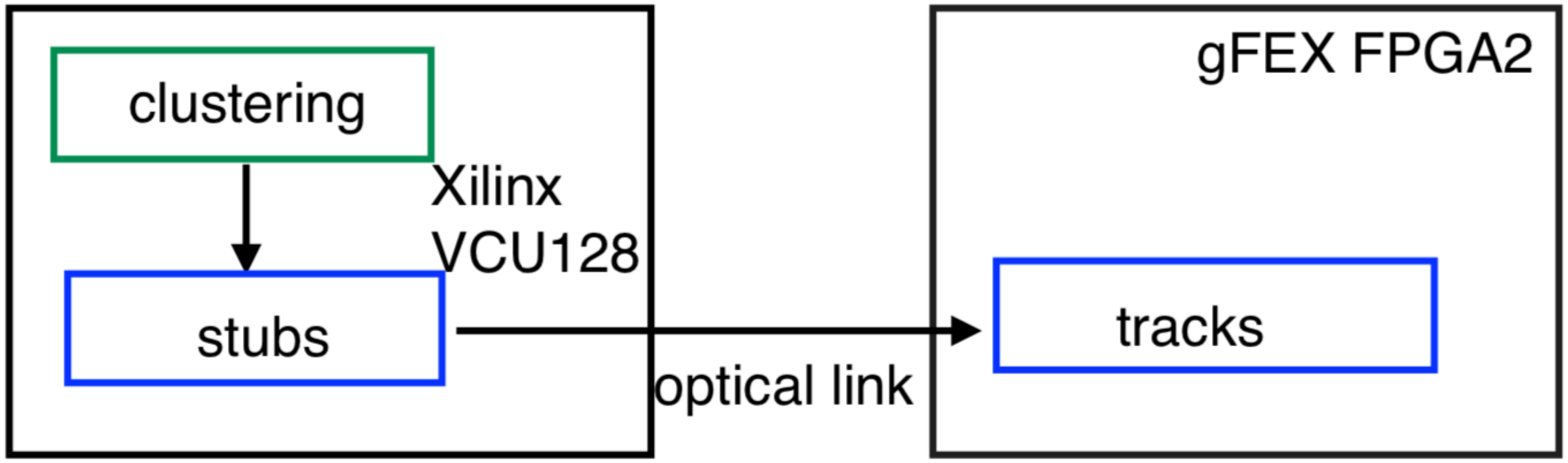
- Clustering efficiency is not symmetric:
  - "wrong" orientation of the local coordinates in the sensor -> to be fixed
  - by definition (see seed patterns) -> should be negligible but can eventually be fixed with other patterns to account

Next steps

- Apply tracking and realize performance studies for a FPGA-based clustering
- Improve the algorithm and study its critical issues
- Start the implementation on FPGA

Tracking with FPGA

Tracking device based on FPGA



- VCU128 not yet available, working with VC709 for the construction of stubs

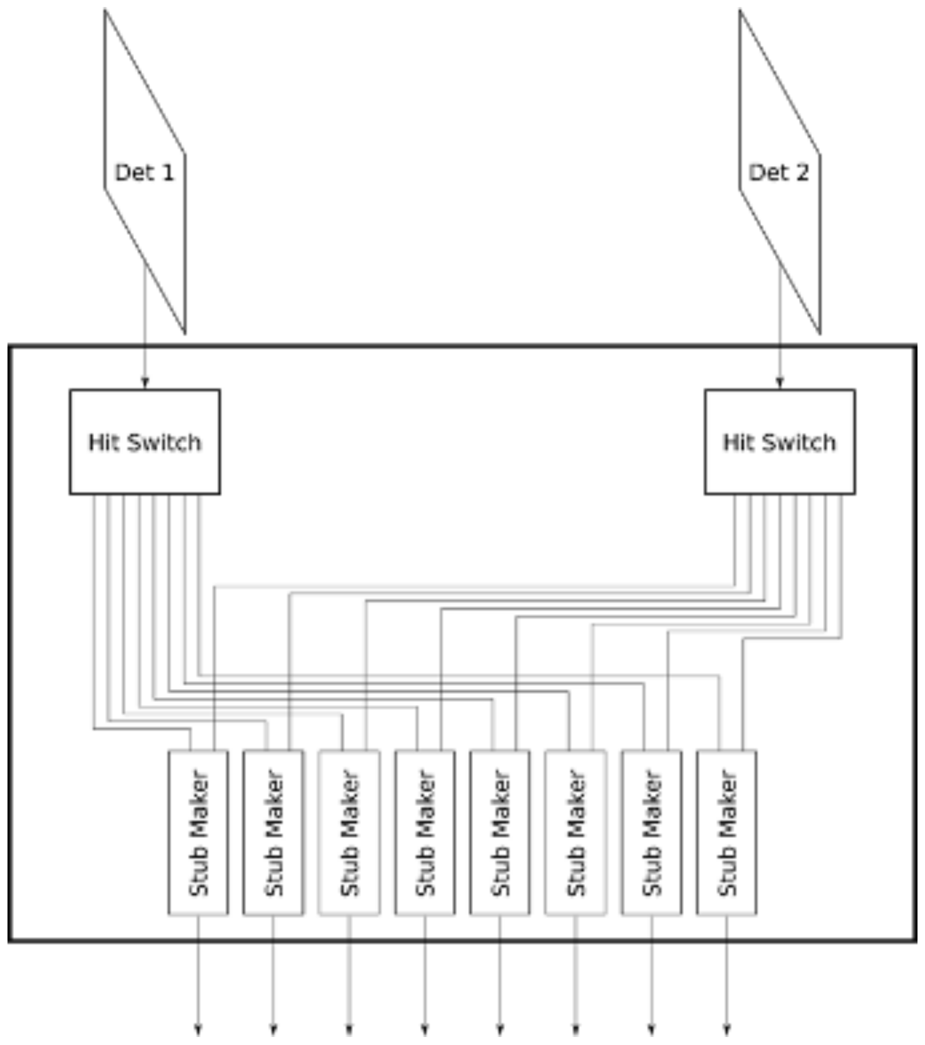
Stub constructor

- The *Stub constructor* receives the detector hits from a **couple** of sensor planes: two switches deliver the hits to a pool of sub makers

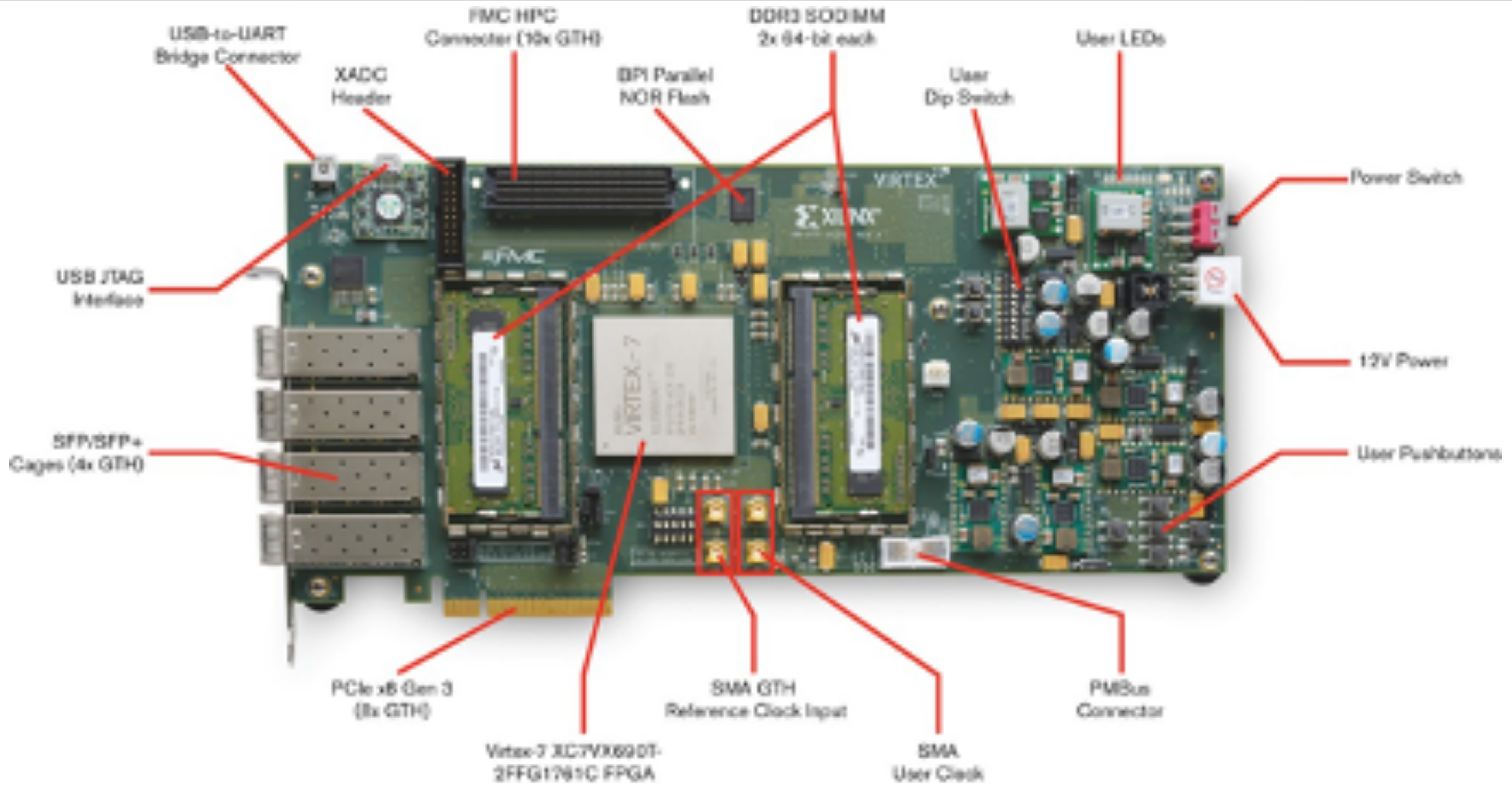
- Each stub maker receives data from exclusive sensor regions, uniformly distributed w.r.t. phi and not uniformly distributed w.r.t. the radial coordinate

- The (r, phi) coordinates are evaluated before the hit switches

- The **Stub maker** processes every combination of hits that are received from the regions it is associated with



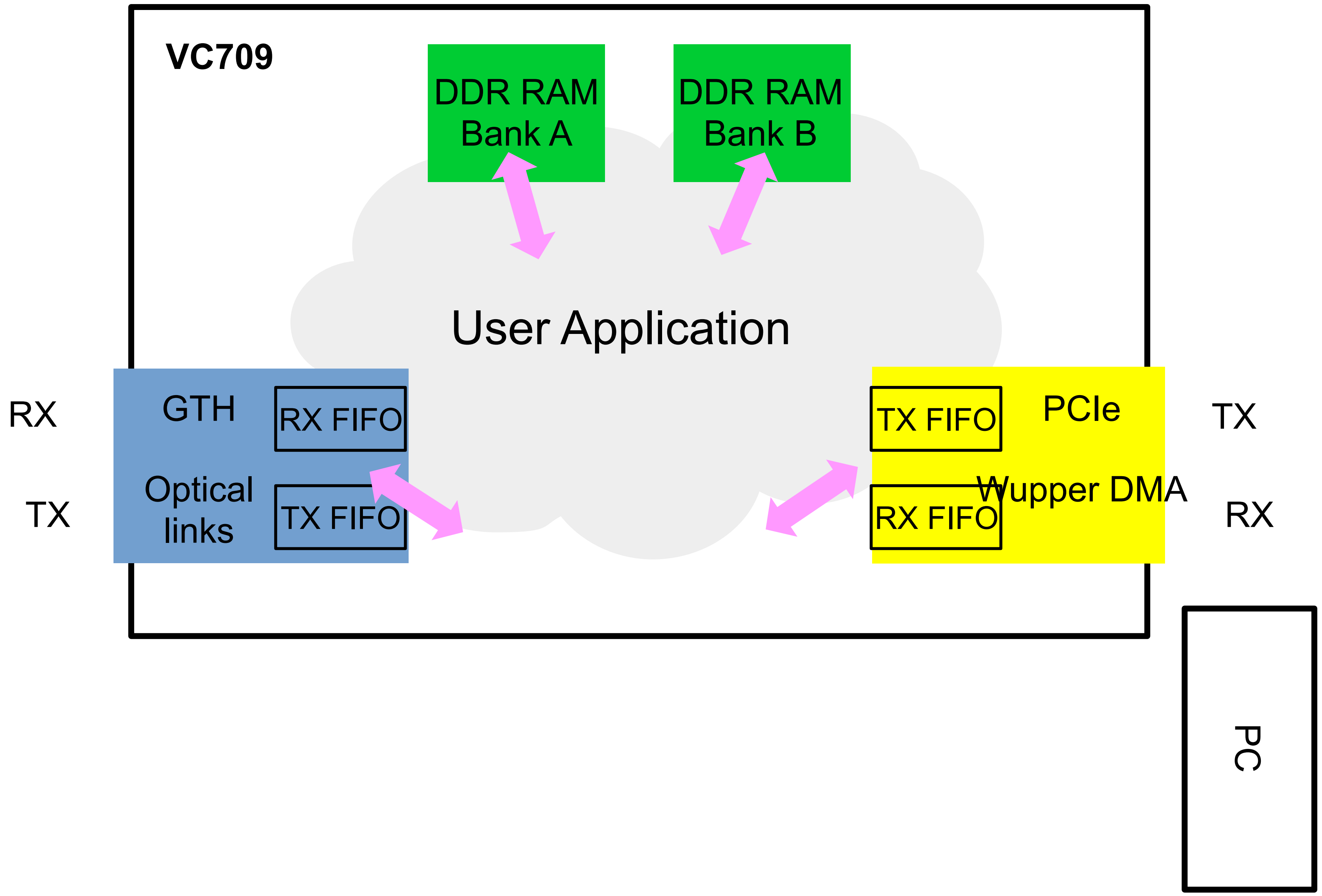
VC709 implementation



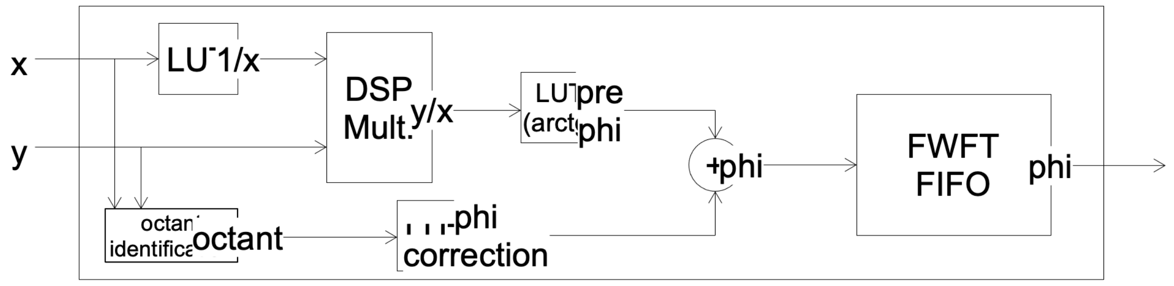
- The main *blocks* implemented:
  - **DDR** banks (2x4GB, Xilinx MIG)
  - **PCIe** communication (using Wupper)
  - **GTH** (over optic fibers)

- An *application* connects the thee blocks and manages the data flow

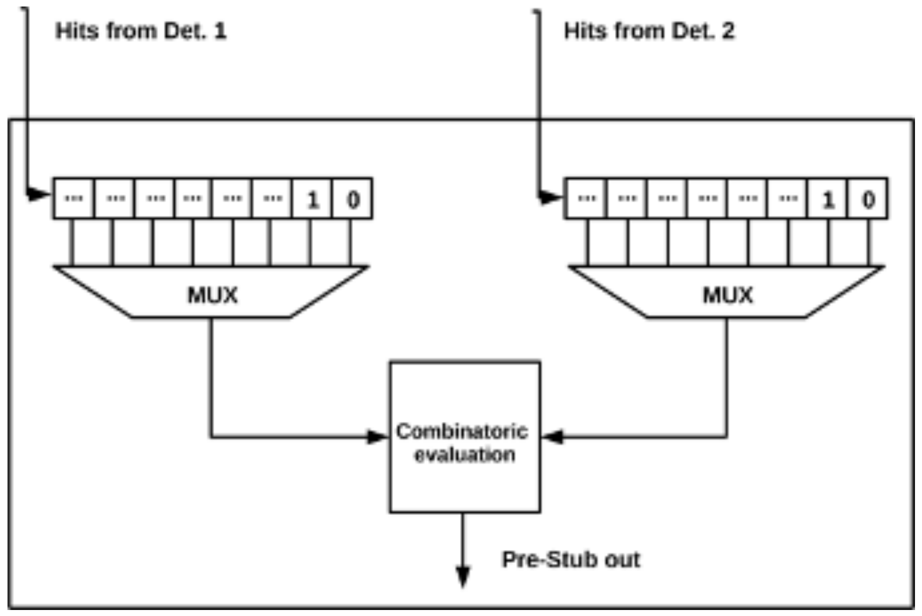
VC709 block scheme



Single module tests

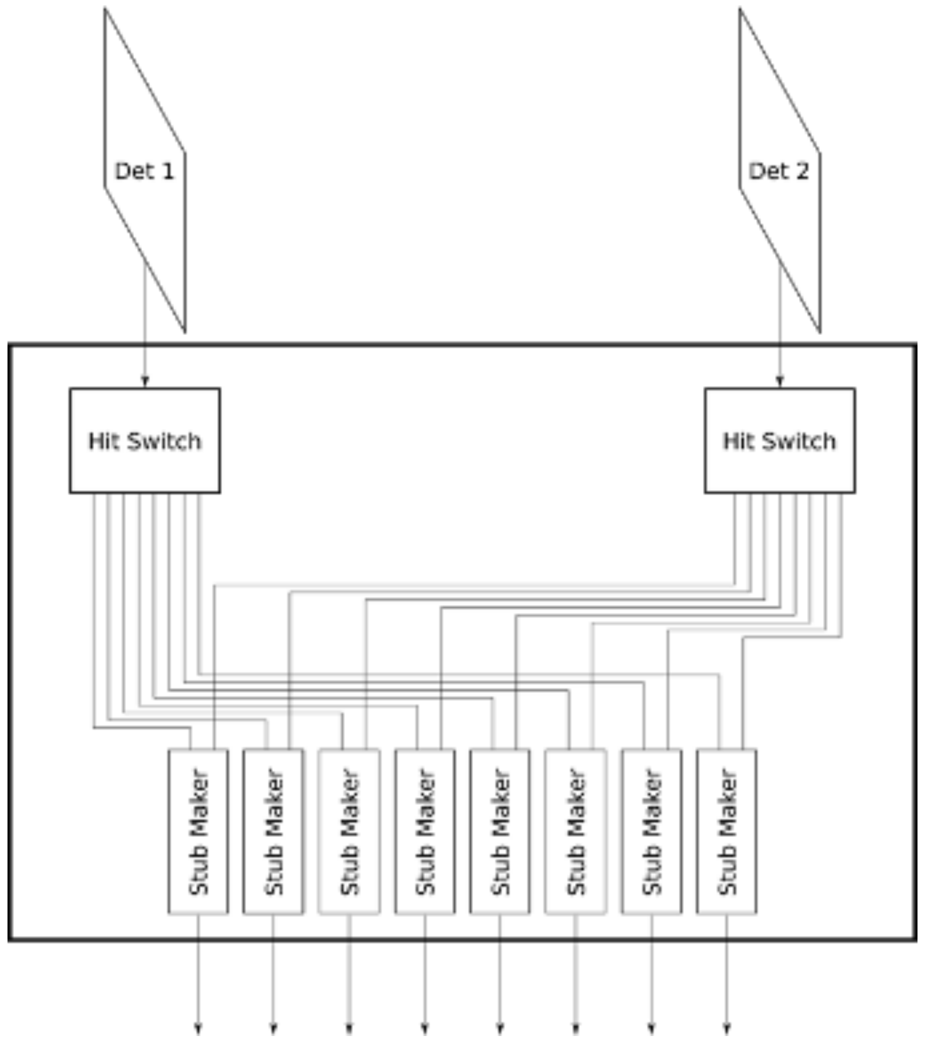


- **xy-to-rphi converter** ✔
  - provides the conversions from (x,y) coordinates to radial coordinate
  - tested with ref. clock = 200 MHz
- **(Pre-)Stub maker** ✔
  - provides the combinations of couple of hits without applying geometrical and timing filtering to the candidate stub
  - tested with ref. clock = 200 MHz



Stub constructor (SC) test

- *Hit data* are provided from the PC via the PCIe interface
- The *hit switches* deliver the hit data to a subset of *Stub Makers* according to the "address" value of the hit data
- The Stub maker that receives one or more data from both the inputs (det1,det2) processes the **N1*N2** combinations and provides the couple of hits (pre-stubs) in output
- a fan-in in instantiated to collect the data from the Stub Makers and deliver to the PC via PCIe interface
- A grid of Stub Makers has been instantiated in the VC709 firmware. The expected combinations (evaluated via software from the input data) are compared to the results from FPGA with success using a ref. clock of 240 MHz. ✔



Implementation of the SC grid

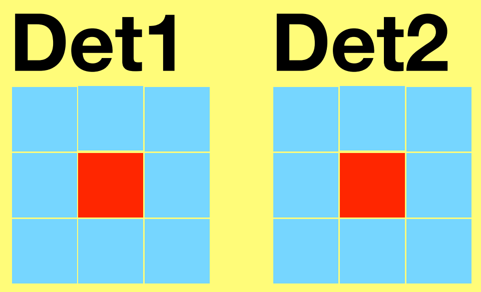
- Naive (old) implementation for the SC *grid* (per couple of sensors):
  - each sensor is divided into **Nr*Nphi** exclusive areas
  - the total number of combinations is (virtually) **$(Nr*Nphi)^2$**, i.e. Nr=8, Nphi=32 -> total 65 536
  - a 4-dimensional grid is declared in the VHDL through "*for … generate*" statements

- Actually, only **(Nr*Nphi)*9** Stub makers should be instantiated (i.e. total active = 2 304)

- **New** VHDL code for the **stub constructor**:
  - using the "*if … generate*" statements
  - VHDL simulation does not give any problem ✔

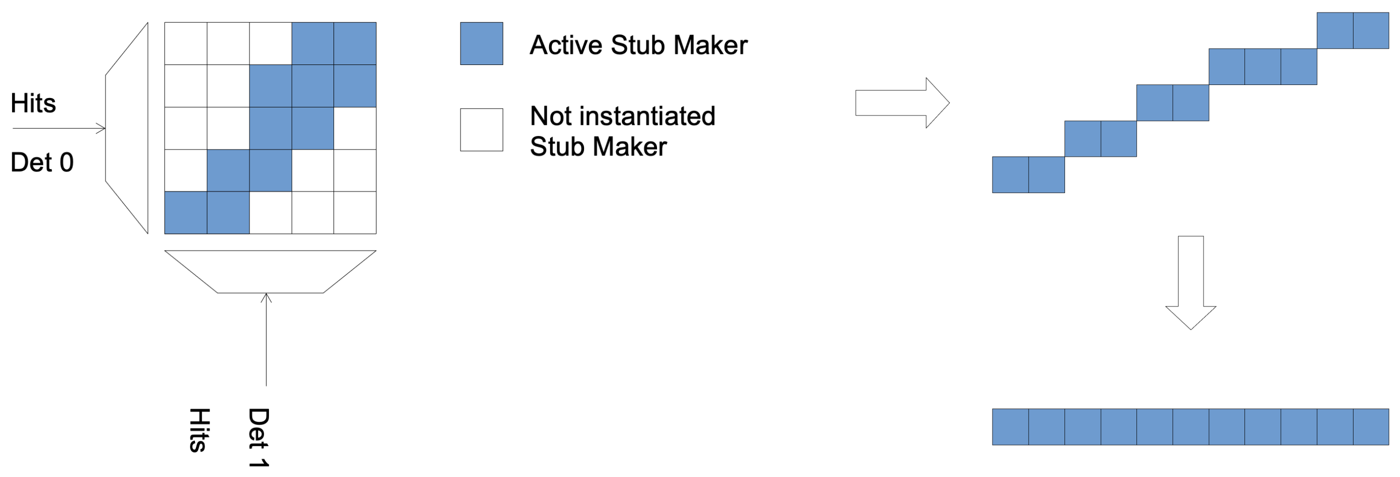
- **Implementation** on FPGA:✘
  - it works only for small values of Nr, Nphi (Nr=4, Nphi=4), i.e. total = 256, with total active = 16 (first neighbor pixels on second sensor not even instantiated
  - proceed with a revision of the implementation

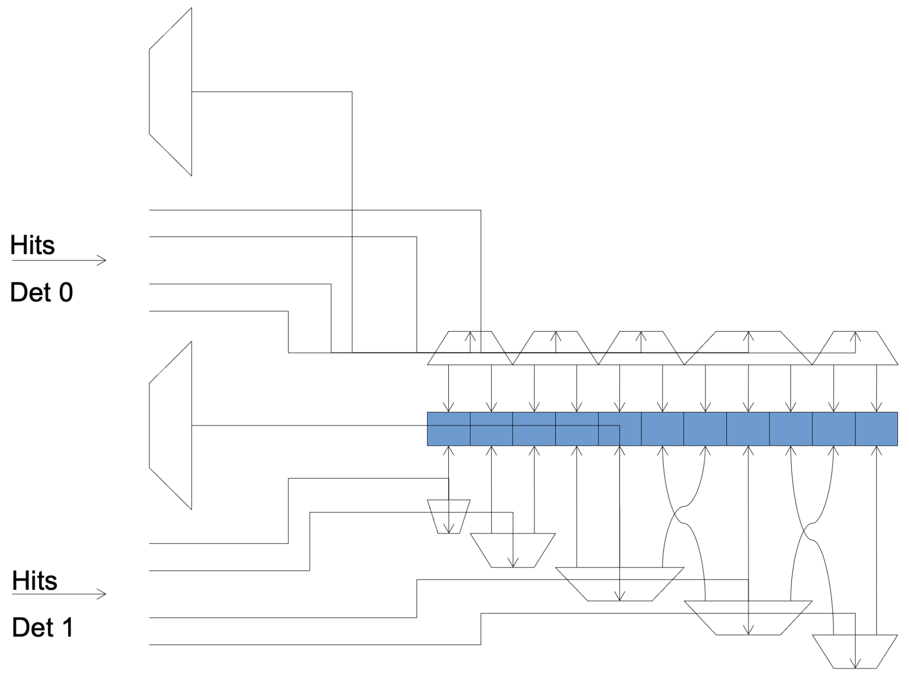
The number "9" is given by the fact that for one hit in a bin of the first sensor, the second hit is typically in the same bin or one of the first neighboring of the second sensor



A possible solution for the SC grid

- Keep the naive implementation
  - the work of "cutting" the Stub Maker instances is demanded to the VHDL compiler and there is a big difference between the dimensions of the 4-dimensional grid and the actual number of active (and instantiated) Stub Makers
  - the 4-dimensional grid has been "**flattened**"
  - the hit switch connections have been re-routed in a smart way
  - also the recollection of the identified stubs has been changed



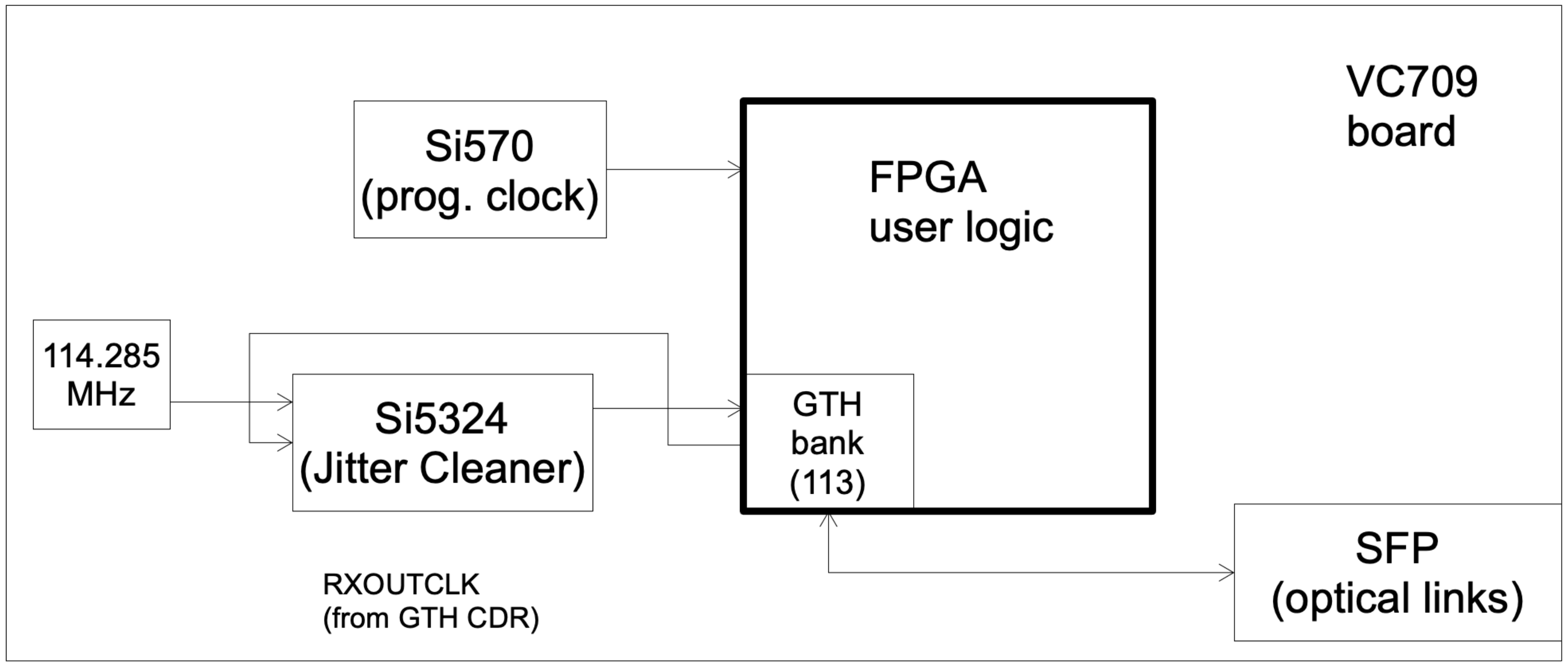


SC grid - status

- System re-tested and working properly in **simulation**, to check the goodness of the modification ✔

- We still have some problems in the **implementation**:✘
  - the compilation *time* increases non-linearly (20 mins for 16 StubMakers vs 5 hours for 64 Stub Makers)
  - S. Riboldi is now having a look to identify critical issues in the implementation
  - in particular issues should be addressed to non-completely-scalable parts of the implementation (i.e. the fan-outs and the fan-ins to deliver and recollect the data)

Communication over GTH



- GTH transceivers used for communication between the **VC709** and the **gFEX** board
  - VC709 hosts the Stub Constructor, gFEX hosts the rest of the firmware (switch + engines)

- **Two** on-board components for *clocking*:
  - Si570 programmable clock used as a reference for the application on the VC709 (i.e. the stub constructor)
  - Si5324: Jitter cleaner, used to provide the reference clock to the GTH block

VC709 - gFex communication

- **Setup of the GTH lane (on the two boards)**
  - 11.2 Gbps, with 140 MHz ref. clock
  - one fiber connected from the VC709 to the gFex and viceversa
  - Note: GTH are organized into “quads” and only one lane is used to perform the clock data recovery (CDR) and data synchronization

- **Test performed with PRBS data:**
  - the gFEX receives the data, and the PRBS test passes (received data are within the expected pattern) ✔
  - the VC709 can not “align” to the incoming data, and the PRBS test fails✘