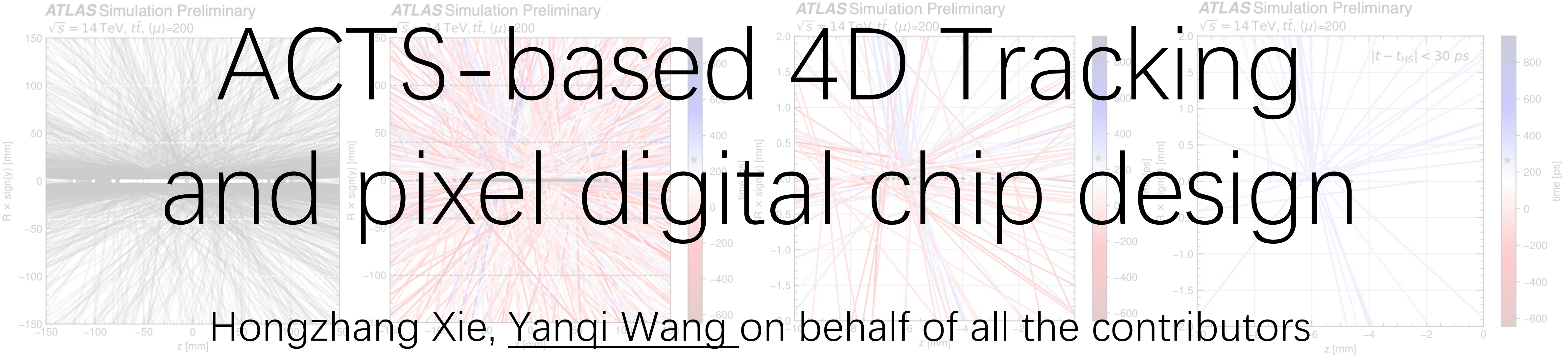
ACTS-based 4D Tracking and pixel digital chip design



Hongzhang Xie, Yanqi Wang on behalf of all the contributors

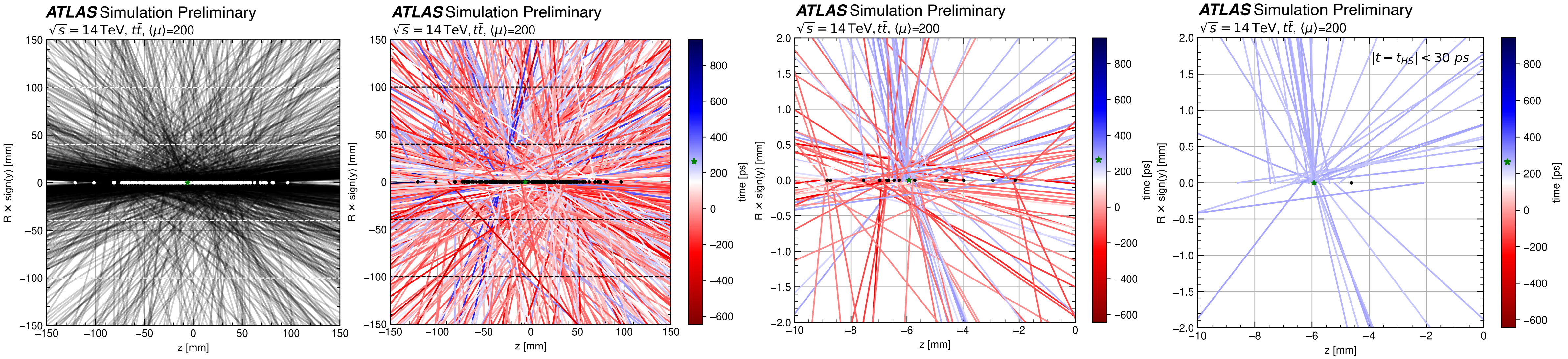
University of Science and Technology of China

CLHCP 2025 @ 河南新乡 on Oct. 31st , 2025

Introduction
Chance of future 4D tracking for ATLAS ITk

- Motivation – LHC upgraded to HL-LHC

  Luminosity up to $7.5\times10^{34}\ \mathrm{cm}^{-2}s^{-1}$, pile-ups up to **200**

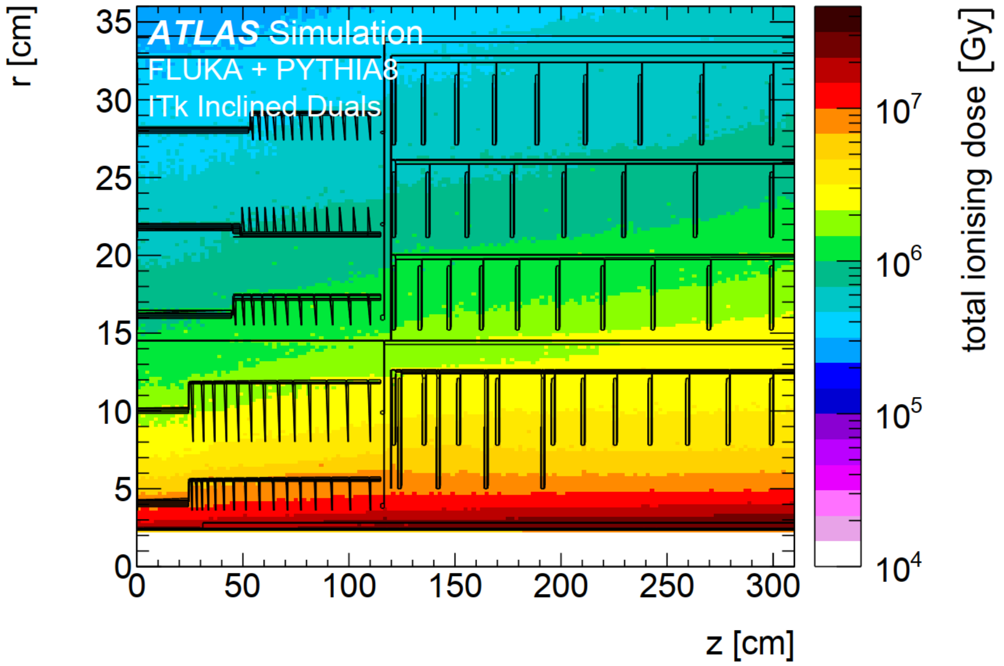


Longitudinal view of a simulated $t\bar{t}$ event with <μ>= 200 in ATLAS ITk

No time information | Add time information | Zoom in space | Separation by time

Introduction
Chance of future 4D tracking for ATLAS ITk

- The two innermost layers of the ATLAS ITk Pixel is designed for a maximum luminosity of **2000 fb^{-1}**.
- Chance to implement 4D tracking for ATLAS in the middle of HL-LHC operation:
  - Efforts in hardware and software
  - Need to understand the physics potential
  - This talk reports the progress of low-level 4D **tracking** and a design of **pixel digital chip**



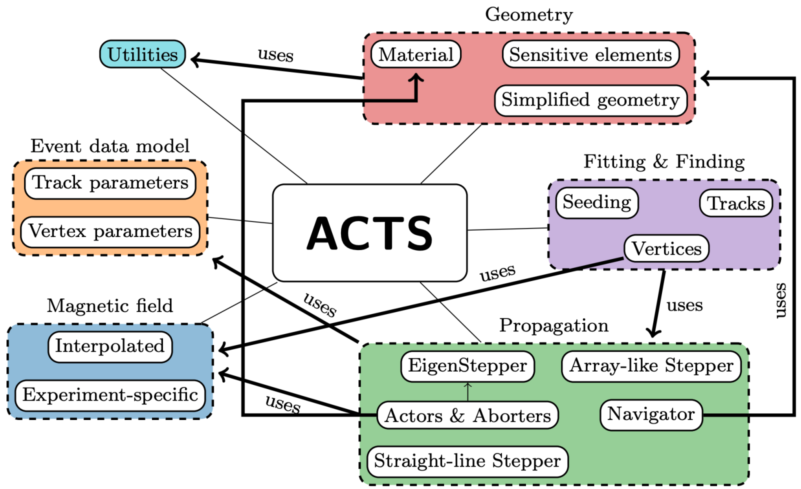
The total ionizing dose distributions for the ITk Pixel Detector.
<5 MGy in 4000 fb^{-1} for unreplaceable part
up to 10 MGy for 2000 fb^{-1} for Layer 0

ATLAS-TDR-030 Inner Tracker Pixel Detector

4D tracking with ACTS

ACTS GitHub: https://github.com/acts-project/acts
ACTS docs : https://acts.readthedocs.io/en/latest/index.html

Introduction to Software



- **ACTS** (A Common Tracking Software) is an experiment-independent toolkit for (charged) particle track reconstruction in (high energy) physics experiments

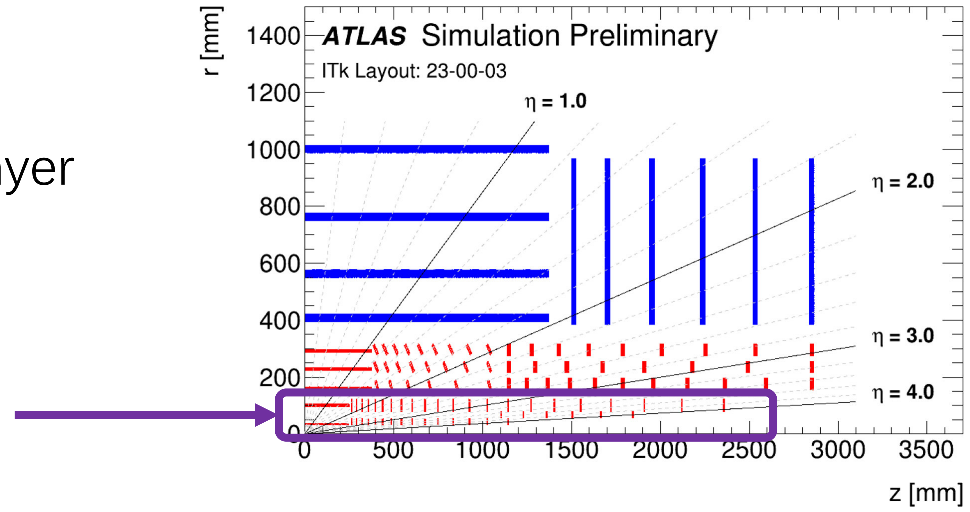
➢ track parameters in ACTS

$$\vec{x} = \left(l_0, l_1, \phi, \theta, \frac{q}{p}, t\right)$$

- Smear truth hit information in the ITk innermost 2-layer pixel region with gauss distribution

$$\sigma(t) = 50\,ps$$

a rough estimation of hardware



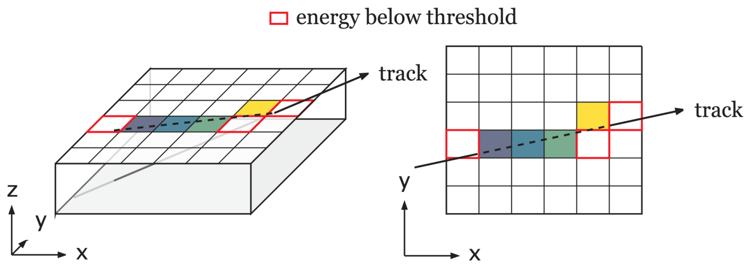
Full chain workflow in ACTS

with time algorithm

propagate time information

Fatras / Geant4

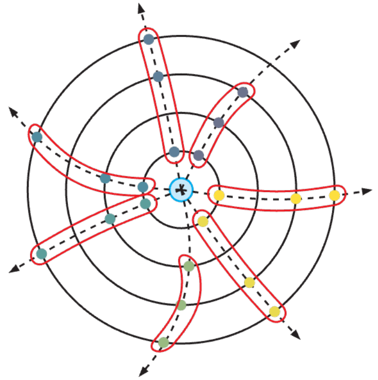
simulation

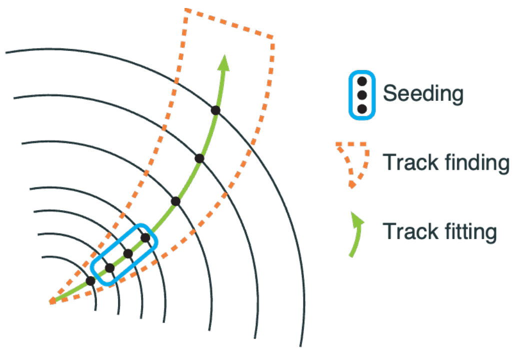


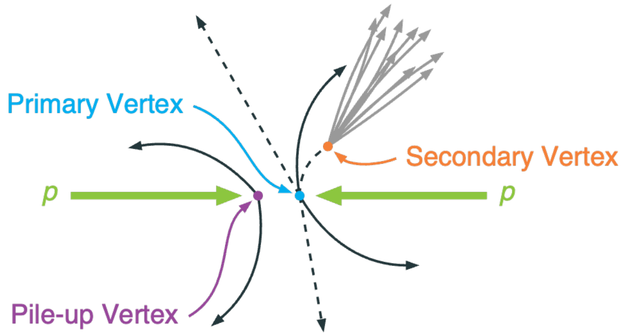
digitization & clustering

time information

use smeared hit information







seeding

track finding& fitting (CKF)

vertexing

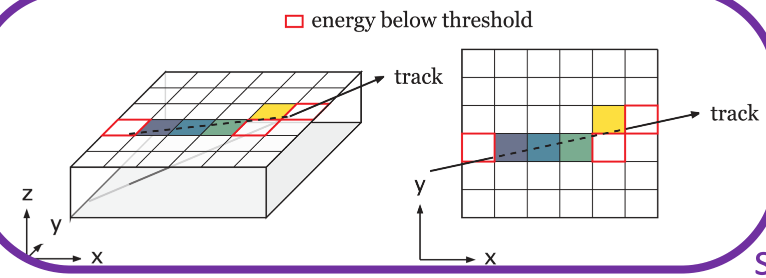
Full chain workflow in ACTS

with time algorithm

propagate time information

Fatras / Geant4

simulation

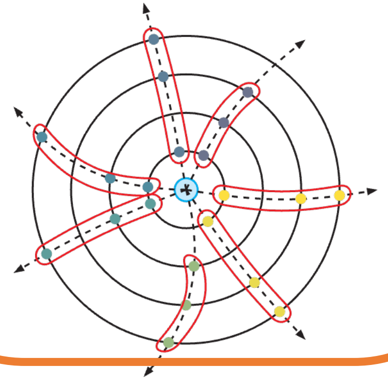


digitization & clustering

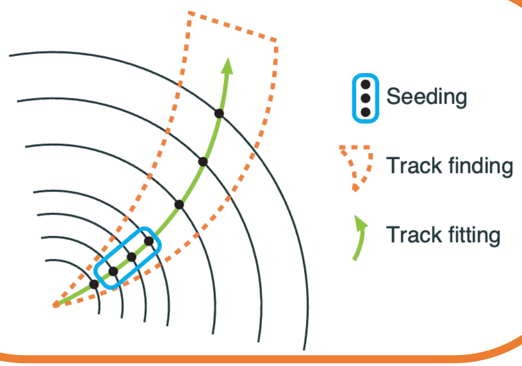
studying in Athena

time information

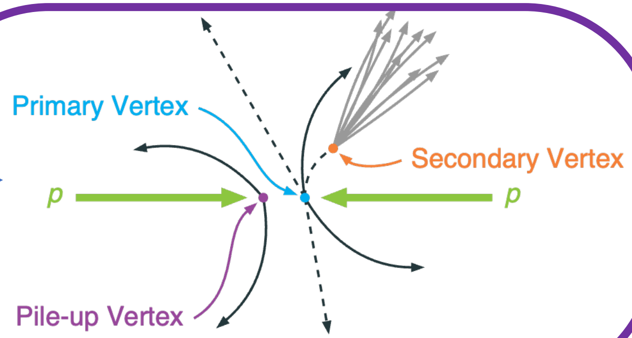
use smeared hit information



seeding



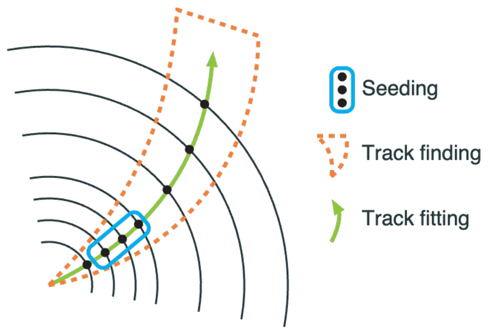
track finding& fitting (CKF)



vertexing

studied in ACTS, not integrated yet

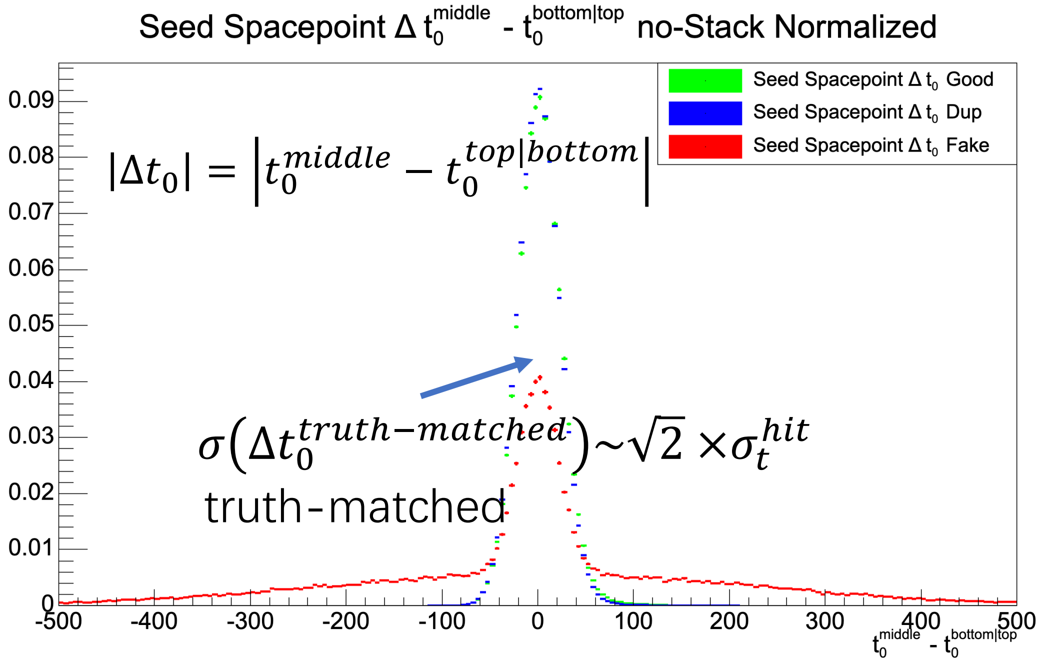
4D seeding algorithm



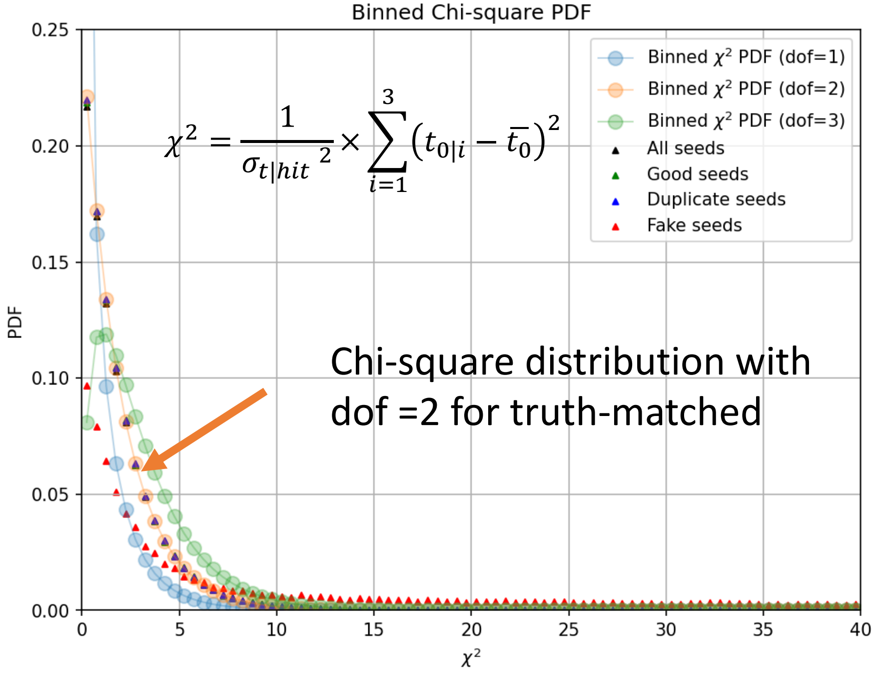
- Variable t_0 for spacepoint

$$t_0 = t_{arrival} - \frac{1}{c} \cdot \sqrt{x^2 + y^2 + z^2} = t_{arrival} - t_{fly}$$

➢ A gauss filter for **doublets** & a chi-square filter for **triplets**

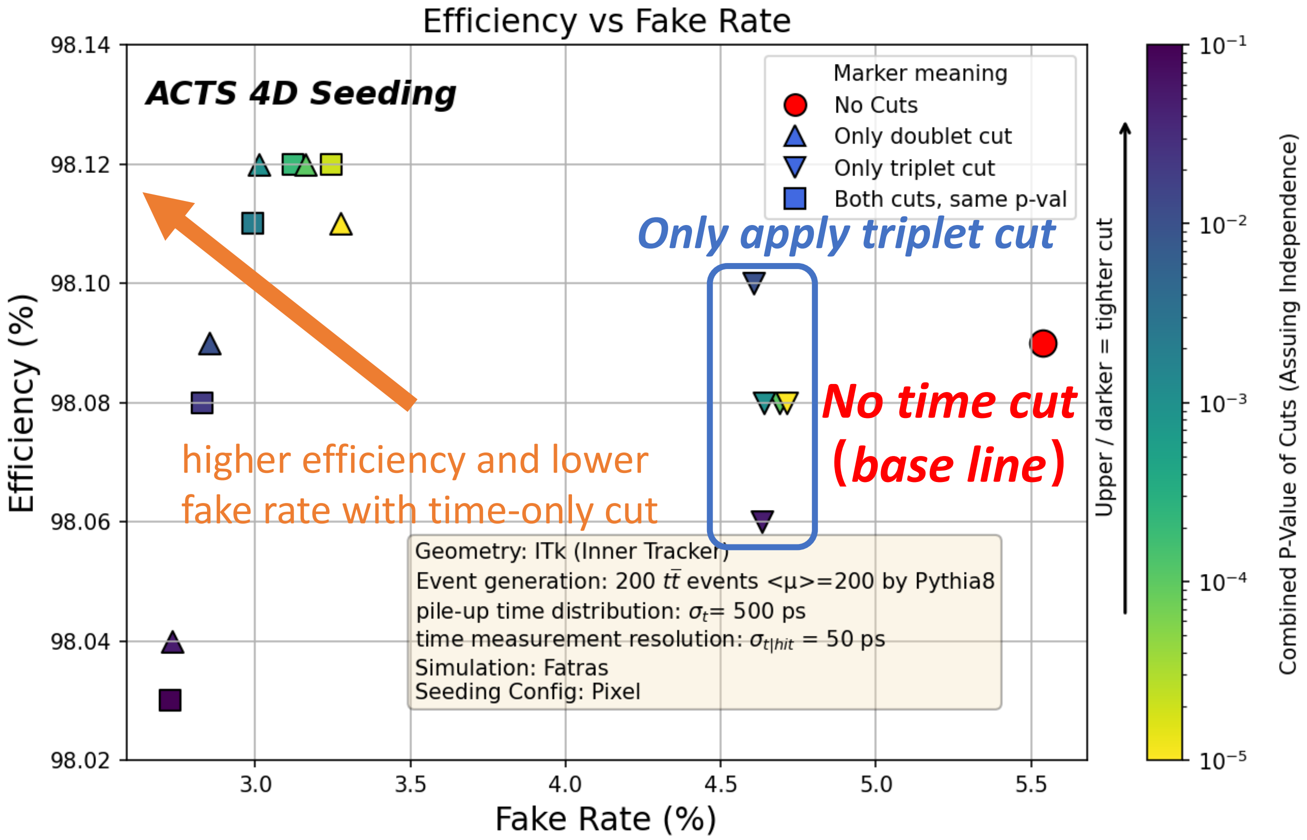


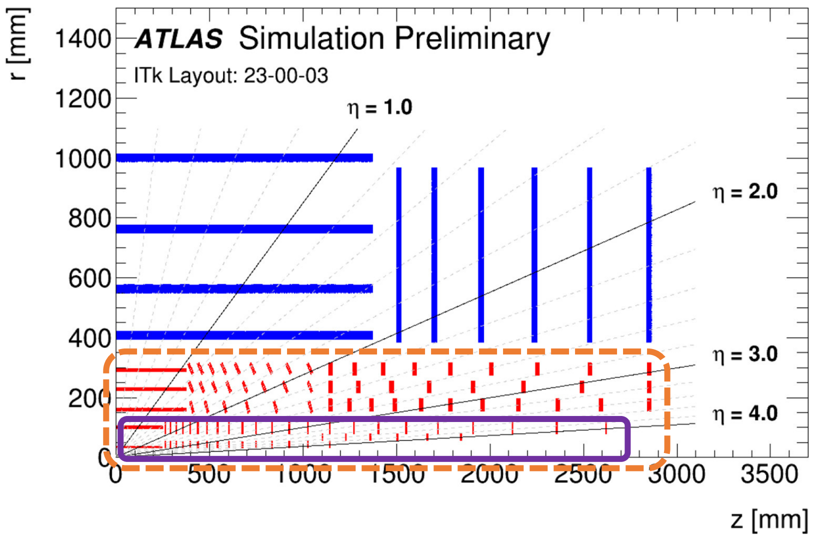
Doublet - Normalized distribution of Δt_0



Triplet - Normalized distribution of χ^2

4D seeding performance



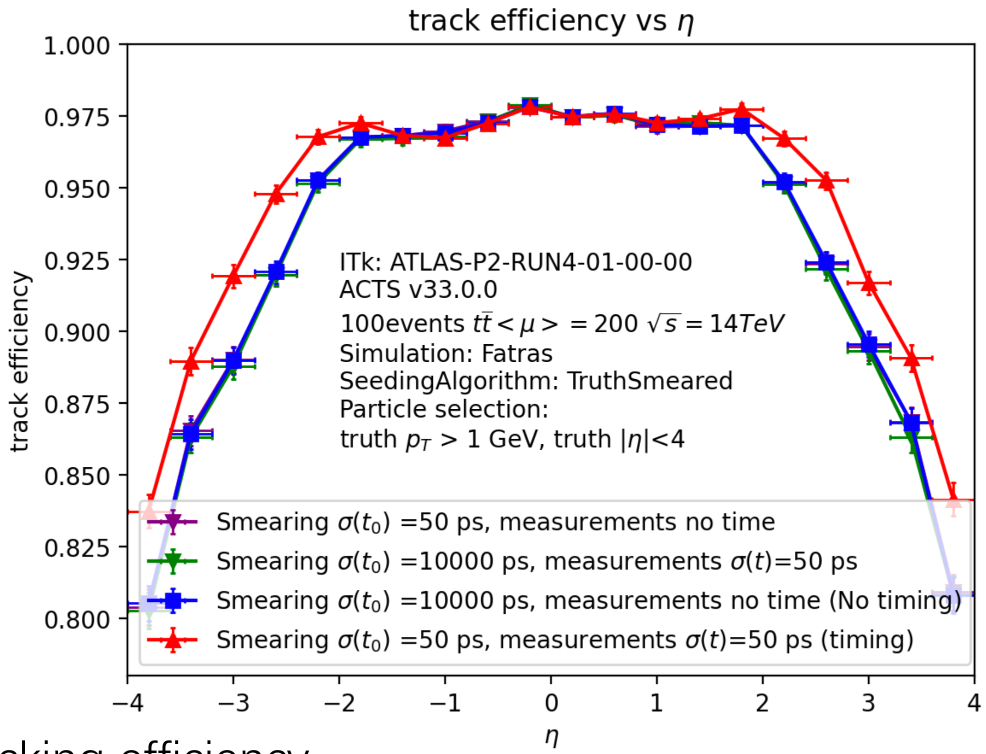


- Using **pixel (5 layers)** spacepoint seeding
- Adding time for innermost **2** layers

p-value: corresponding to the probability of excluding a truth-matched seed by cut

Doublet-cut seems more effective on this condition

4D tracking performance



Tracking efficiency

93.72% (no timing) to 94.87%↑ (50 ps timing)

93.71% (only "measured" time for initial track parameters)

93.63%↓ (only time for measurements)

Combinatorial Kalman Filter $\vec{x} = \left(l_0, l_1, \phi, \theta, \frac{q}{p}, t\right)$

$$\vec{x}_k = \vec{x}_k^{k-1} + \mathrm{K}_k(\vec{m}_k - \mathrm{H}_k \vec{x}_k^{k-1})$$

- Add time for both **initial track parameters** and **measurements** of innermost 2-layer pixel
- Event generation: ***$t\bar{t}$* events** $\langle\mu\rangle$=200 with Pythia 8
- **Fatras** simulation & **Truth-Smeared** seeding

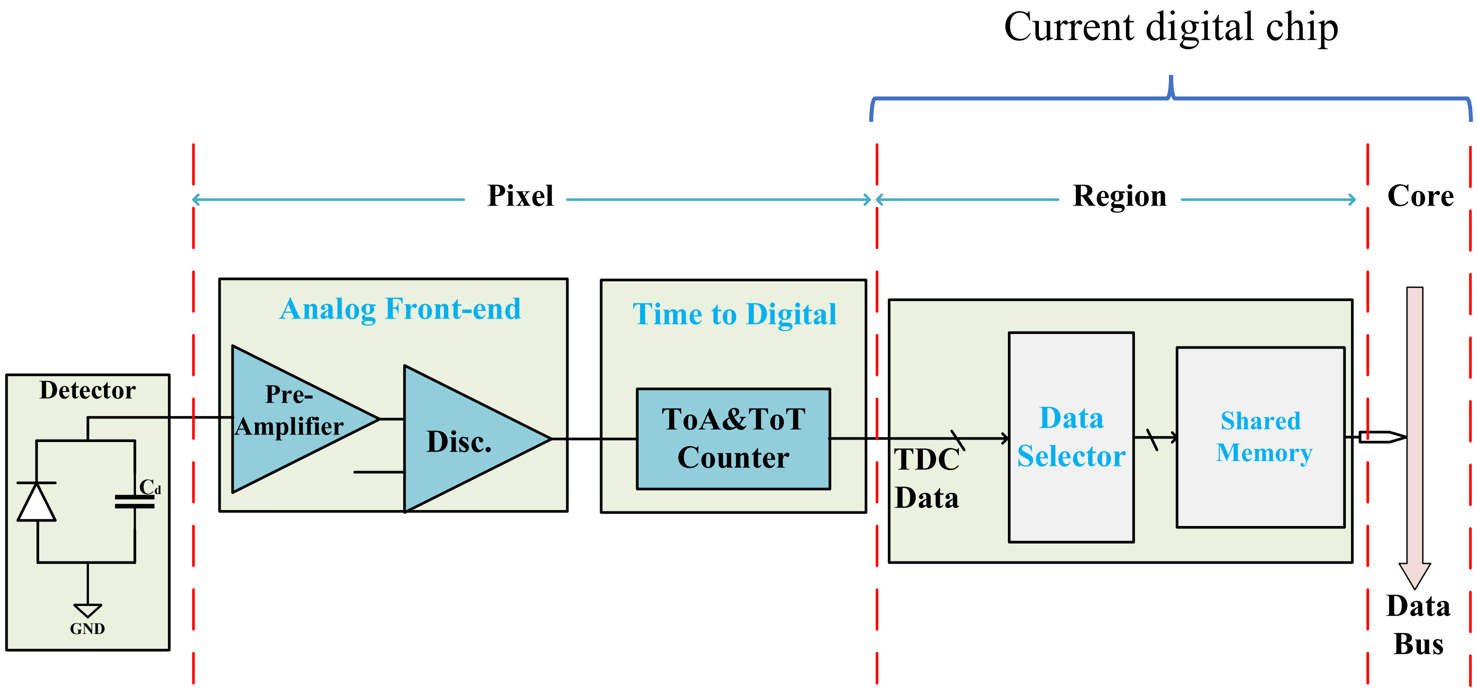
RED (4D tracking): CKF initial track parameters and measurements have time information

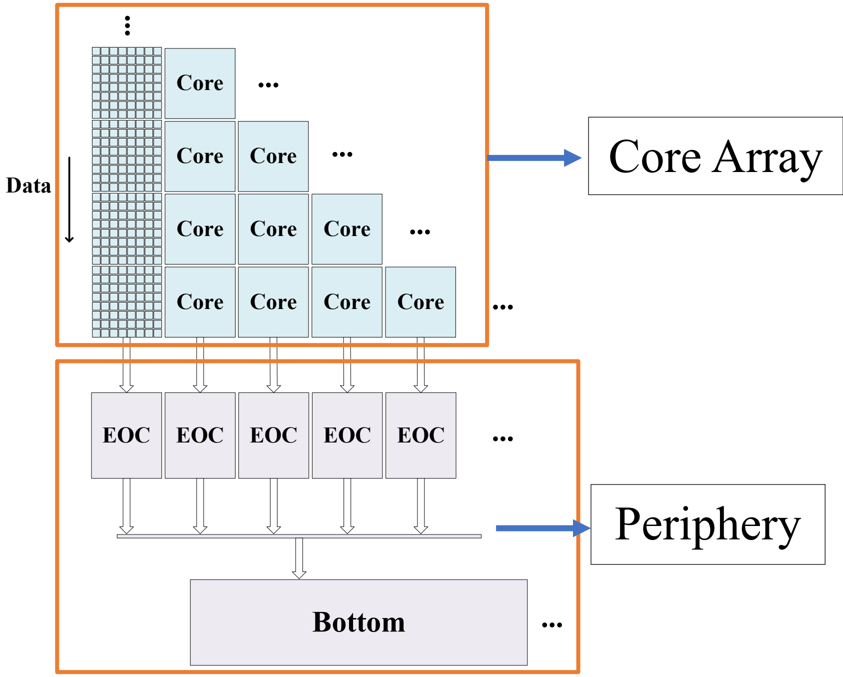
BLUE (3D tracking): no time information for CKF initial track parameters and measurements

Pixel digital chip design

Overview

- Modules for the Digital Readout ASIC:
  - **Cores Array**: 1 core includes 8×8 pixels, same as ITKPix. A specific number of pixels are in one region.
  - **Periphery**: two level readout structure: End of Column, Chip Bottom.

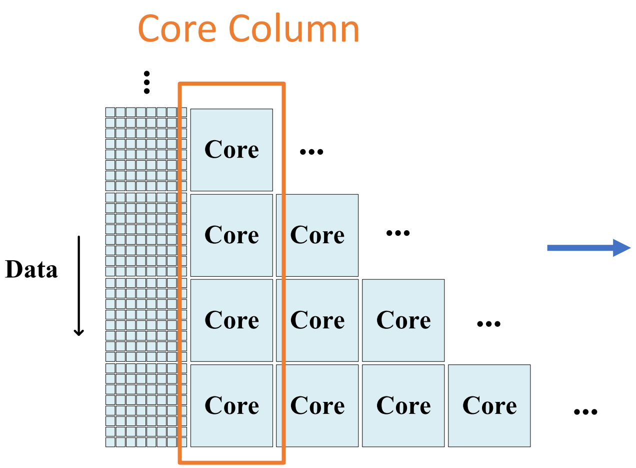




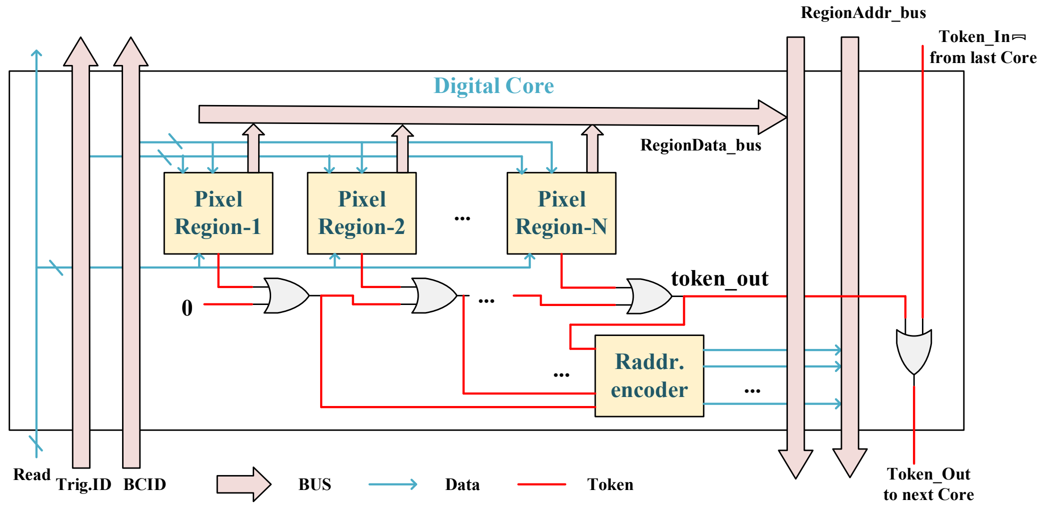
Overview

Core Array

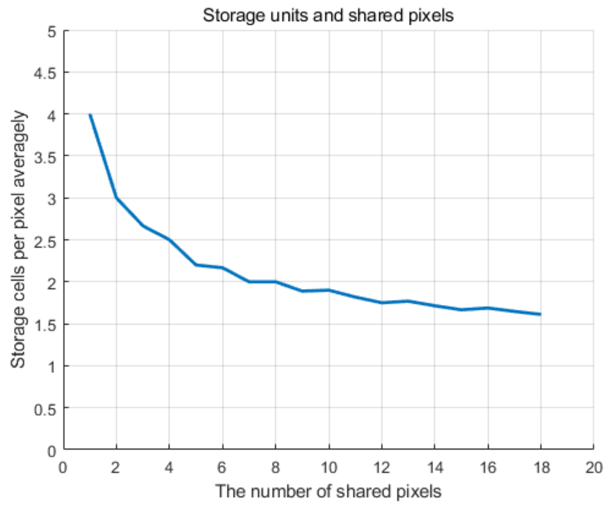
- **Function**: receiving and buffering data from TDC, and completed the trigger matching.
- Each core is same as others and they are connected as a chain
- Each core consists of several regions. All pixels in the same region **shared a same memory**
- The Shared Memory could reduce the storage space as it shown obviously in theory.



Core array



Core design



Storage space

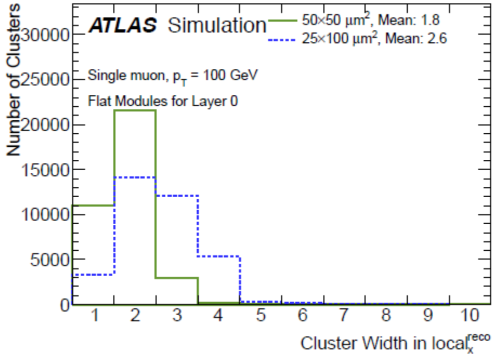
Core

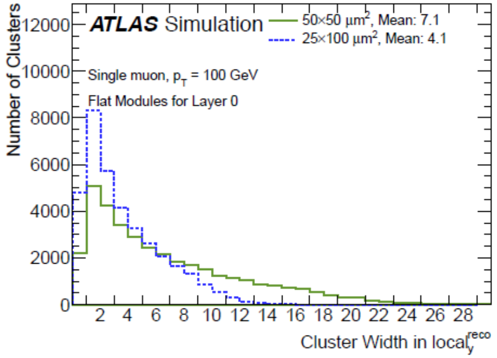
Shared memory structure:

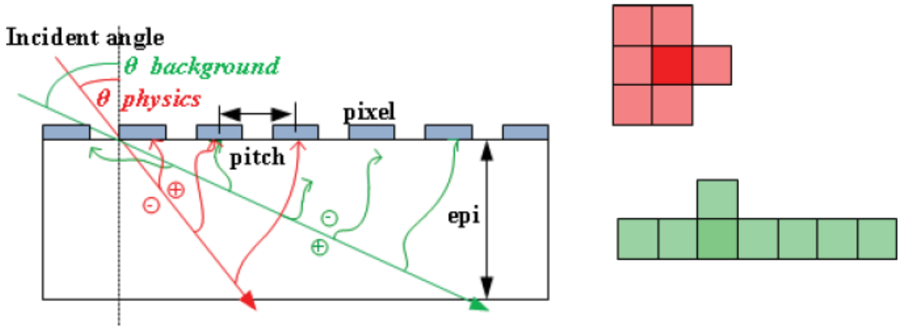
- All pixels shared are need to be independent.
- The cluster size effect is universal in the detector.

Eliminating the cluster effect:

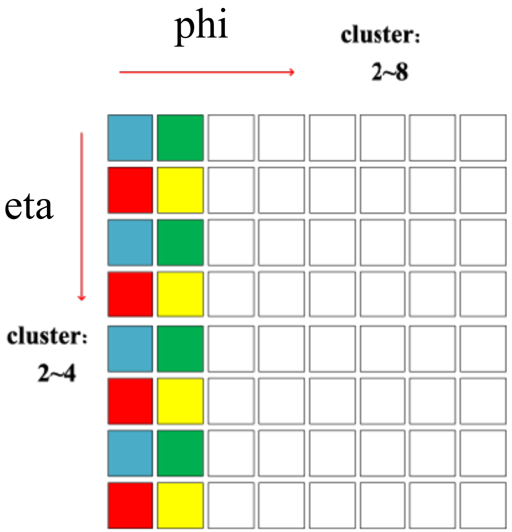
- Using a specific map to share the pixels



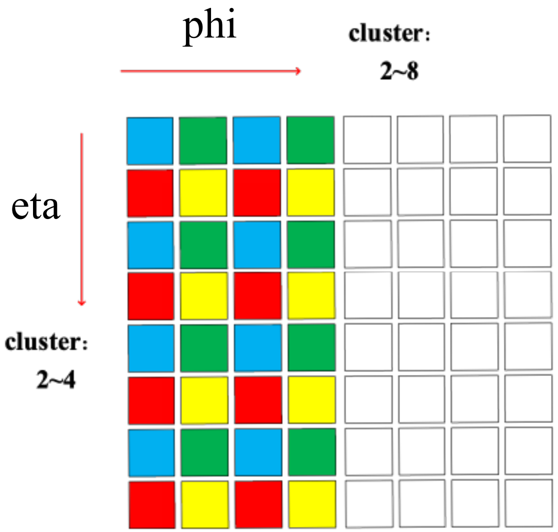




The cluster size



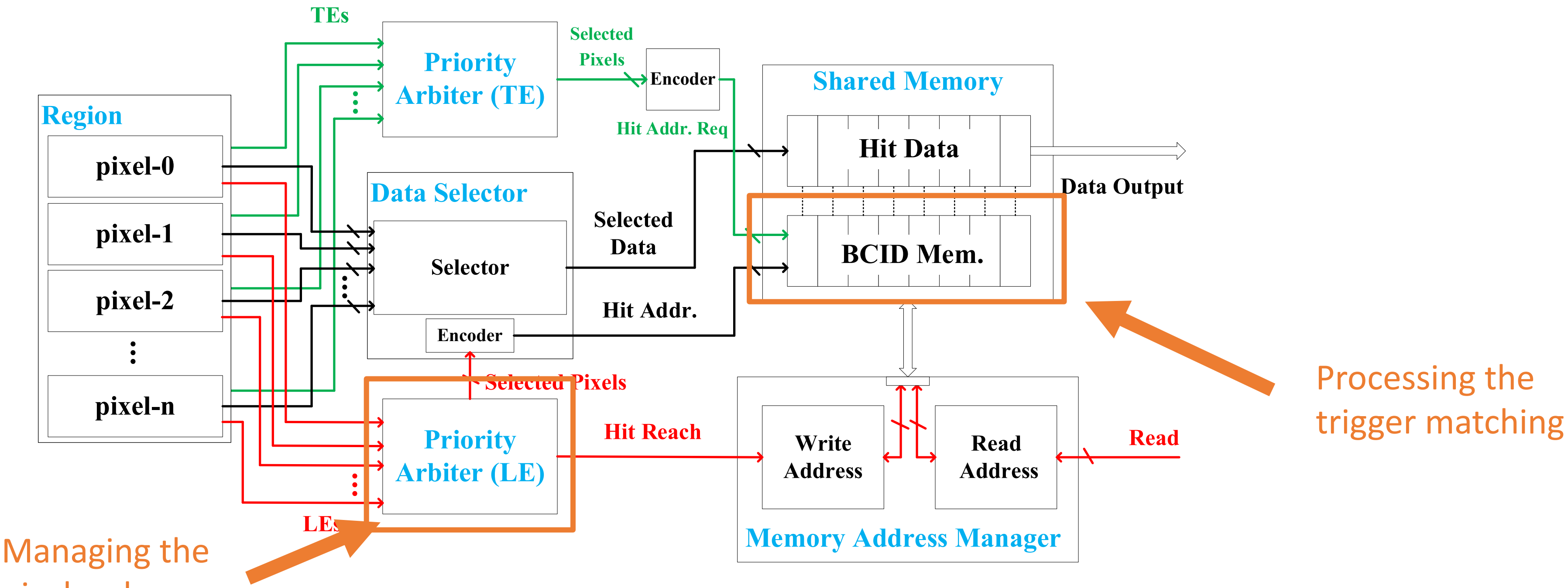
4 pixels are shared



8 pixels are shared

Region design

- Function: storing the time stamp (BCID) and TDC data, doing the trigger matching.
- Two priority arbiter for LEs and TEs, to manage the address of memory.
- One memory cell for one pixel hit. Every cell in shared memory needs a FSM to process trigger matching.



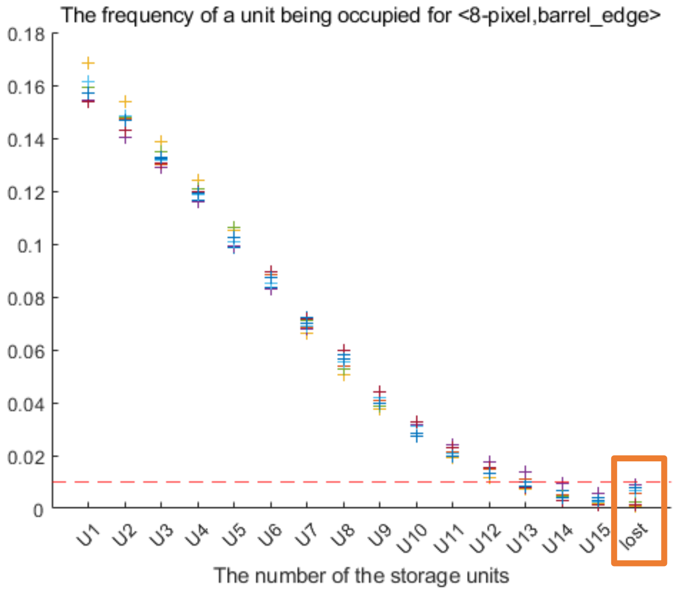
Shared memory structure (eg. 1 region)

Simulation

The simulation results (in 130 nm) (158 μm×158 μm pixel):

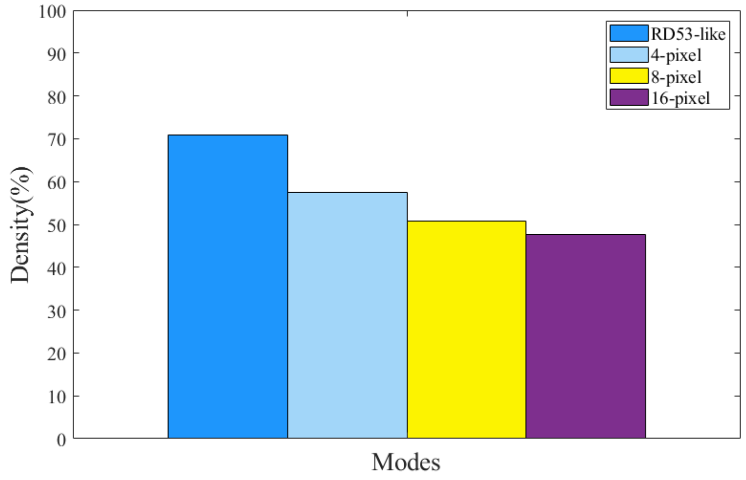
- Holding the corresponding data rate (~75 kHz/pixel)
- Lower power, Lower density.

Shared memory could reduce power and area obviously.

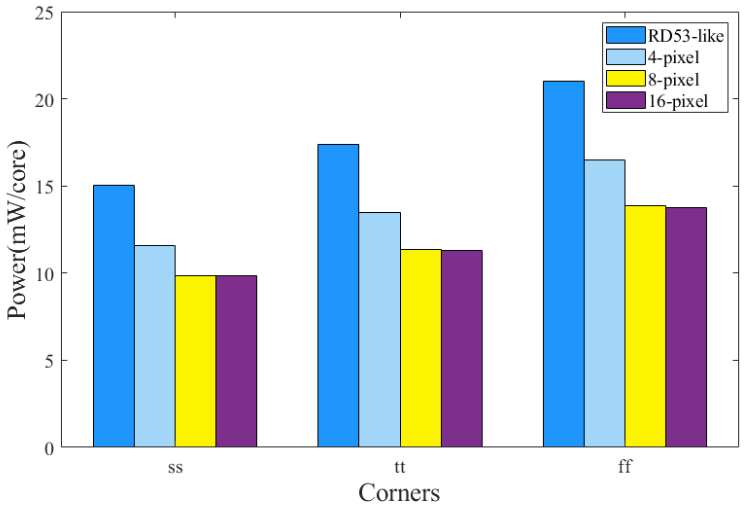


Lost rate lower than the red line (1%)

The performance of shared memory (eg.8-pixel)



The area comparison for the layout

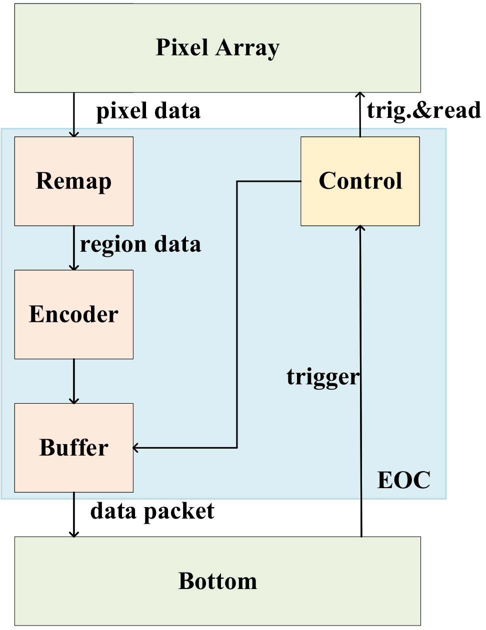


The power comparison for the layout

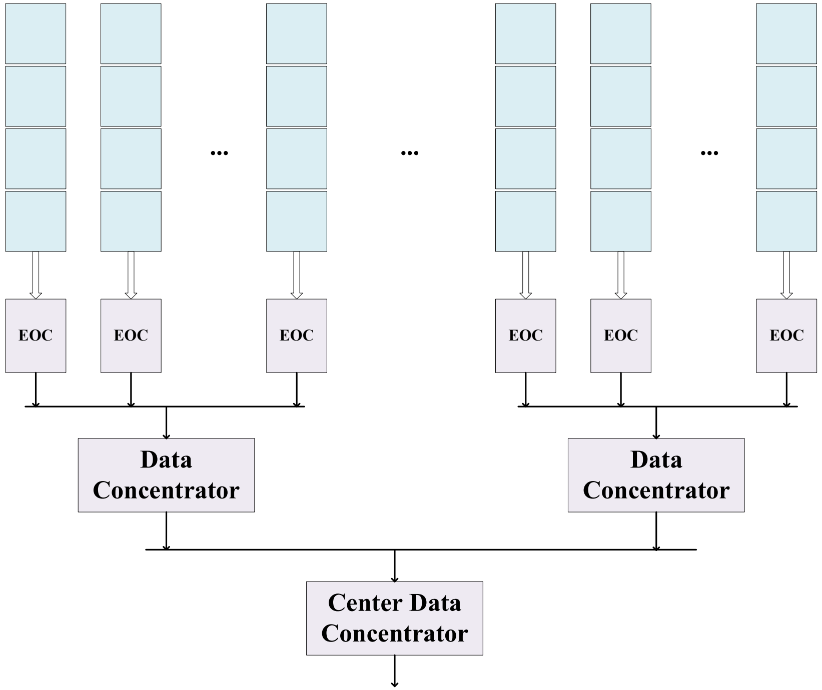
Periphery

Modules for the Periphery:

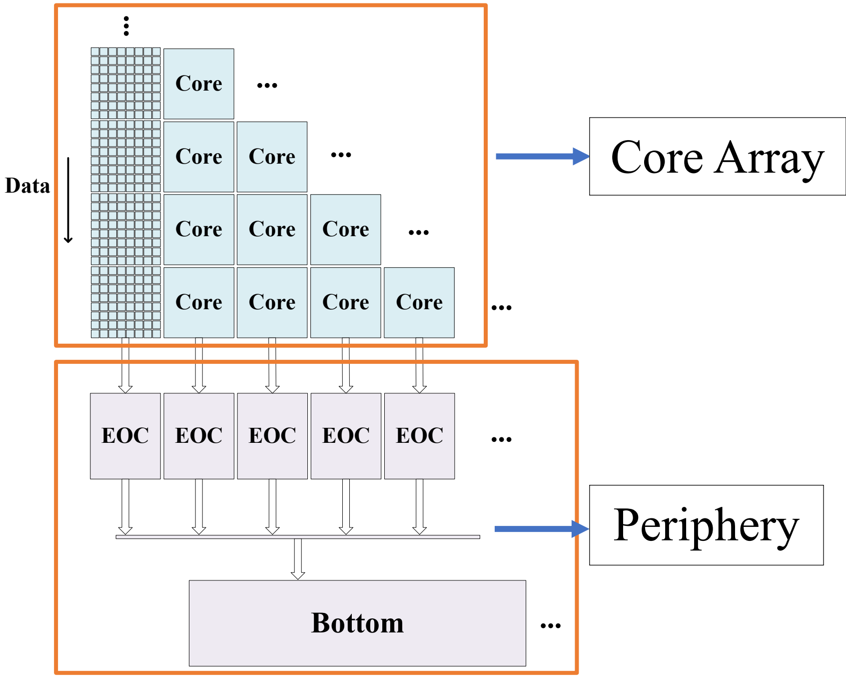
- **End of Column**: packaging and encoding and compressing the data.
- Chip Bottom: concentrating the chip data.



EOC design



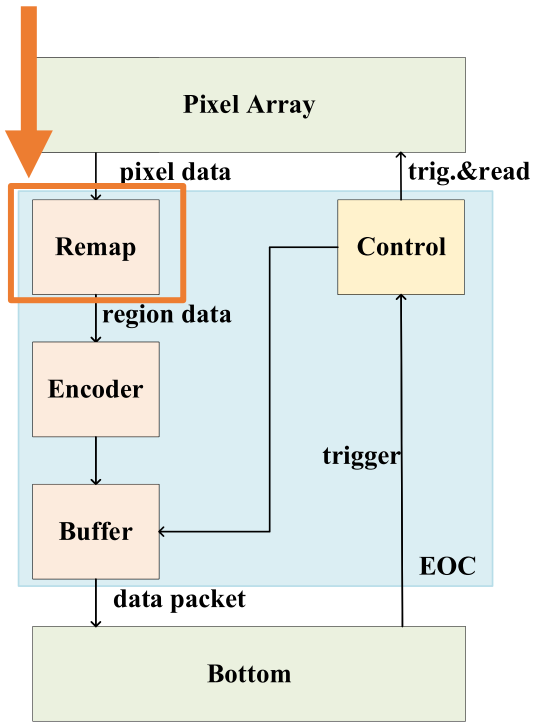
Data flow



Overview

End of Column

Remap data to make it more concentrated



EOC design

End of Column:

- controlling the readout of cores, and packaging the hit data

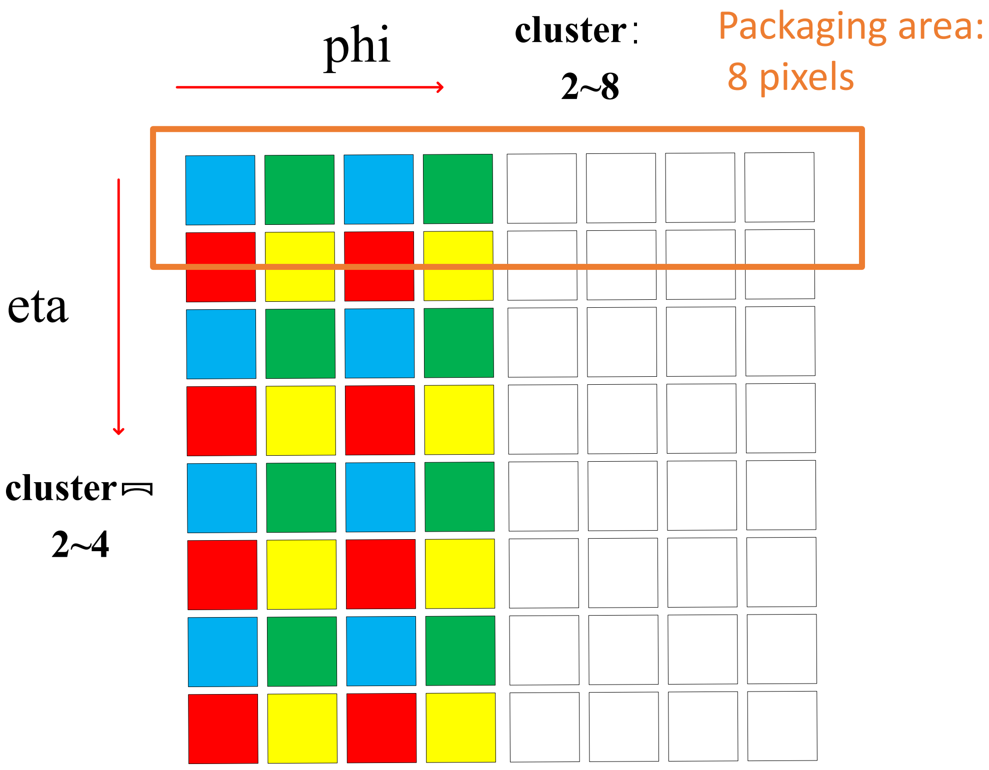
Shared memory:

- Disrupt the order of reading data from the core.
- Breaking up the concentration of clusters. Making data more dispersed which is not conducive to data compression

Remap module:

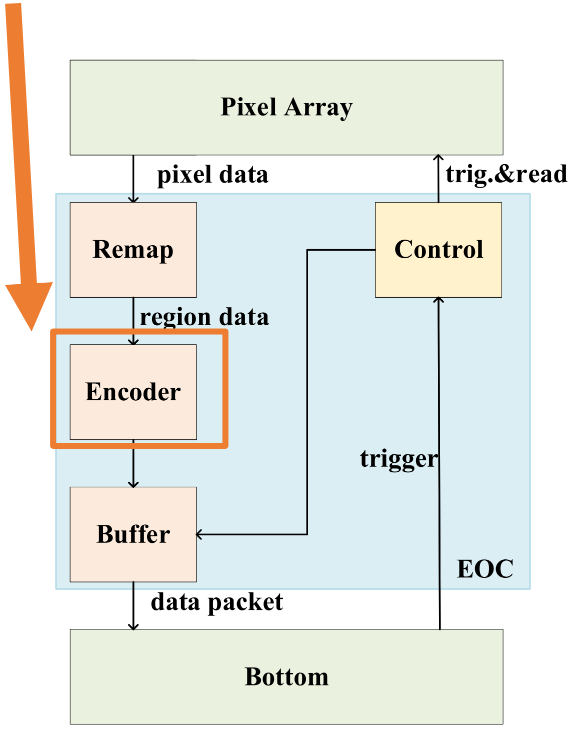
- Reorganizing the hit as the packaging area.

The packaging area use the cluster effect to concentrate the data.



End of Column

Hoffman coding to compress



EOC design

Packaging data: coarse address + hit map + hit data

Encoder:

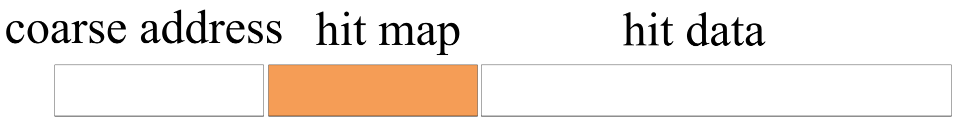
- Based on the frequency distribution of hit-map in simulation data, use Huffman encoding to compress the hit-map

Encoding schemes:

- Comparing several compression schemes to get the best

Compression efficiency:

- packaging + Huffman → ~73% (27% reduced)



A data packet

Encoding of hit map		Entropy
Bitmask		8
Binary-tree		7.79
Independent address		11.20
Multi-stages Hoffman	4-6-14	7.17
	3-6-14	7.84
	6-14	6.34

Encoding schemes

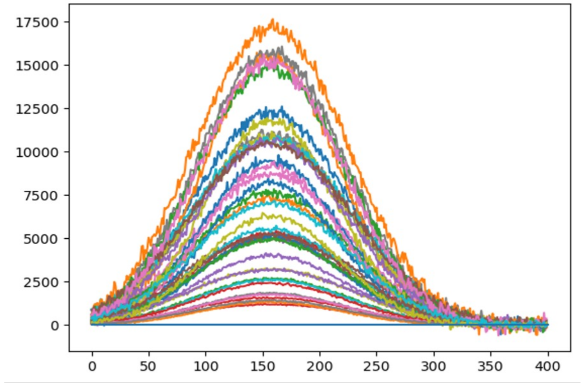
Summary

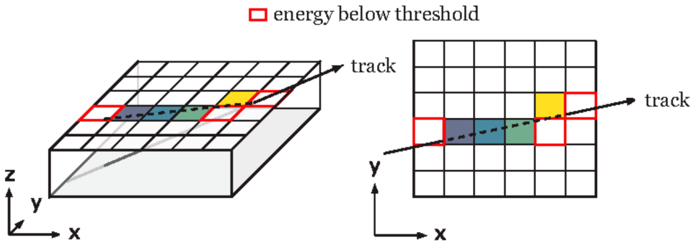
Thanks!

- 4D tracking
  - ACTS simulation & tracking workflow is powerful for 4D tracking study
  - ➢ Investigating 4D seeding & track finding/fitting algorithm with ACTS – possibilities of improvement
  - ➢ Implementing 4D tracking workflow in ATLAS Software Framework
  - ➢ Outlook: physics case using the 4D tracking + 4D vertexing + 4D flavor tagging
- Pixel digital design
  - the overview of the chip design scheme: Core array, Periphery.
  - ➢ Core array: the shared memory structure, Eliminating the cluster effect
  - ➢ Periphery:
    - End of column: remap, package, encode the hit data to compress data bits
    - Chip Bottom: concentrating the chip data

Back-ups

Timing pixel digitization in Athena





Graphs from [ACTS docs]

- Digitization
  - simulate detector response from hit simulation information
  - HIT (Geant4 simulation) -> RDO (Raw data output)
  - (position, charge…) -> (identifier/position of cells, ToT, ToA)

Geant4 hit time

smeared time
(add random gaussian distribution)

simulate pulse and calculate ToA
based on parameterized pulse waveform

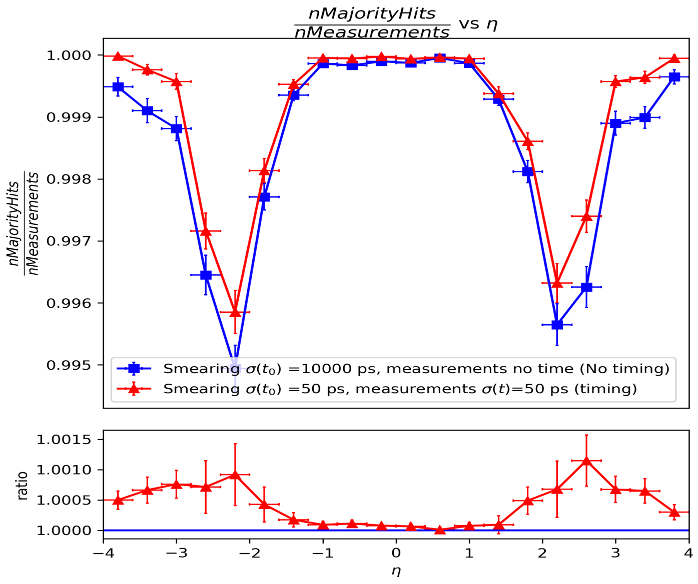
fast time digitization

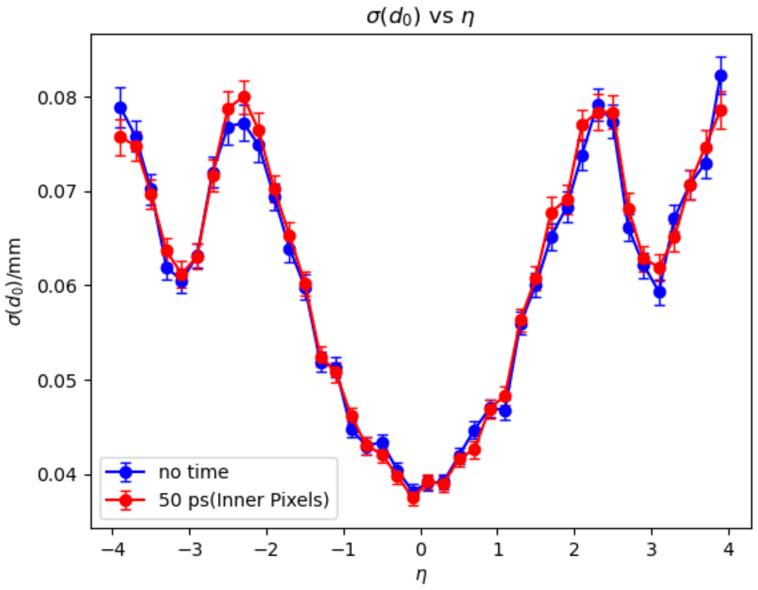
digitized time
stored in RDO

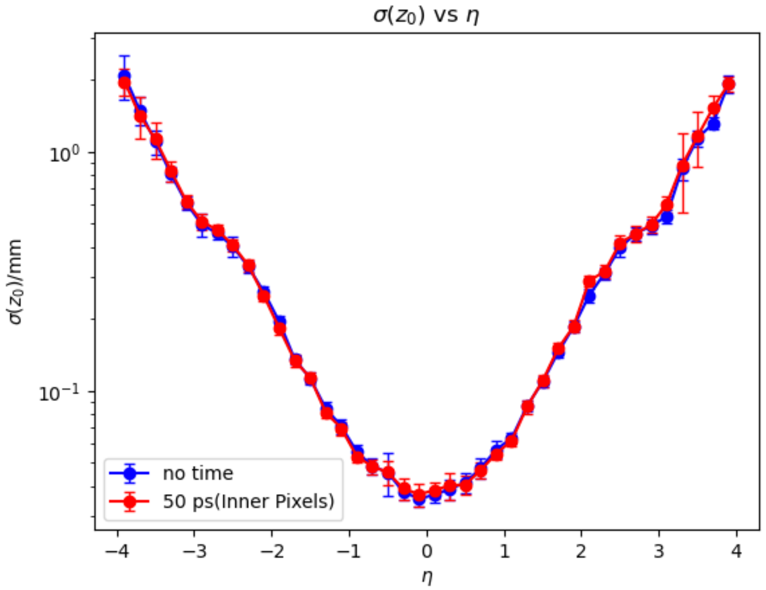
more realistic digitization
(the same method with HGTD)

example 1.732 ns 1.851 ns 00011010…

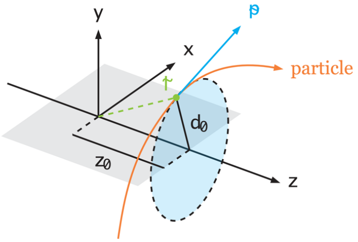
Purity and track-only impact parameter resolution



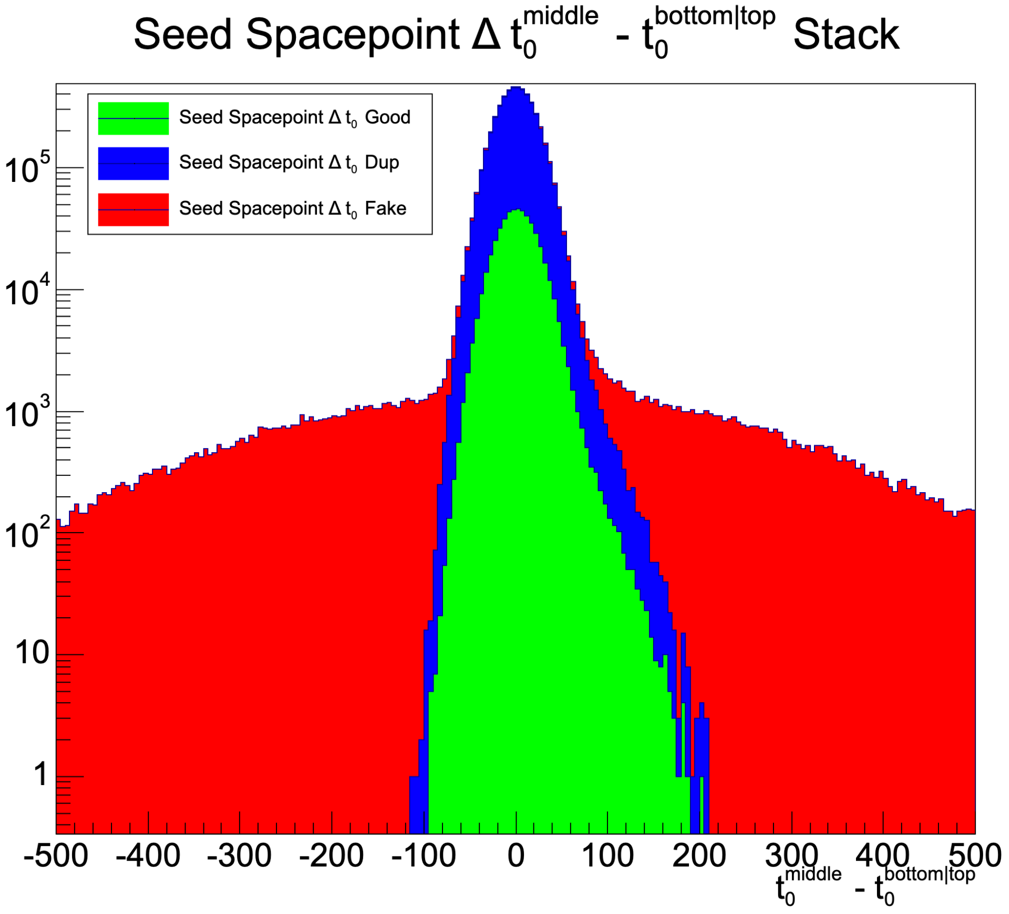




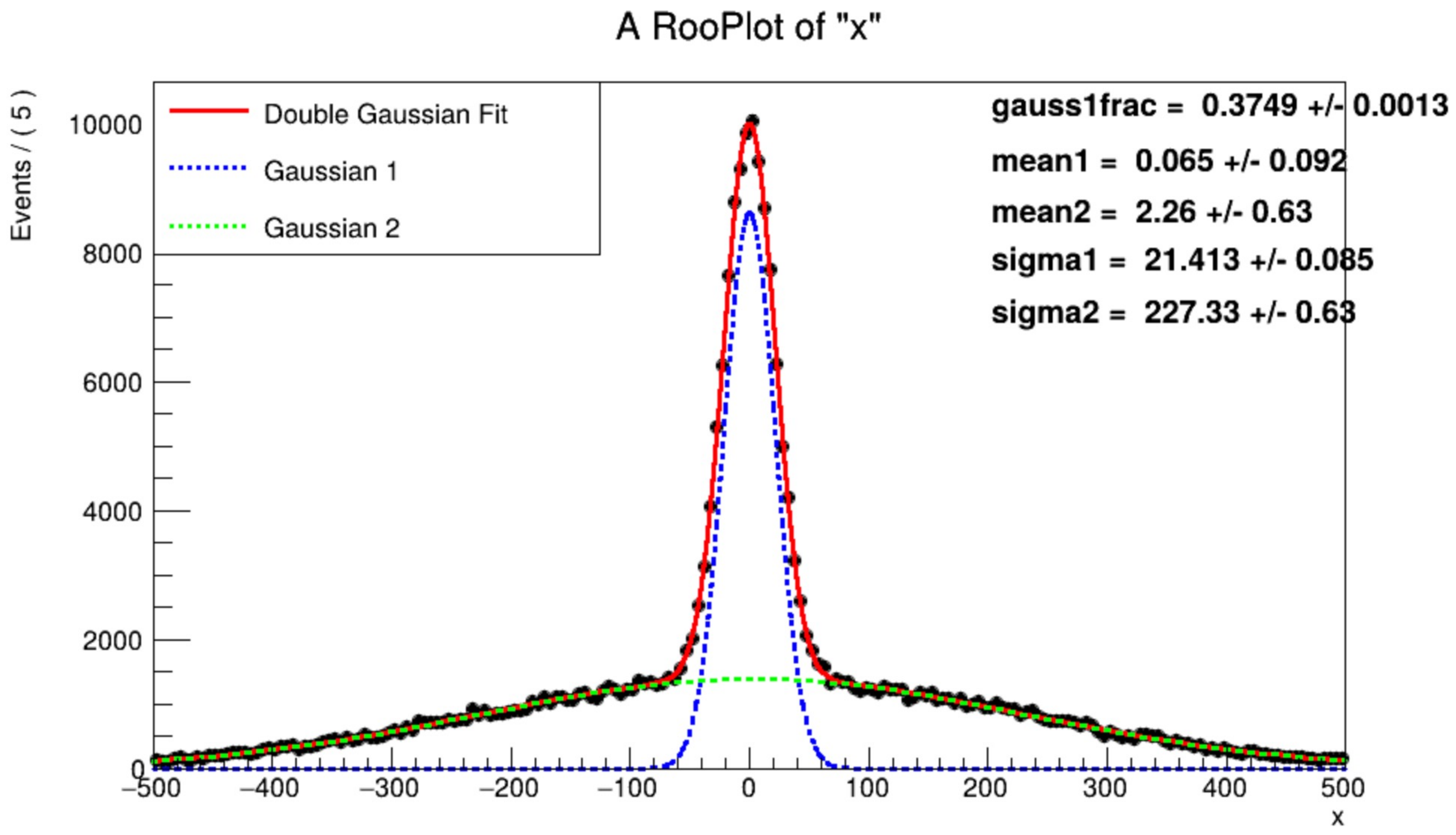
$$\frac{nMajorityHits}{nMeasurements} = \frac{\#\ hits\ that\ belongs\ to\ the\ particle\ with\ the\ highest\ fraction\ of\ hits\ on\ the\ track}{\#\ total\ measuremts\ on\ a\ track}$$



No significant change on the resolution of impact parameters d_0/z_0

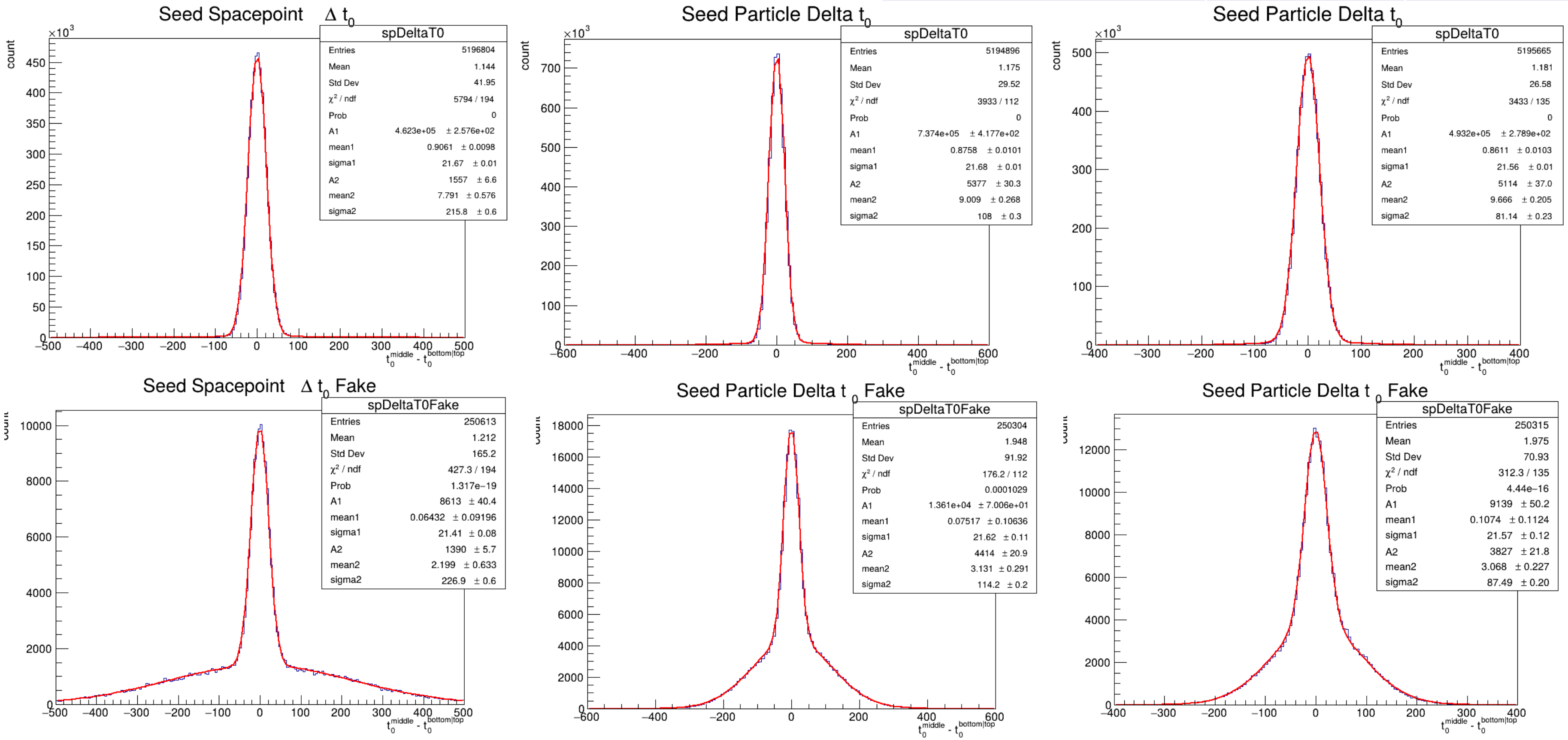


Seed Spacepoint Δ t_0^{middle} - $t_0^{bottom|top}$ Stack
Seed Spacepoint Δ t_0 Good
Seed Spacepoint Δ t_0 Dup
Seed Spacepoint Δ t_0 Fake
10^5
10^4
10^3
10^2
10
1
-500
-400
-300
-200
-100
0
100
200
300
400
500
t_0^{middle} - $t_0^{bottom|top}$



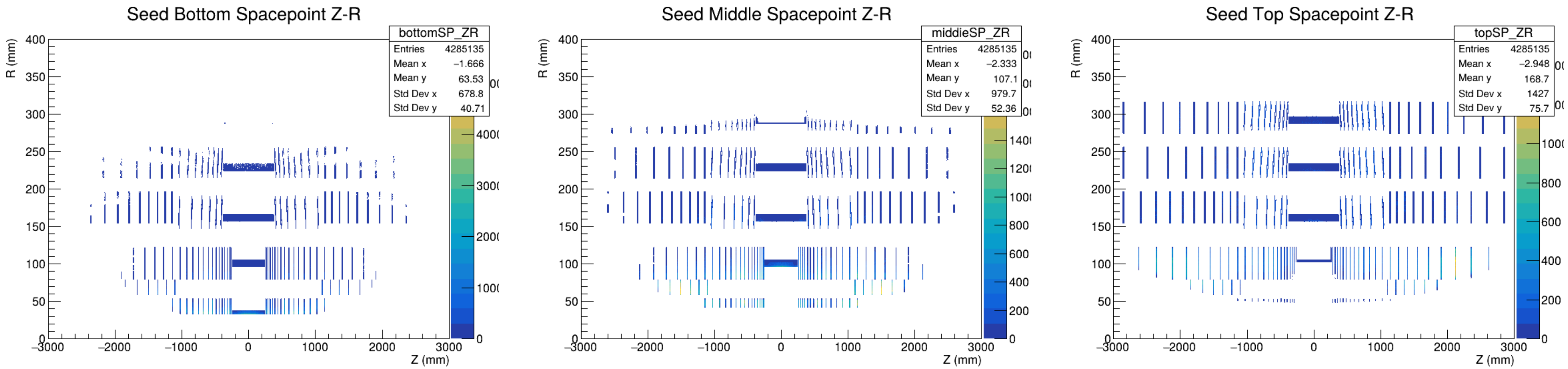
A RooPlot of "x"
Double Gaussian Fit
Gaussian 1
Gaussian 2
gauss1frac = 0.3749 +/- 0.0013
mean1 = 0.065 +/- 0.092
mean2 = 2.26 +/- 0.63
sigma1 = 21.413 +/- 0.085
sigma2 = 227.33 +/- 0.63
Events / (5)
10000
8000
6000
4000
2000
0
−500
−400
−300
−200
−100
0
100
200
300
400
500
x

Vertex distribution / ps	5000	500	200	100
σ_{time1} / mm		21.67	21.68	21.56
σ_{time2} / mm		215.8	108	81.14
fake σ_{time1} / mm		21.41	21.62	21.57
fake σ_{time2} / mm		226.9	114.2	87.49
fake σ_{time2} / ps		757	381	292
$\sqrt{2\sigma_{t\|hit}^2 + 2\sigma_{t\|vertex}^2}$ /ps		711	292	158

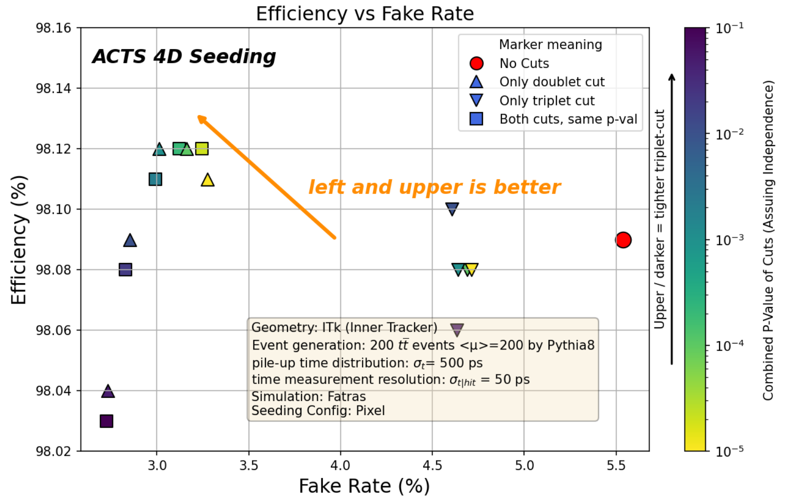


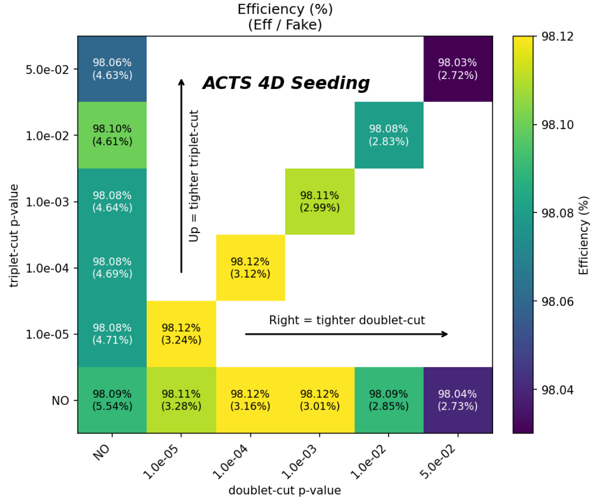
$$\sqrt{2\sigma_{t|hit}^2} = 21.199 \text{ mm}$$

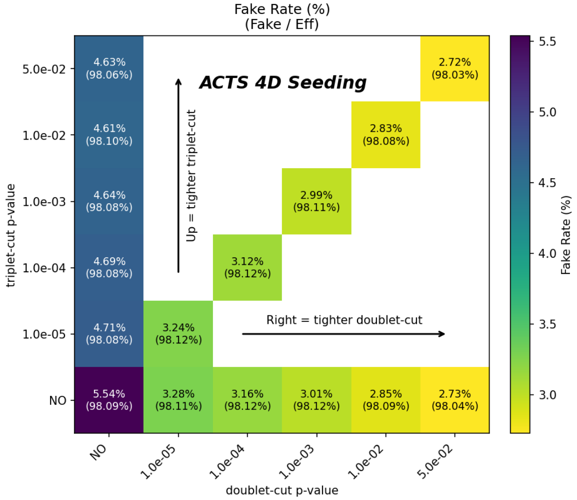
ITk spacepoint distribution

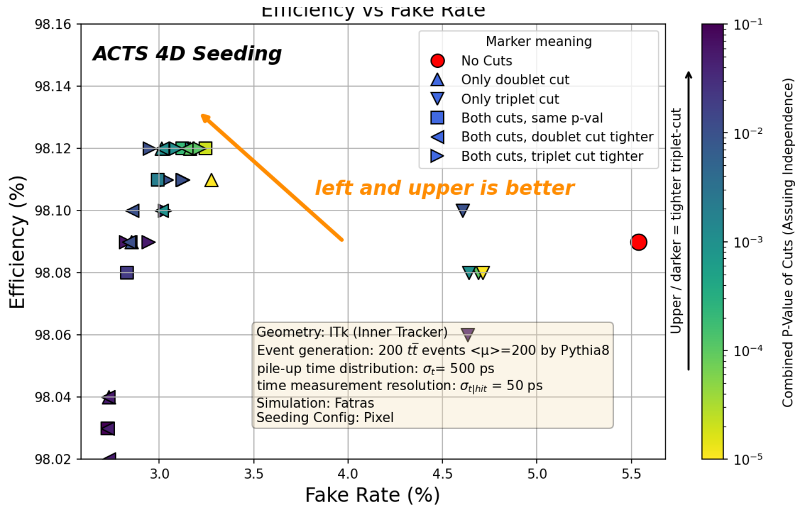


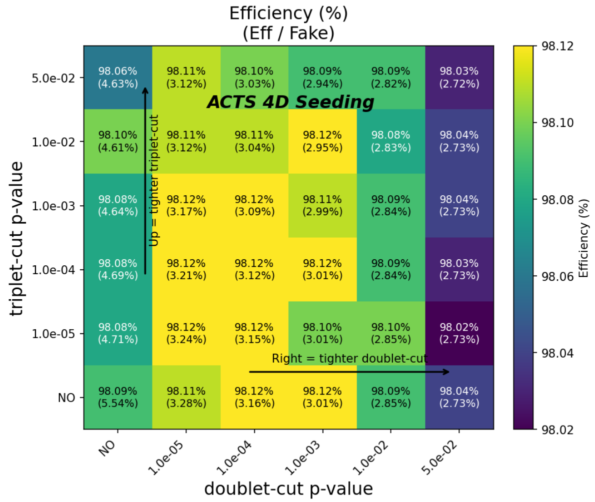
4D seeding performance with ITk

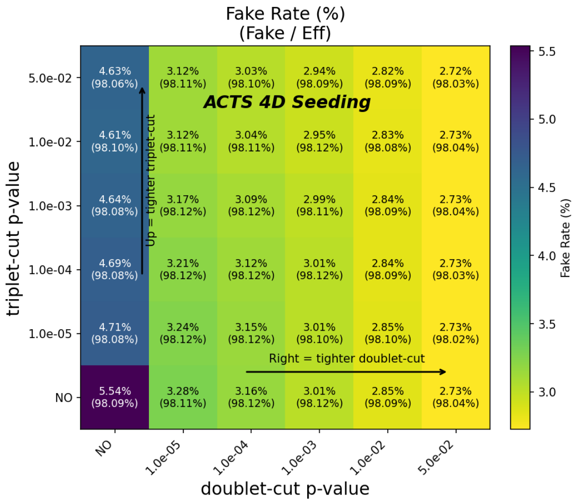






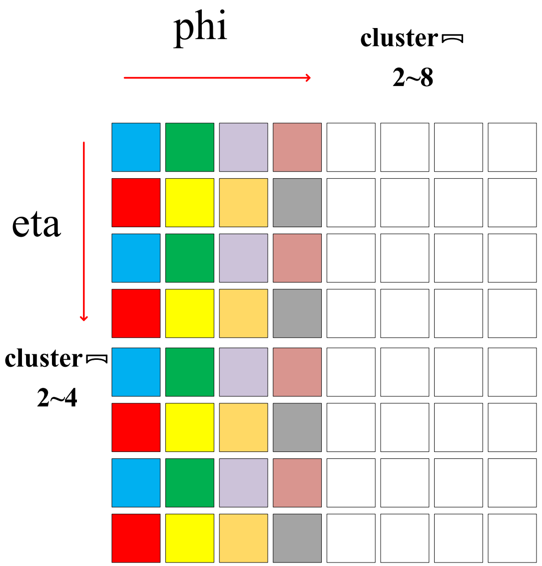




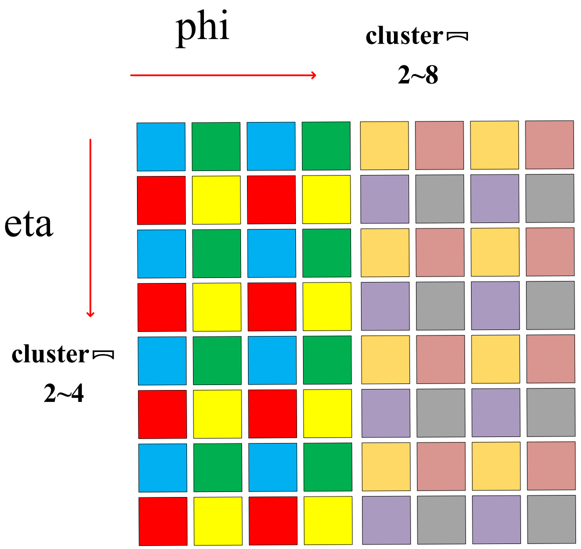


Pixels shared map

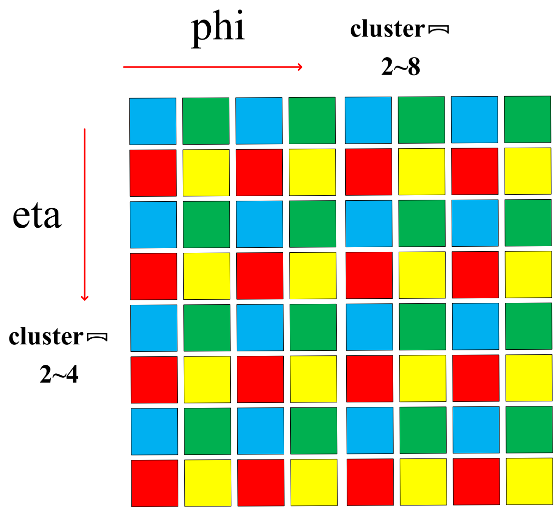
Shared pixels	Hit depth
4-pixel	9
8-pixel	15
16-pixel	27



4 pixels are shared:
16 regions



8 pixels are shared:
8 regions



16 pixels are shared:
4 regions

Huffman coding

Sort	Hit-map (bitmask)	Frequency	"6-14" mode bits
1	10101010	6.966%	6
2	01010101	6.418%	6
3	01000000	6.183%	6
4	00000010	5.784%	6
5	00000011	5.175%	6
6	11000000	4.533%	6
7	00000001	3.960%	6
…	…	…	
64	11101010	0.173%	6+8
65	11110100	0.173%	6+8
66	00110101	0.150%	6+8
…	…	…	

Encoding of hit map		Entropy
Bitmask		8
Binary-tree		7.79
Independent address		11.20
Multi-stages Hoffman	4-6-14	7.17
	3-6-14	7.84
	6-14	6.34

Encoding schemes

Huffman Encoding schemes (eg. 6-14 mode):

- For the first 63 high-frequency hit-maps: use 6 bits to encode.
- Since 6 bits correspond to 64 combinations, the last combination will be used as the header for other 193 hit-maps.
- For the other 193 hit-maps: 6 bits header + the original bitmask hit-map (8 bits)

Bitmask: 1 bit for 1 pixel,
1 for hit and 0 for not