# Raw Banks

## SuperPixel Logic

Angelo Carbone, Serena Maccolini, Tommaso Fulghesu

Bologna

March 1, 2021

# SuperPixel

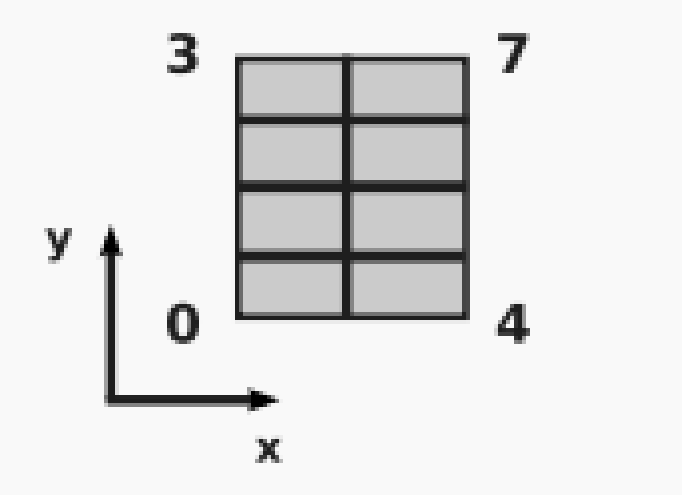



Figure: SuperPixel format

- 8 neighbouring pixels
- Orientation depends on the Sensor to which it belongs to
- First clustering techniques
- FPGA-friendly using Raw Bank

# Raw Bank

A raw bank contains information relative to each hit SuperPixel of a Sensor (192 x 128 SP)

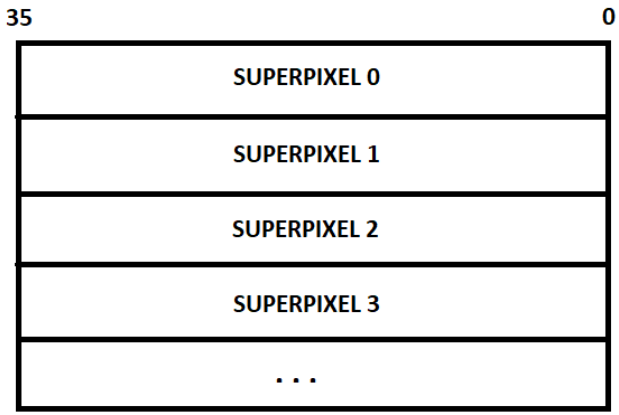



Figure: Raw bank format

# SP word

Each raw bank is composed by a 36-bit word


- 1 bits for the "hint"
- 12 bits for the SP time information (25ns sampling, 6ps time resolution)
- 8 and 7 bits for SP spatial position
- 8 bits for px inside SP

# SP Time distribution

Time coordinate associated to a superpixel is O(ns) $\rightarrow$ 10 bits

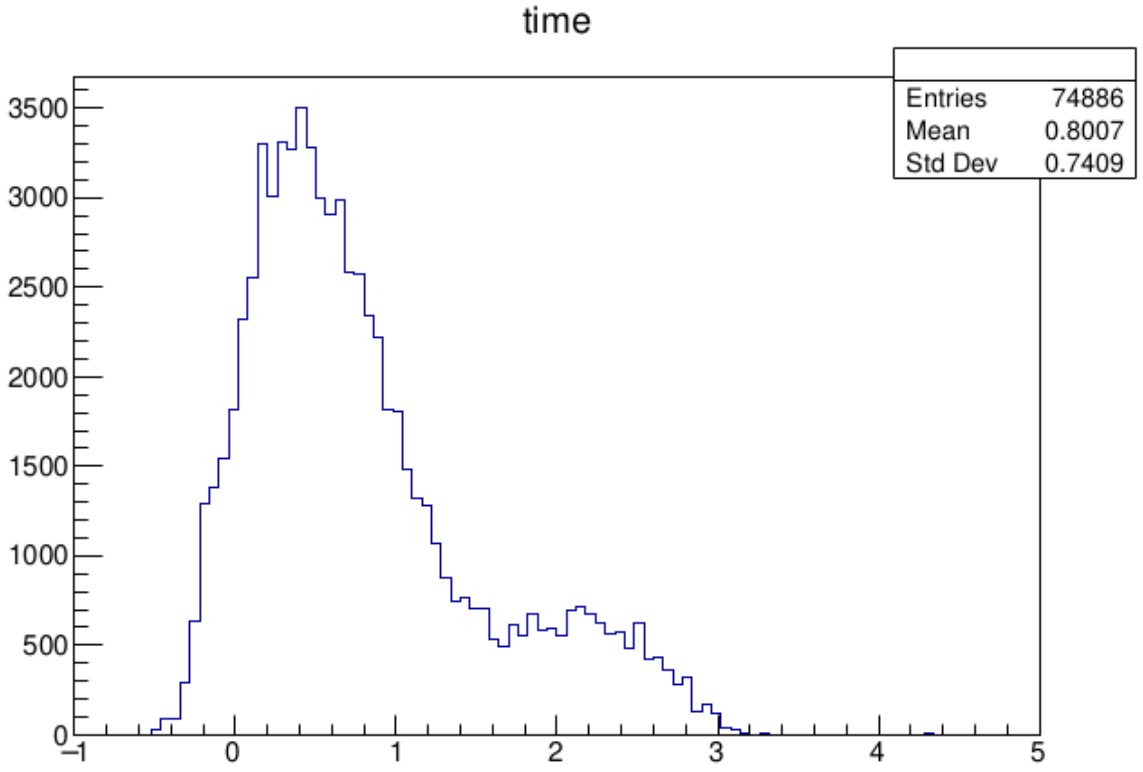



Figure: Time associated to a superpixel wrt event time

# SP time assumption

The time associated to each pixel is given by two terms:

$$t_{px} = t_{ov} + \frac{1}{v_{part}} * |z_{px} - z_{ov}| * \frac{p}{p_z}$$

**Assumption:** $t_{SP} = t_{px_{first}}$

**TEST:** RMSE for each SP

**RESULT:** Few number of SPs with more than 1 hit pixels have RMSE $\neq 0$ (264 entries vs 74886 SP)

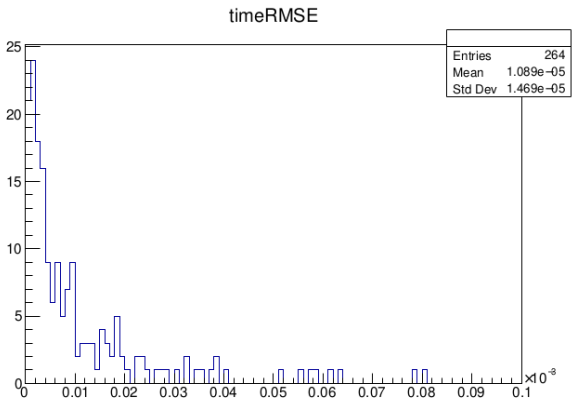



Figure: RMSE$\neq$ 0 associated to a superpixel with more than one ON pixel

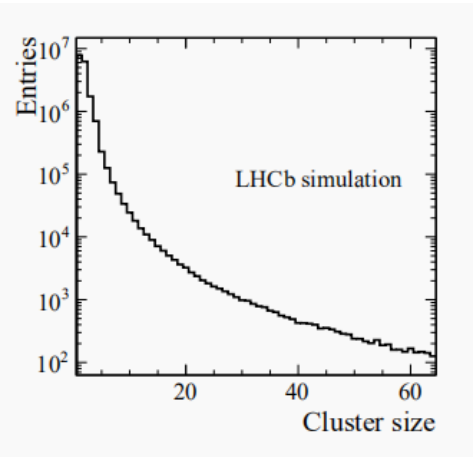
# Next steps

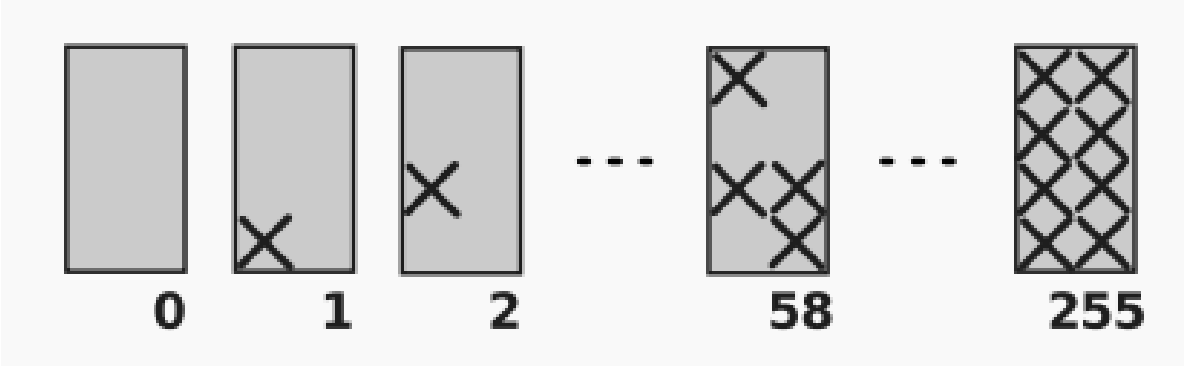
- Test directly with FPGA;
- Build raw bank even using pixel logic;
- Clustering algorithms based on the SP.

# FPGA Clustering

WHY?

Large amount of clusters are inside an isolated SP $\rightarrow$ It is possible to use a look-up table (from 0 to 255) to see the active pixels inside the SP and creates the clusters (2x faster)

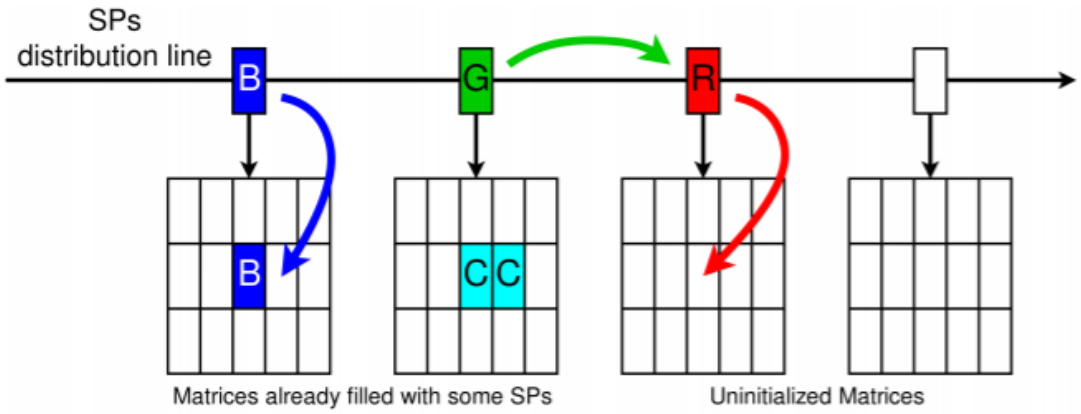


# FPGA Clustering

HOW IT WORKS?

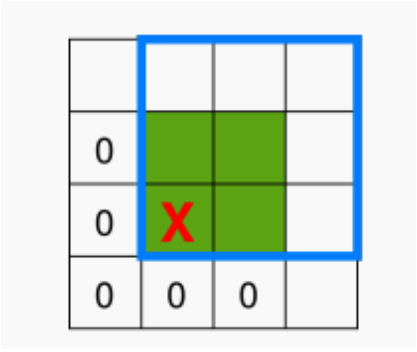
STEP 1: Matrices with dimension $5 \times 3$ SPs ($10 \times 12$ pixels) at every clock cycle change SP input.



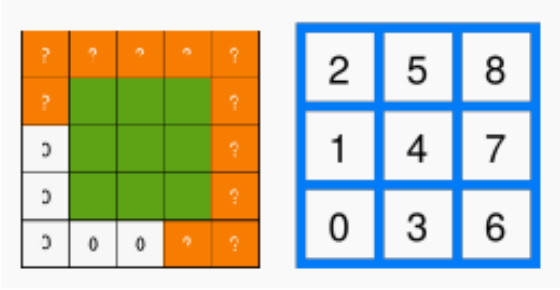
# FPGA Clustering

HOW IT WORKS?

STEP 2: Construction of cluster candidate from the **SEED** pixel.



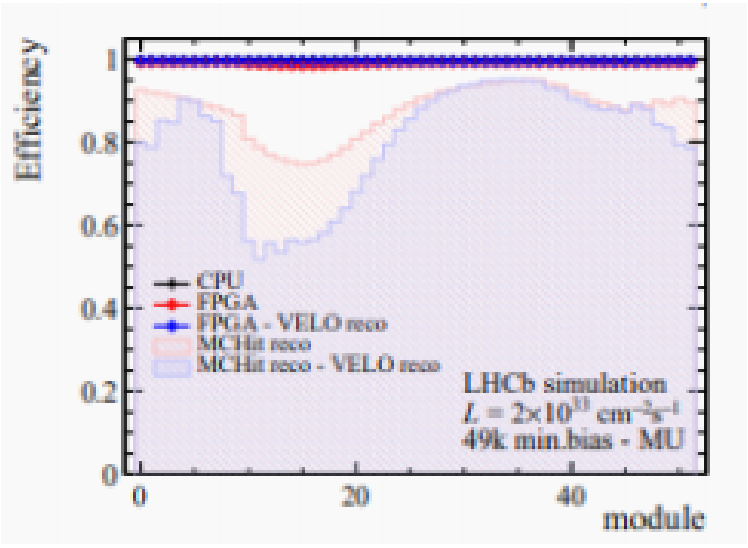

**TOPOLOGY**

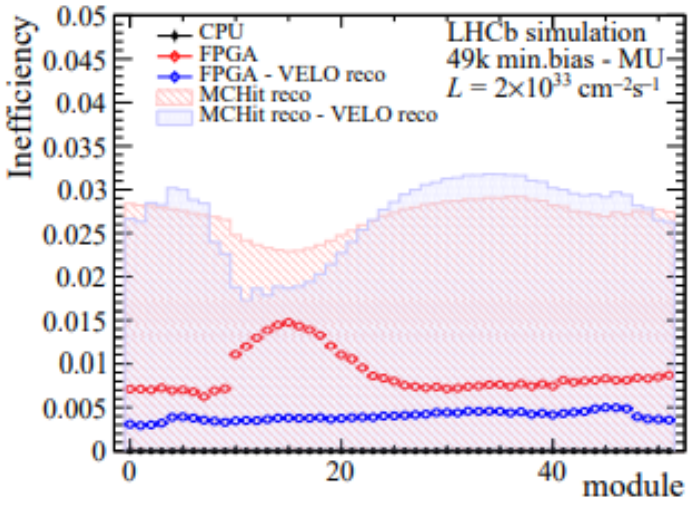



To every seed pixel is associated a lookup table with *flags* which characterize the cluster

| Meaning | Flag |
|---|---|
| Isolated | 101 |
| Overflow | 100 |
| Self-contained & edge | 011 |
| Self-contained & not-edge | 010 |
| Not-self-contained & edge | 001 |
| Not-self-contained & not-edge | 000 |

# Clustering efficiency

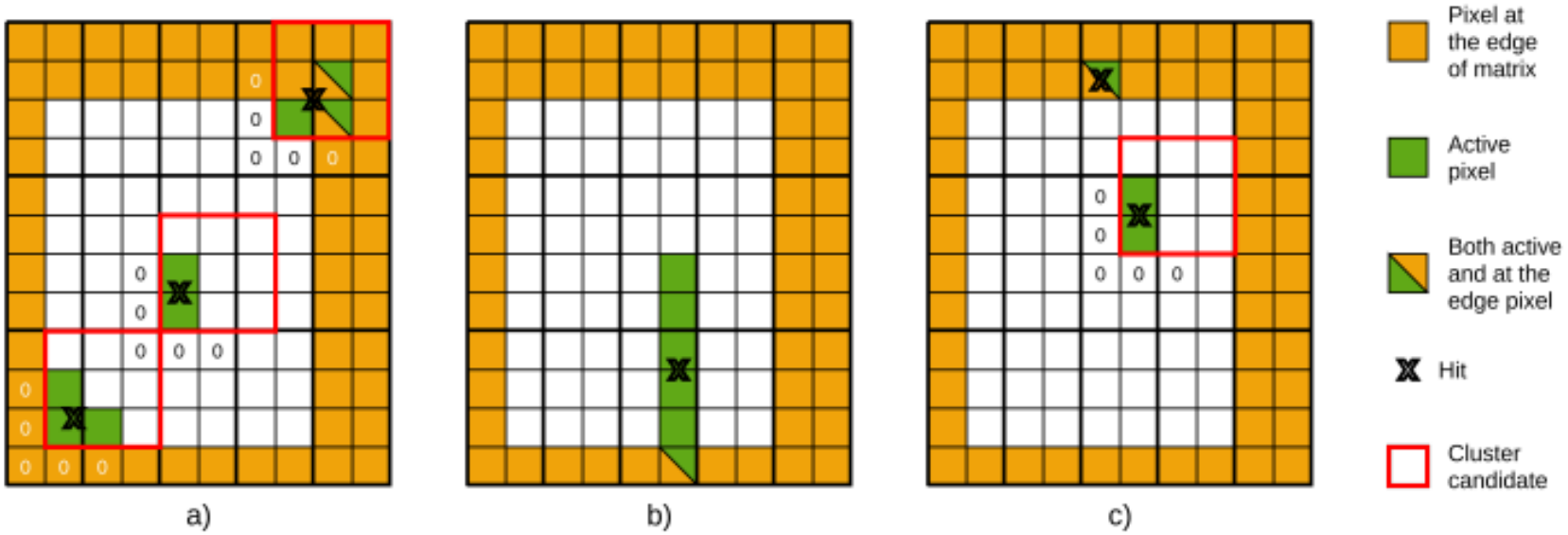



$$\epsilon = \frac{N_{MC_{linked}}}{N_{MC}}$$

$N_{MC_{linked}} = \#$ hits with linked reconstructed cluster.
$N_{MC} = \#$ reconstructible hits

# Cluster Inefficiency




FPGA efficiency depends on VELO occupancy

- larger prob. of non isolated SPs and larger cluster dimensions
- larger prob. of overflow

# Clustering based on Raw Bank

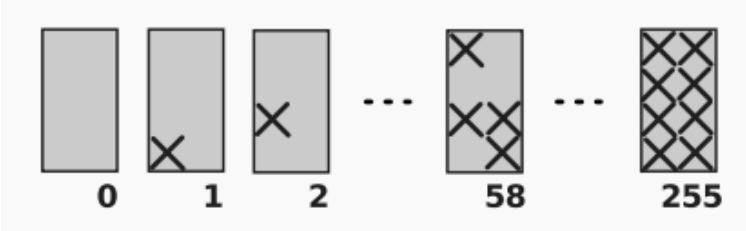
Each raw bank is composed by N 36-bit SP word:


The HINT specifies the kind of SP

- If HINT $= 0$ : ISOLATED SP (ISP) $\rightarrow$ Clusters are directly retrieved from SP.
- If HINT $= 1$ : NON-ISOLATED SP $\rightarrow$ Save all SPs in a *SP-CACHE* .

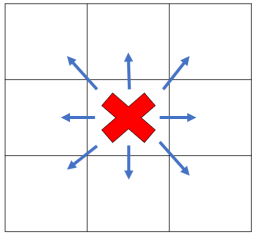
# Isolated SP

## FPGA-based




8-bit Look Up Table to determine the clusters associated to the ISP

## CPU-based



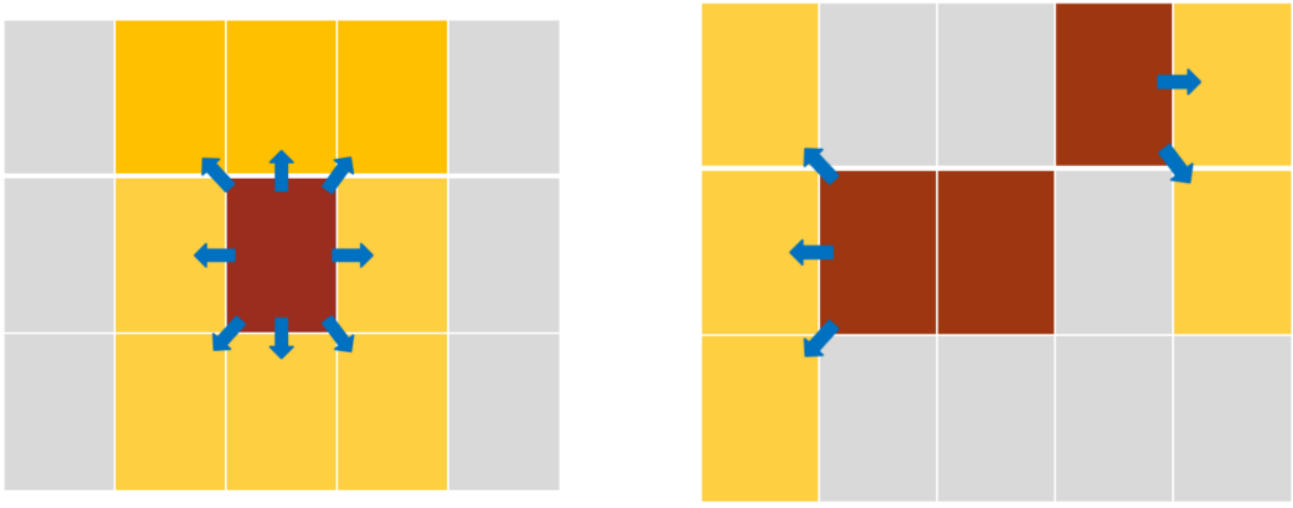
Recursive method to determine clusters

# of Reconstructed clusters:
FPGA method: 1772
CPU method: 1772

$\epsilon_{ISP} = 100\%$

# Neighbouring SP

1. Construction of the **reading matrices** ($5 \times 3$ SP) from SPs in the *SP-CACHE*



2. Research of the **seed pixel** and construction of the **cluster candidate** (CC)

# Neighbouring SP

## FPGA-based

9-bit Look Up Table to determine the clusters associated to the CC

## CPU-based

Recursive method to determine clusters

# of Reconstructed clusters:
FPGA method: 1201
CPU method: 1206

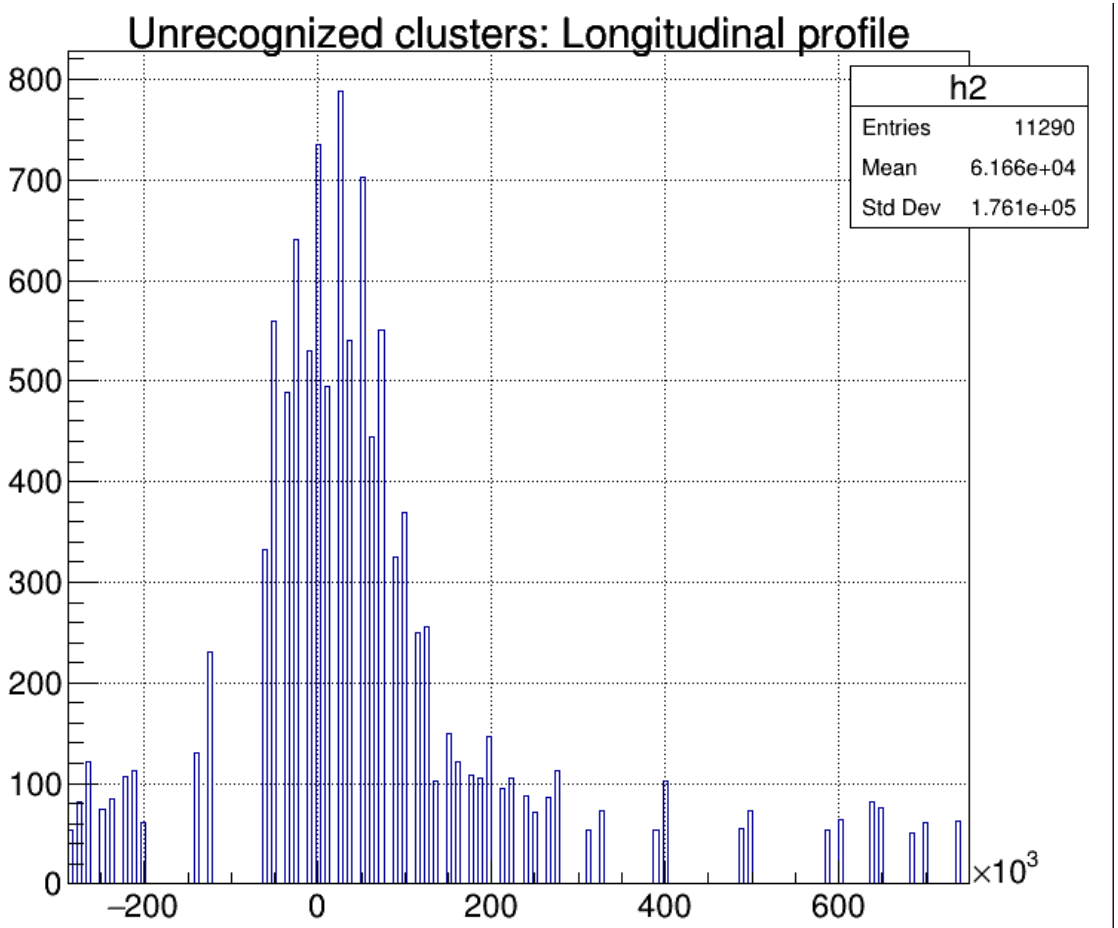
$$\epsilon_{NSP} = 96.2\%$$

TOTAL:
$$\epsilon = \frac{2932}{2978} = 98.5\%$$

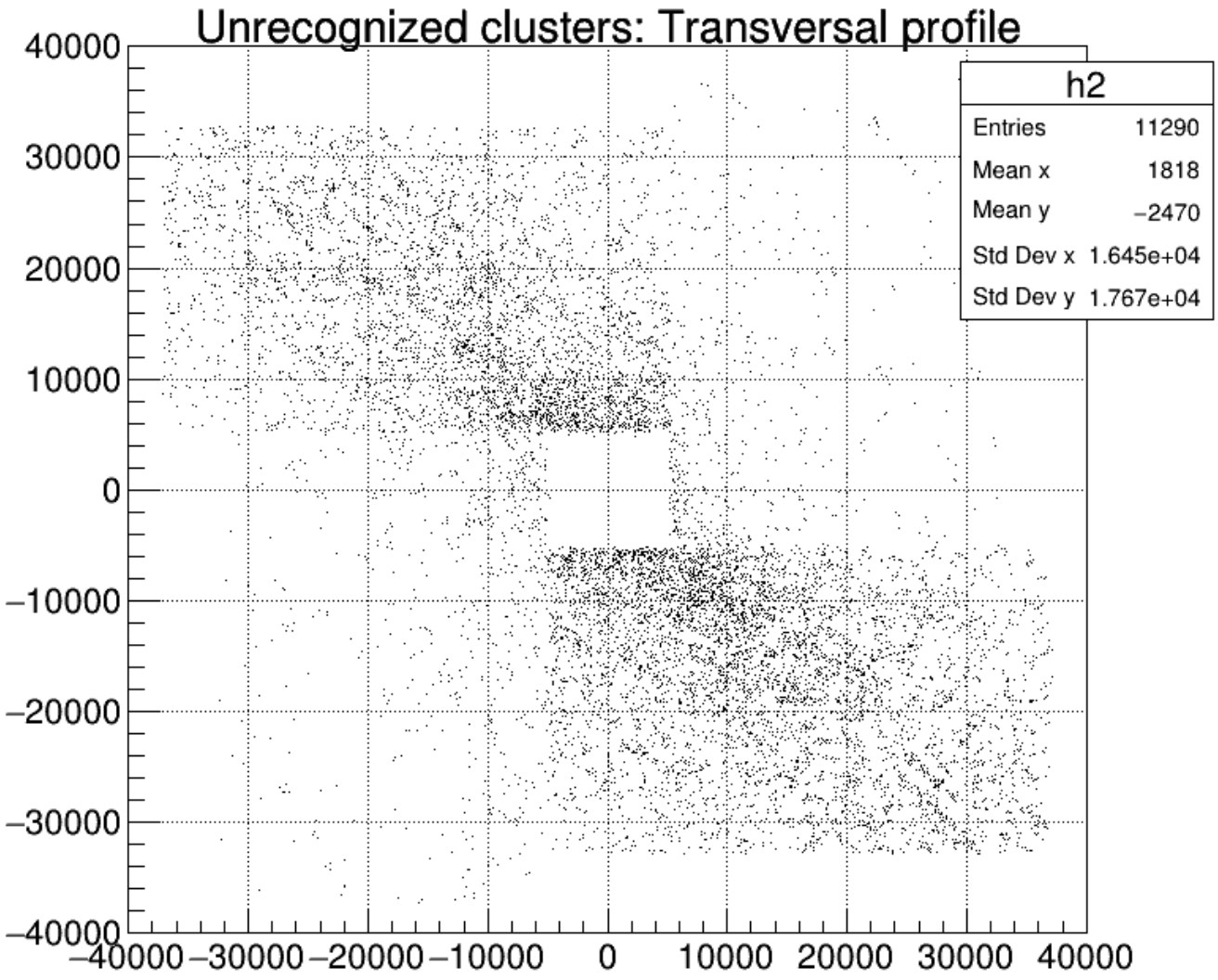
# Inefficiencies

On large statistics (737541 clusters), we obtained:

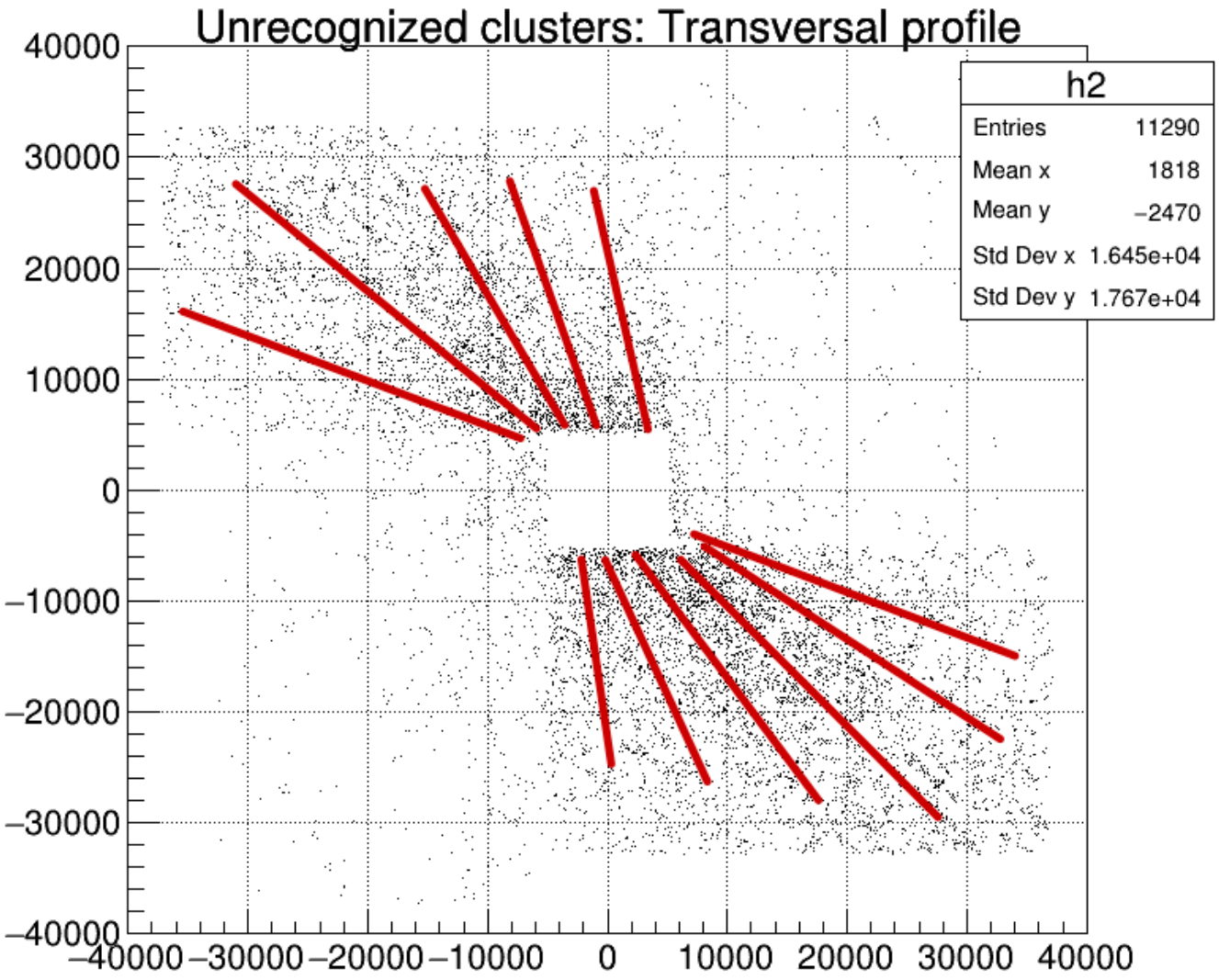
$$\epsilon = \frac{\#\text{ FPGA-reco clusters}}{\#\text{ CPU-reco clusters}} = 98.5\%$$



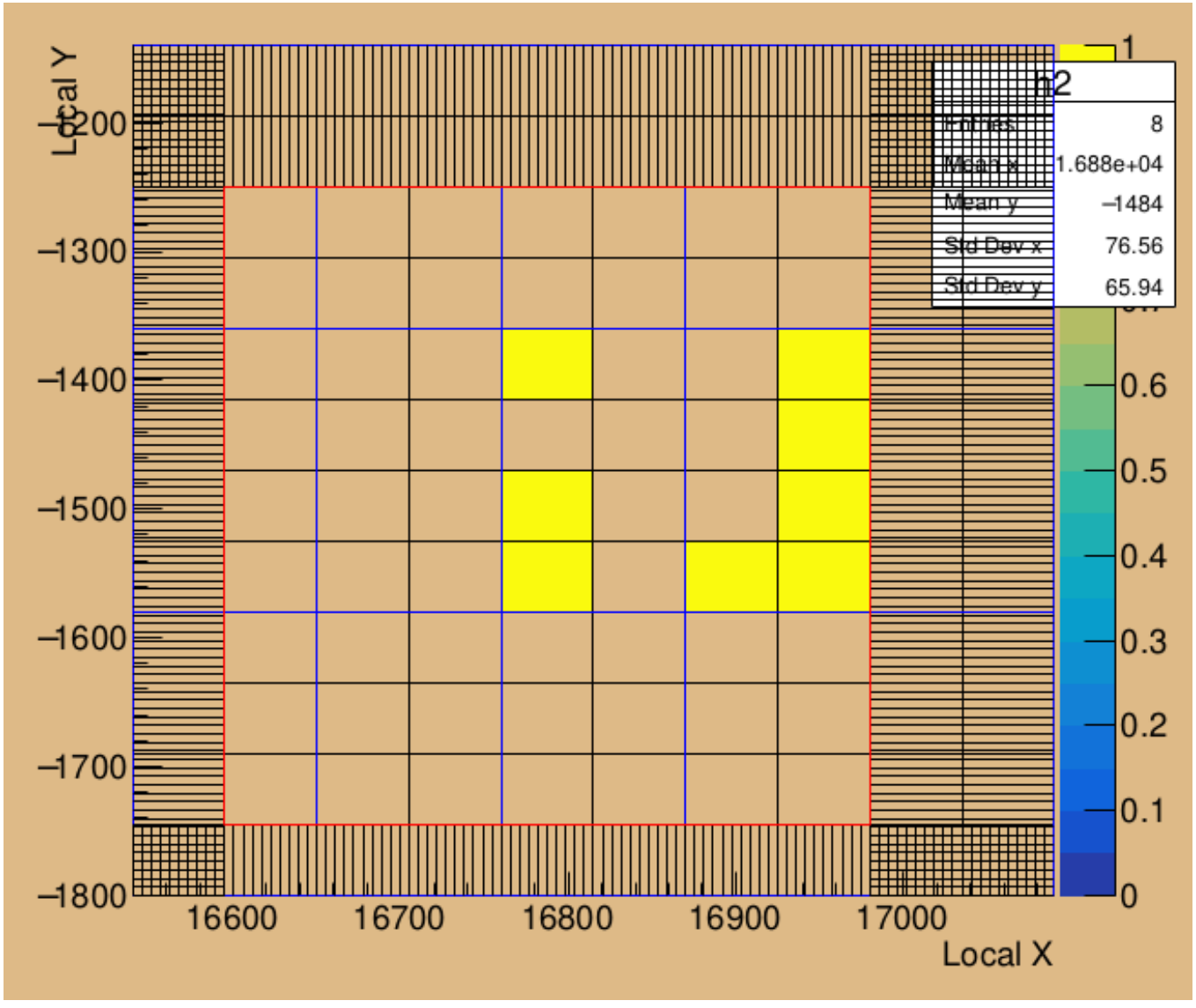
# X-Y Plane


Unrecognized clusters: Transversal profile
h2
Entries 11290
Mean x 1818
Mean y −2470
Std Dev x 1.645e+04
Std Dev y 1.767e+04
40000
30000
20000
10000
0
−10000
−20000
−30000
−40000
−40000 −30000 −20000 −10000 0 10000 20000 30000 40000

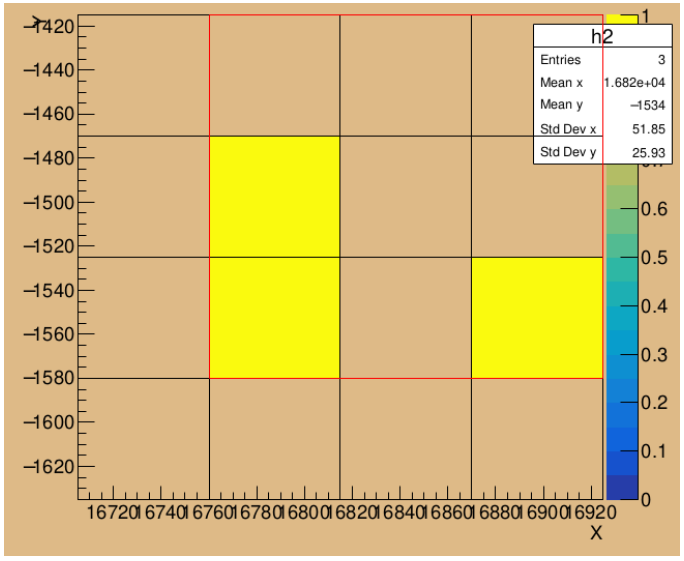
# Problem: region asymmetry

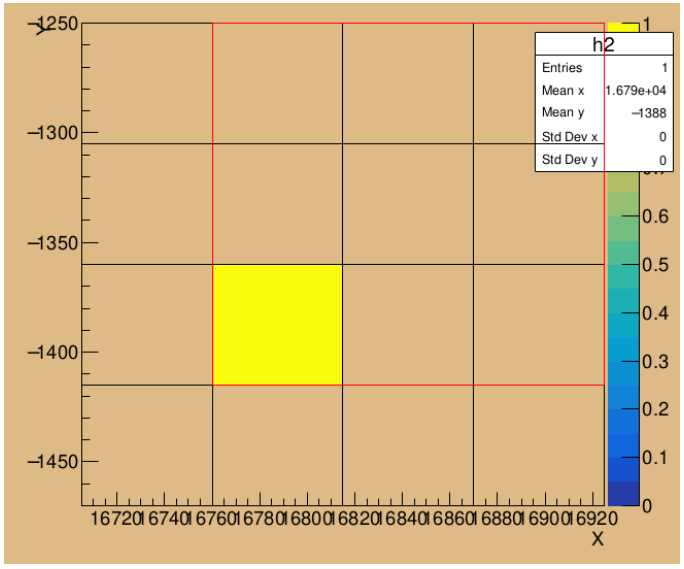


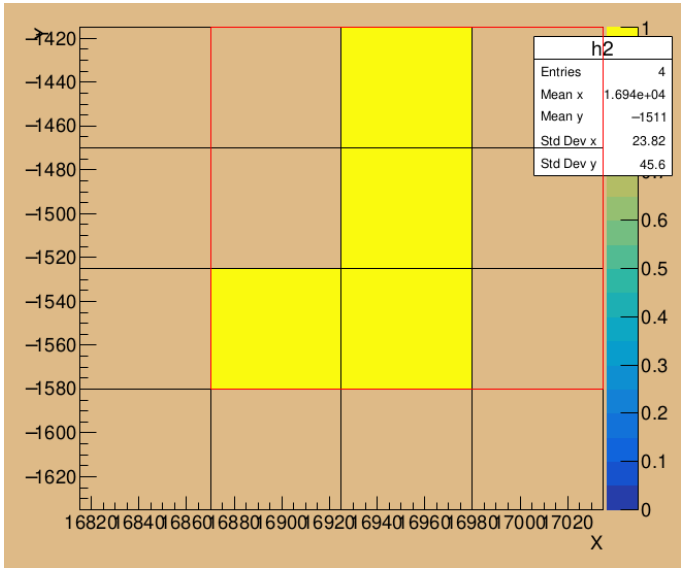
# Some examples: region 0



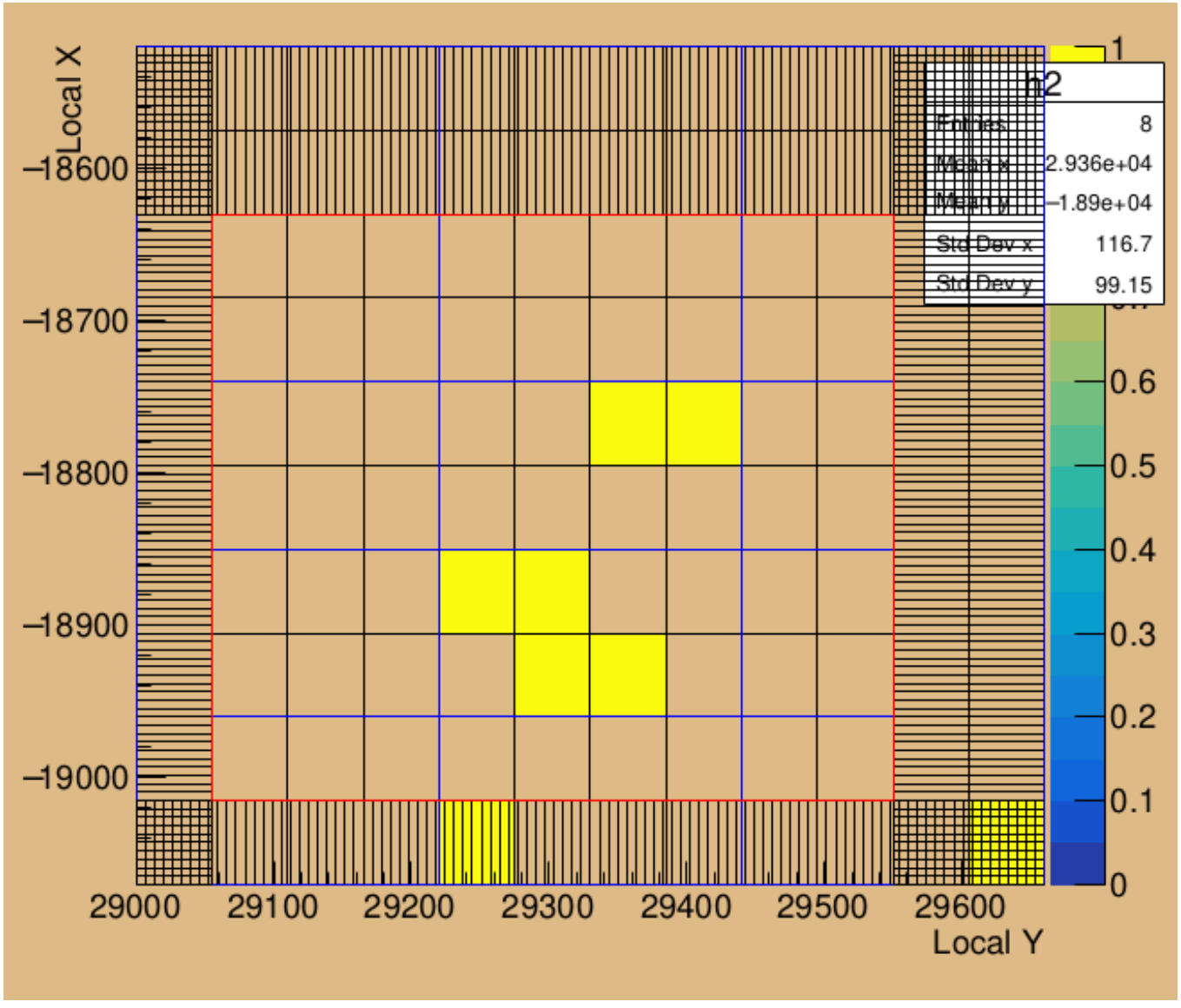
# FPGA-Recognized clusters



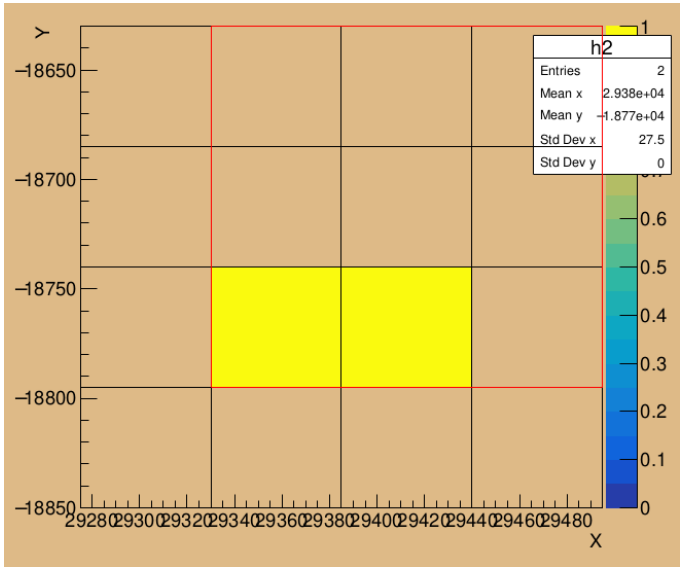



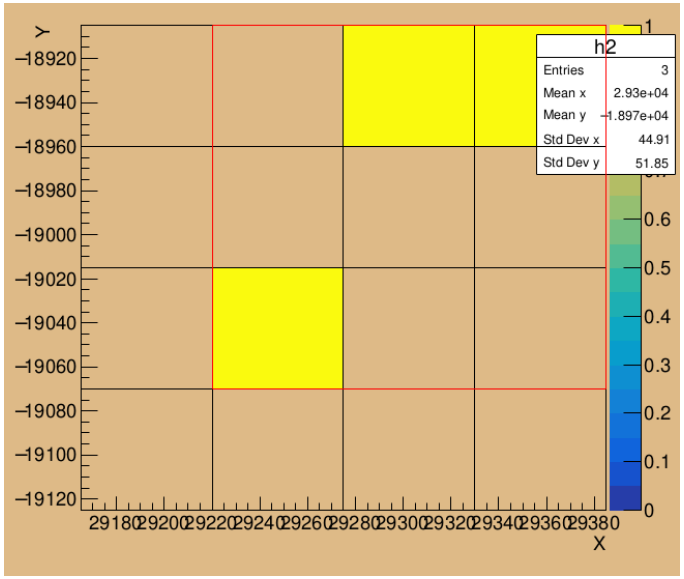



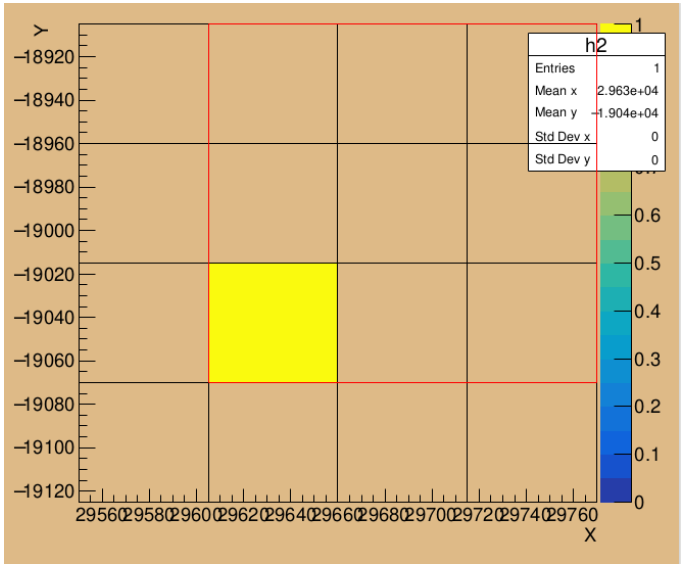
# Some examples: region 3



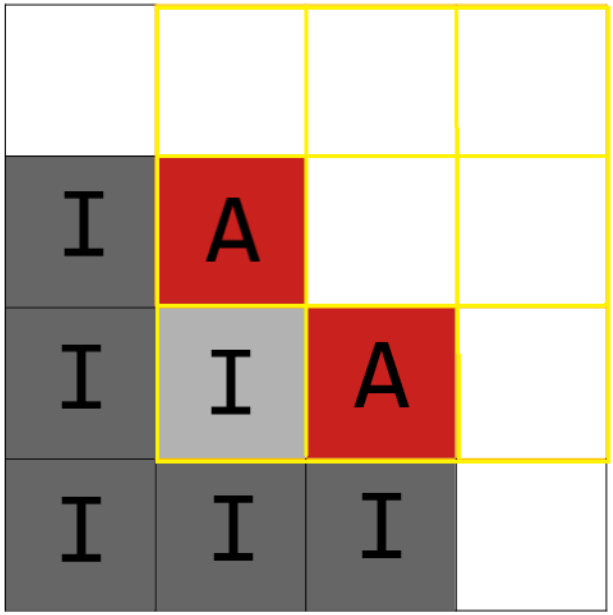
# FPGA-Recognized clusters

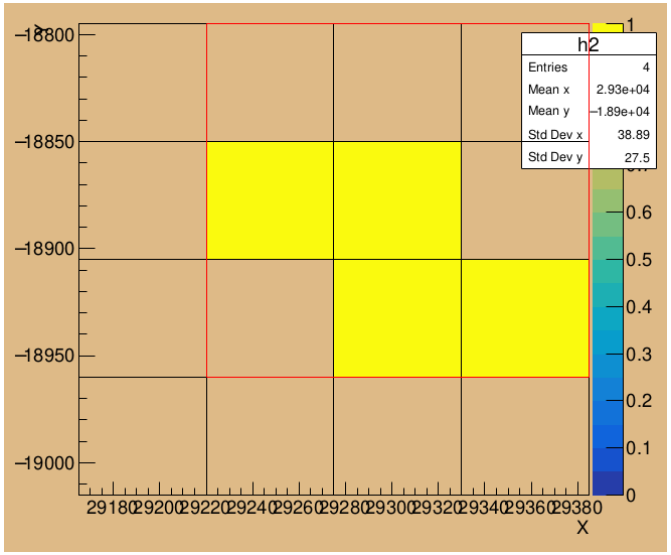


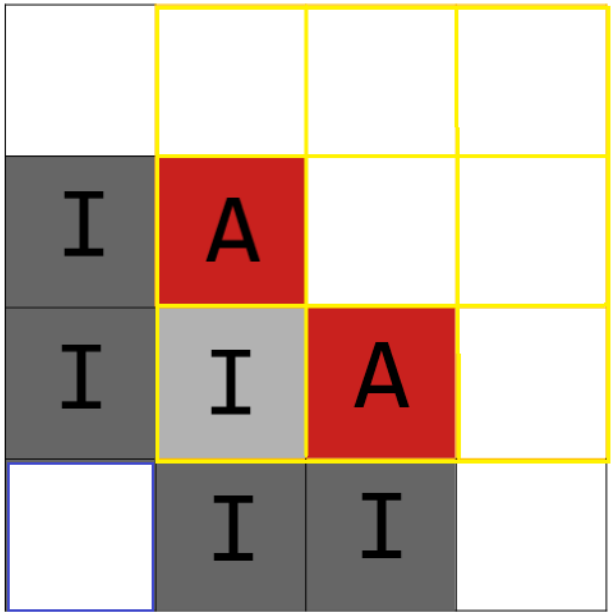
# Introduction of a new pattern

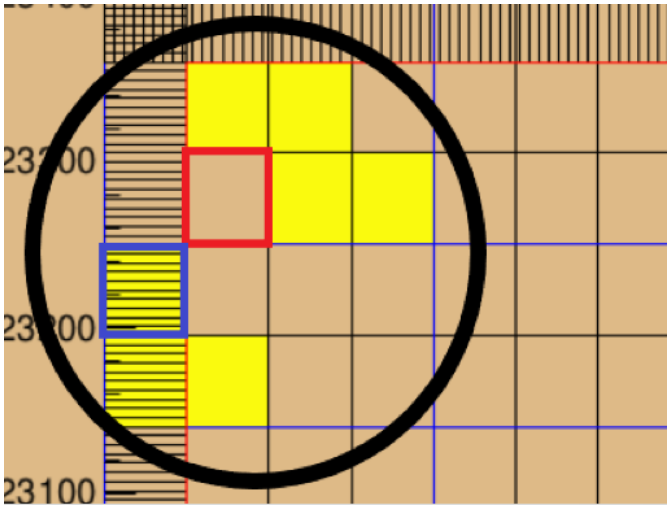



Introduction of the new pattern to take in consideration the off-diagonal tracks

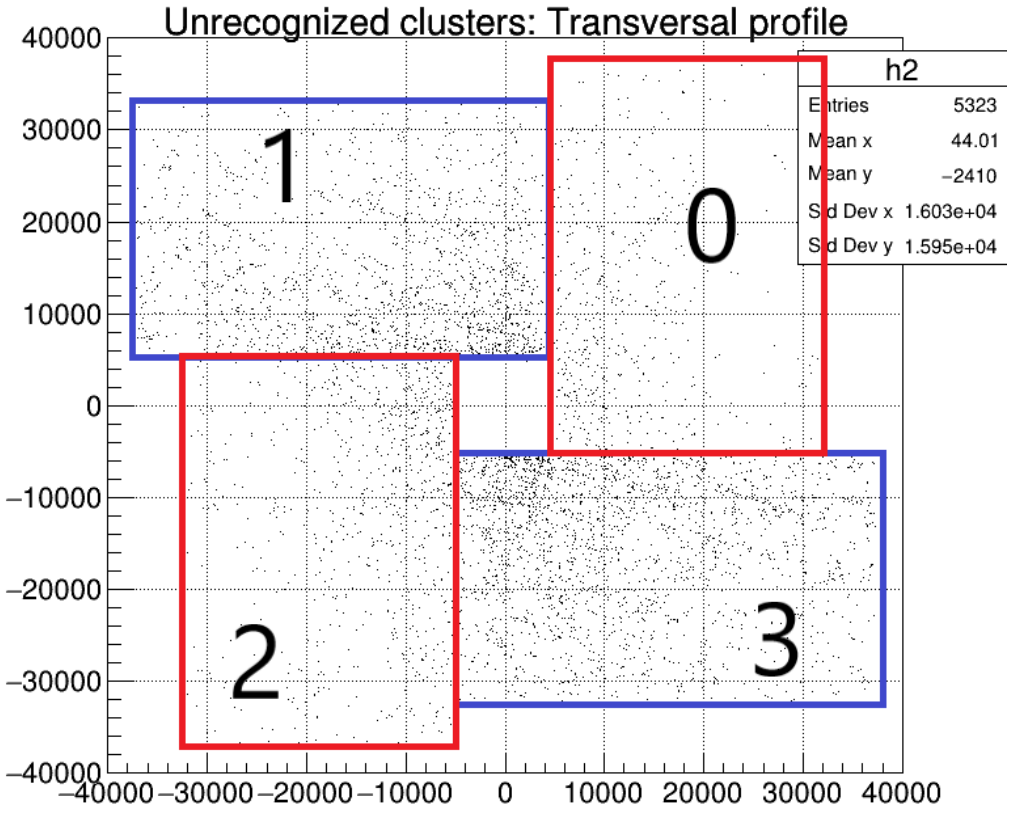
# Modification of the new pattern




Problem in the recognition of some close clusters: modification of previous pattern.

# Results




| | CPU-Reconstructed | Unreconstructed | Inefficiency[%] |
|---|---|---|---|
| HORIZONTAL REGIONS | 737541 | 4006 | 1.03 |
| VERTICAL REGIONS | 737541 | 1317 | 0.34 |

# Maybe a MS problem??

Build again the RB switching off the boolean for MS.
The number of active pixel are reported as function of the region.

| REGION # | **With MS** | **Without MS** |
|---|---|---|
| 0 | 203834 (+0.5%) | 202246 (+0.35%) |
| 1 | 202506 (-0.15%) | 202423 (+0.43%) |
| 2 | 203564 (+0.37%) | 201054 (-0.24%) |
| 3 | 201314 (-0.73%) | 200431 (-0.54%) |
| TOTAL | 811218 | 806154 |

Also trials with different events and considering only prompt particles, but same problem! Any ideas for any test to perform?