A Fast General-Purpose Clustering Algorithm Based on FPGAs for High-Throughput Data Processing

A. Annovi, M. Beretta, P. Laurelli,

G. Maccarrone, A. Sansoni

INFN - Frascati

FRONTIER DETECTORS FOR FRONTIER PHYSICS

11th Pisa meeting on advanced detectors

24-30 May 2009

Why 2D pixel clustering?

- Several applications for Pixel detectors in general
  - Particle tracking (including trigger)
  - Medical imaging
  - Astrophysics image acquisition etc.

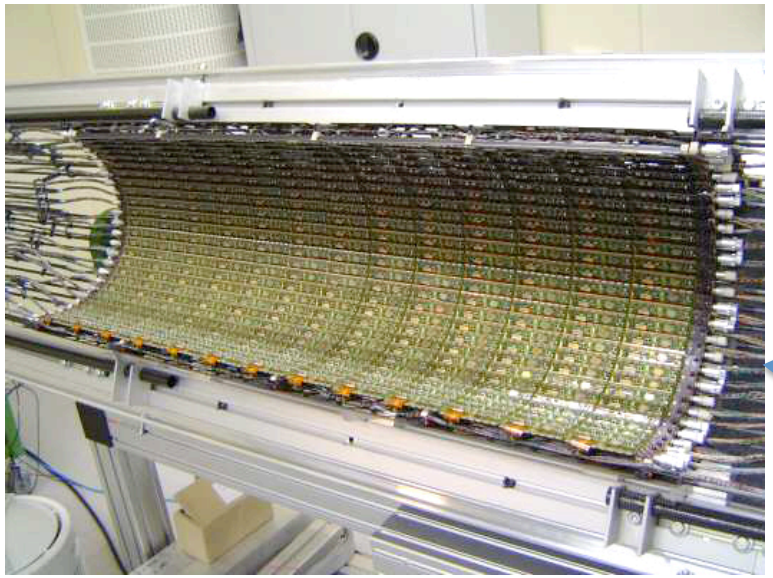
SDSS

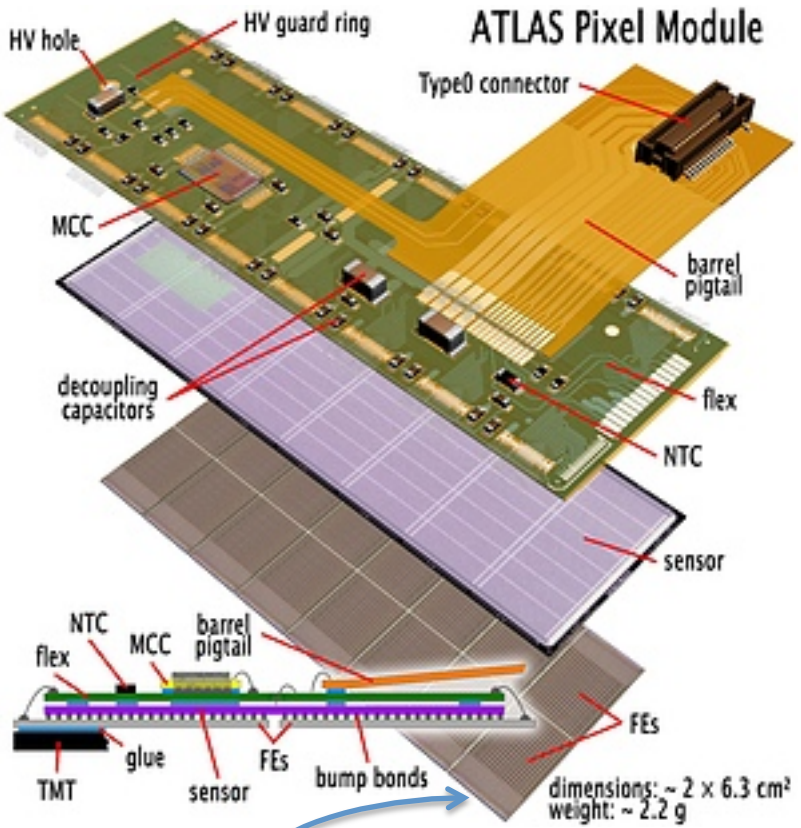
- All can benefits from a high-throughput clustering

- The clustering algorithm has several functions
  - Improves resolution (e.g. spatial or other parameters)
  - Reduce the amount of data (N hits → 1 cluster)
  - Identifies objects (e.g. for medical imaging)
  - Perform advanced cluster shape analysis
    [A. Retico et al., Comput Biol Med. 2008 Apr; 38(4):525-34]

Imagine a self clustering detector

ATLAS pixel layer 2





Thousands of fibers sending raw-data….
Imagine a device that clusters data on the fly:

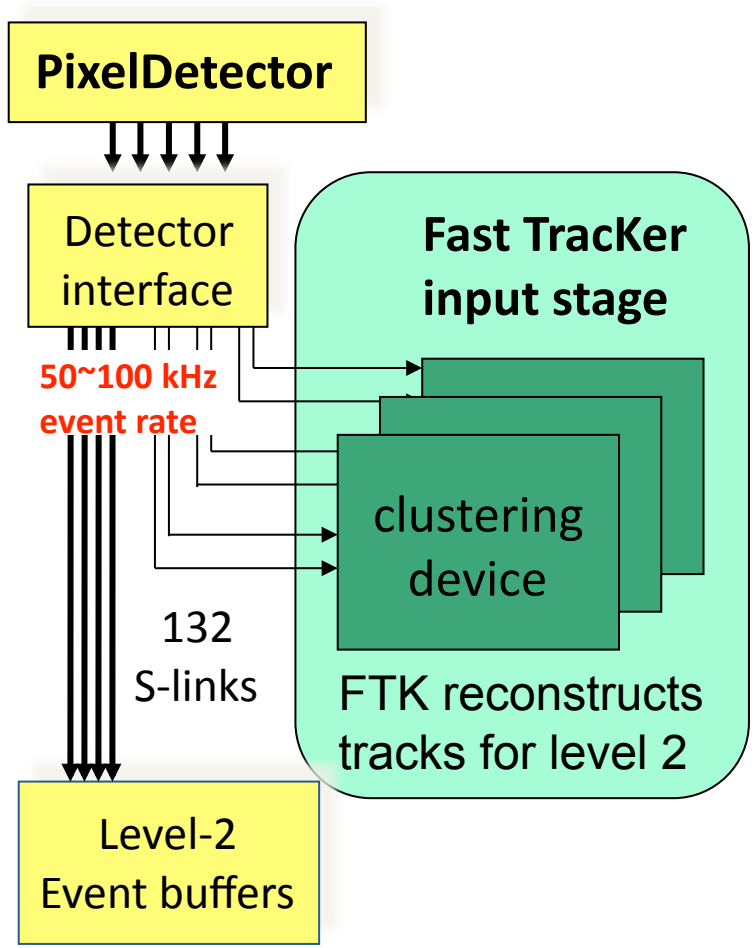
1. Connected to the remote fiber end (off detector)
2. Or directly on the pixel front-end electronics

[2008 JINST 3 P07007]

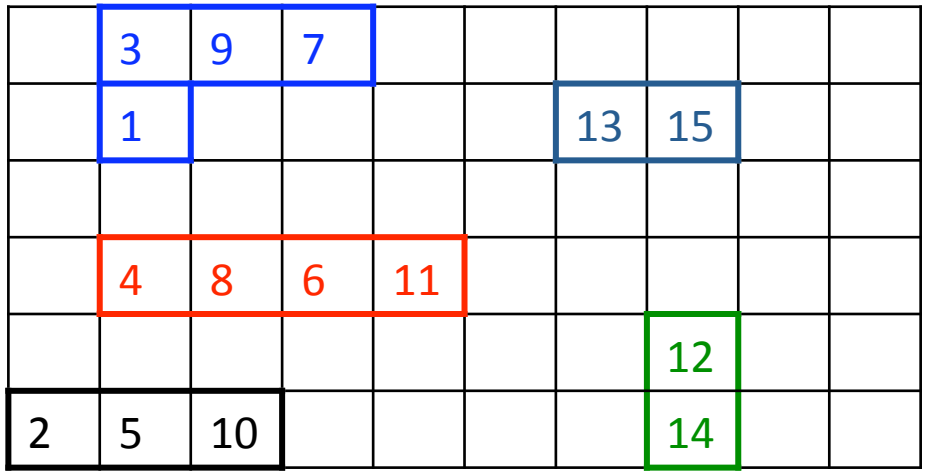
Front-end with partial clustering ATLAS Insertable B-Layer (see H. Pernegger Friday)

A real application: clustering for the ATLAS Fast TracKer

- Pixel clustering device for the ATLAS FastTracKer processor
  - 1st application & design motivation
    http://twiki.cern.ch/twiki/bin/view/Atlas/FastTracker
- Main challenge: input rate 160Gibts
  - 132 S-link fibers from all pixel RODs
    - Running at 1.2 Gbits (total 160Gbits)
    - 32bit words at 40MHz, 1 hit/word
  - Use hits at 40MHz as benchmark
- Focus on clustering quality for level-2
- Illustrate a general clustering strategy



The problem

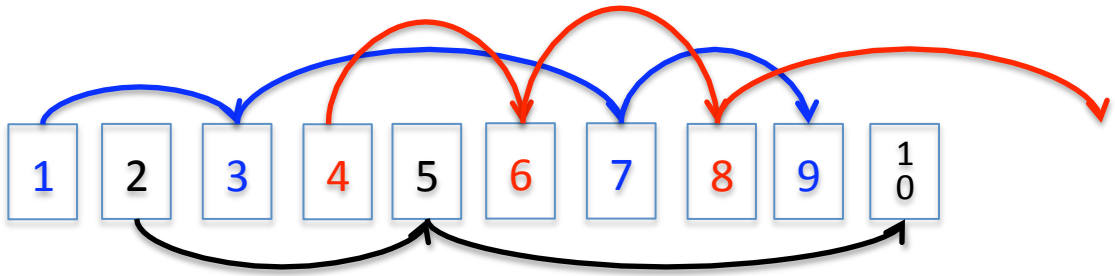


- Clustering is a 2D problem

1. Associate hits from same cluster
   - Loop over hit list
   - Time increases with occoupancy & instantneous luminosty
   - Non linear execution time
2. Calculate cluster properties
   - e.g. center, size, shape …

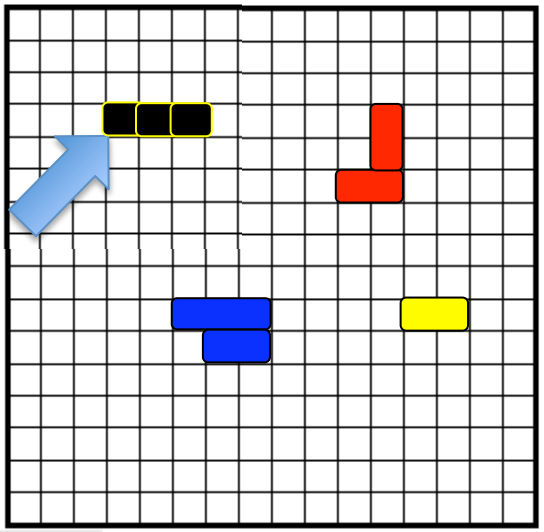
- Goal: execution time linear with number of hits
  - Not a limiting factor even at highest inst. Luminosity

Loop over list of hits



The algorithm working principle

FPGA replica of pixel matrix



Eta direction -->

1st phase:

The pixel module is a 328x144 matrix.

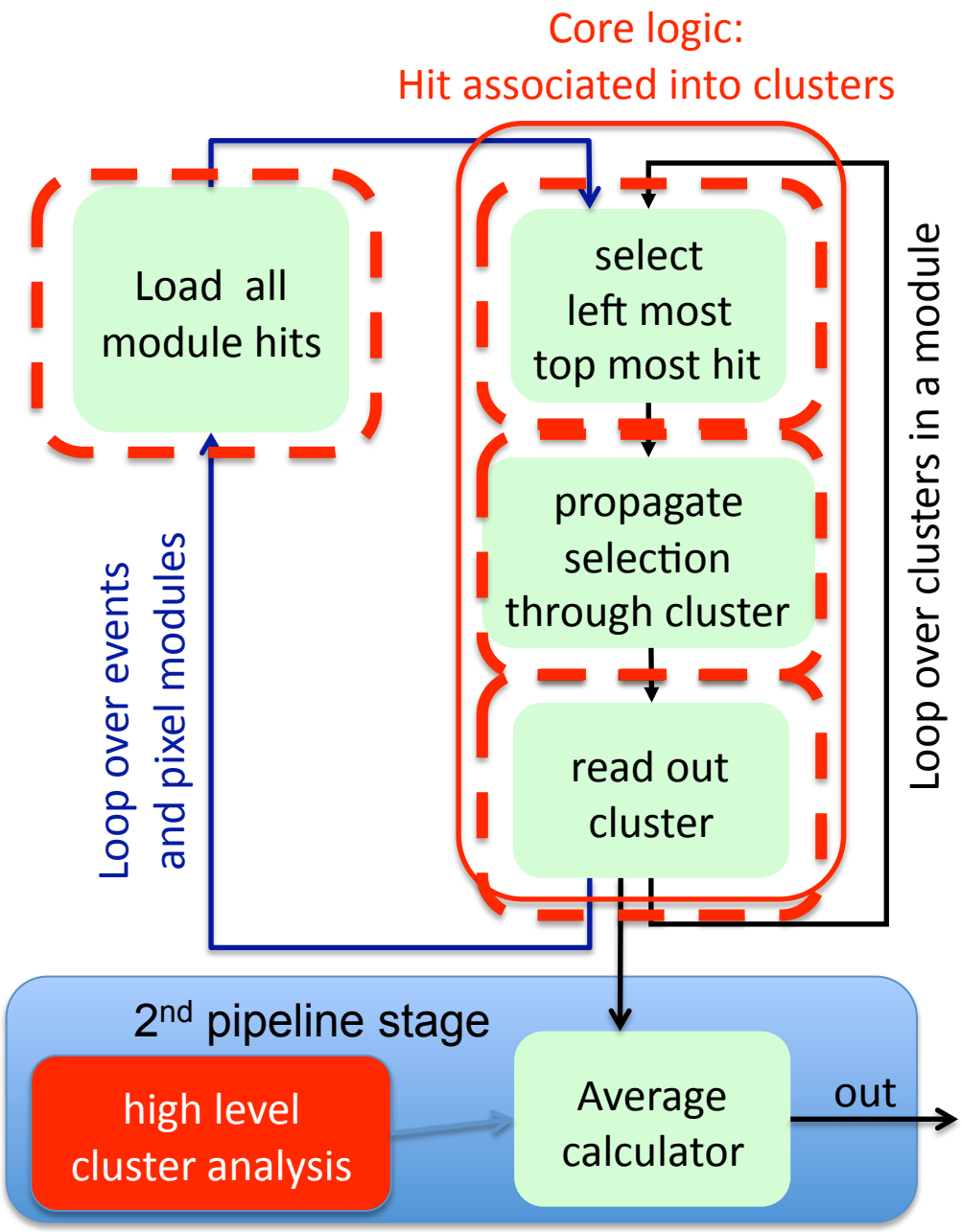
Replicate it in a hardware matrix.

The matrix identifies hits in the same cluster (local connections).

2nd phase:

Hits in cluster are analyzed (averaged).

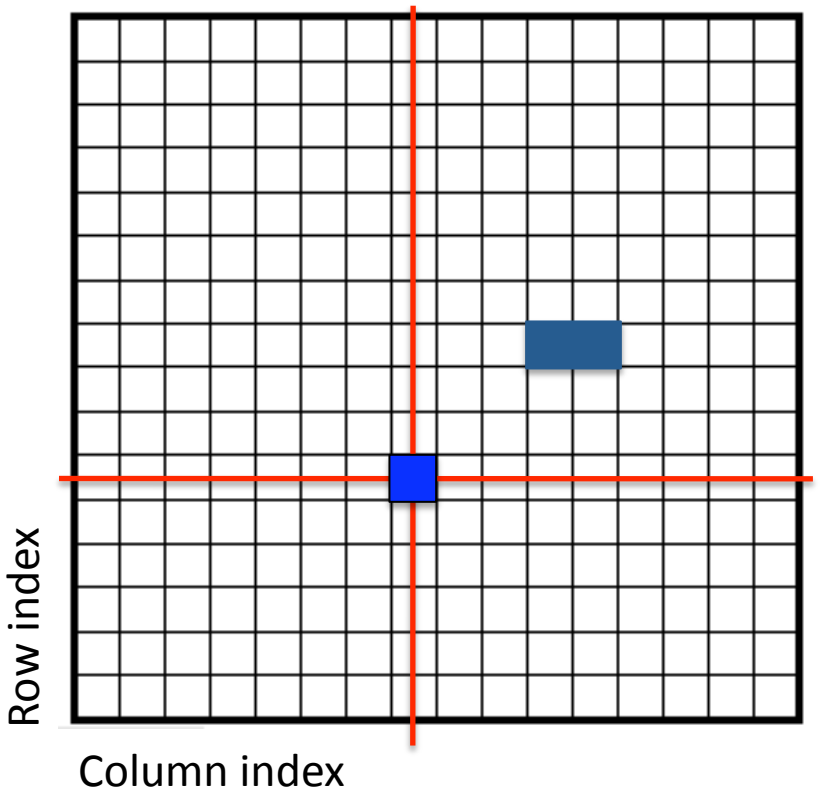
Flexibility to choose algorithm!



Core logic

Logic functions

1. Load hits
2. Select left-top-most hit
3. Propagate “selected”
4. Readout cluster



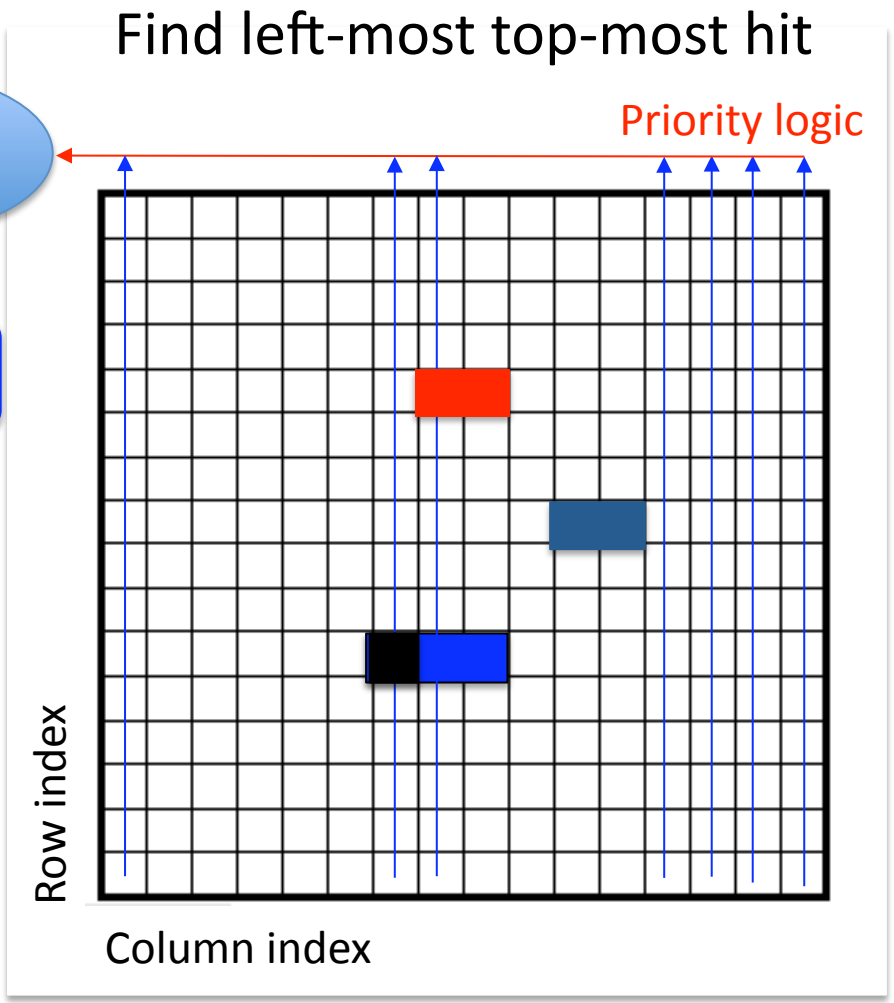
Load hits regardless of readout order.

Any readout order is allowed.

Core logic

Logic functions

1. Load hits
2. Select left-top-most hit
3. Propagate “selected”
4. Readout cluster

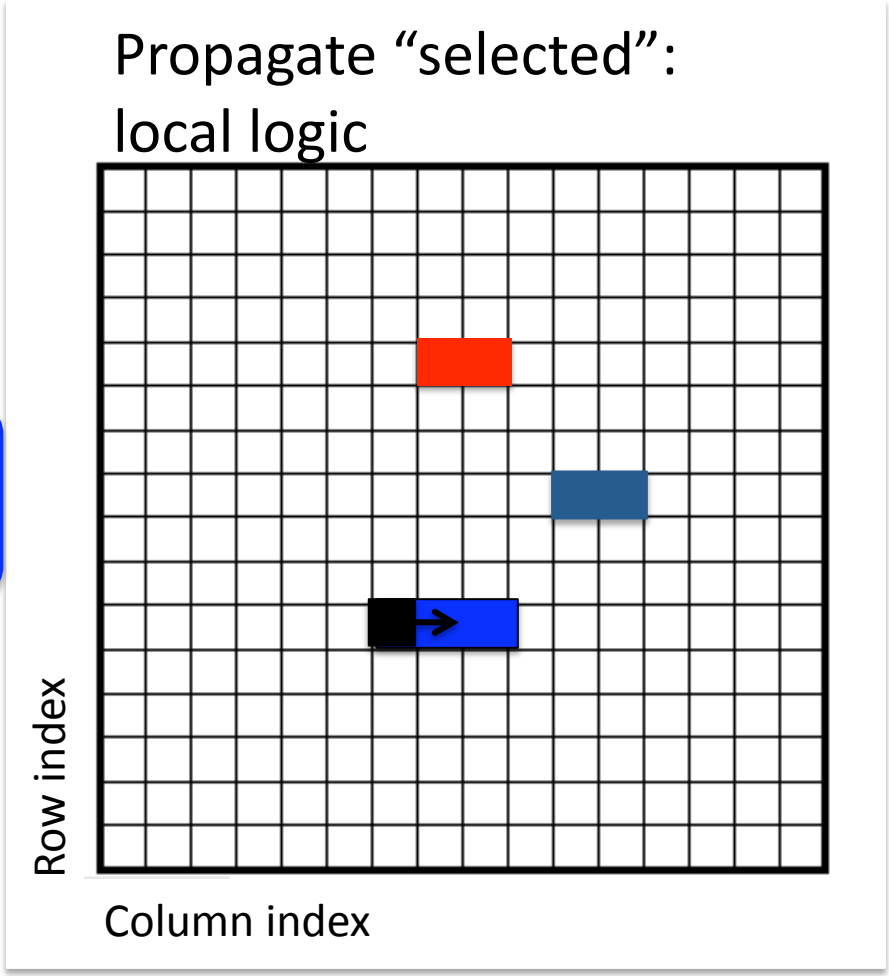


MANY levels of logic (328+144)
Will determine algorithm (clock) speed.
(See slide # 16)

Core logic

Logic functions

1. Load hits
2. Select left-top-most hit
3. Propagate “selected”
   - Black pixel
4. Readout cluster

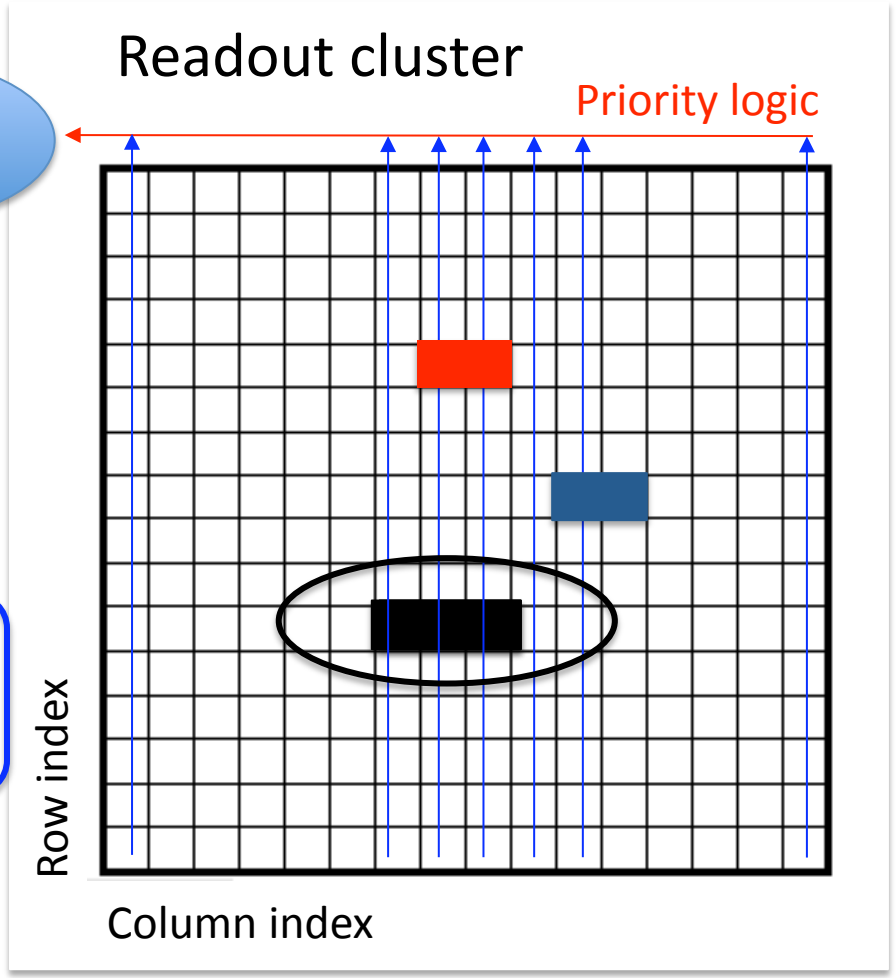


Core logic

Logic functions

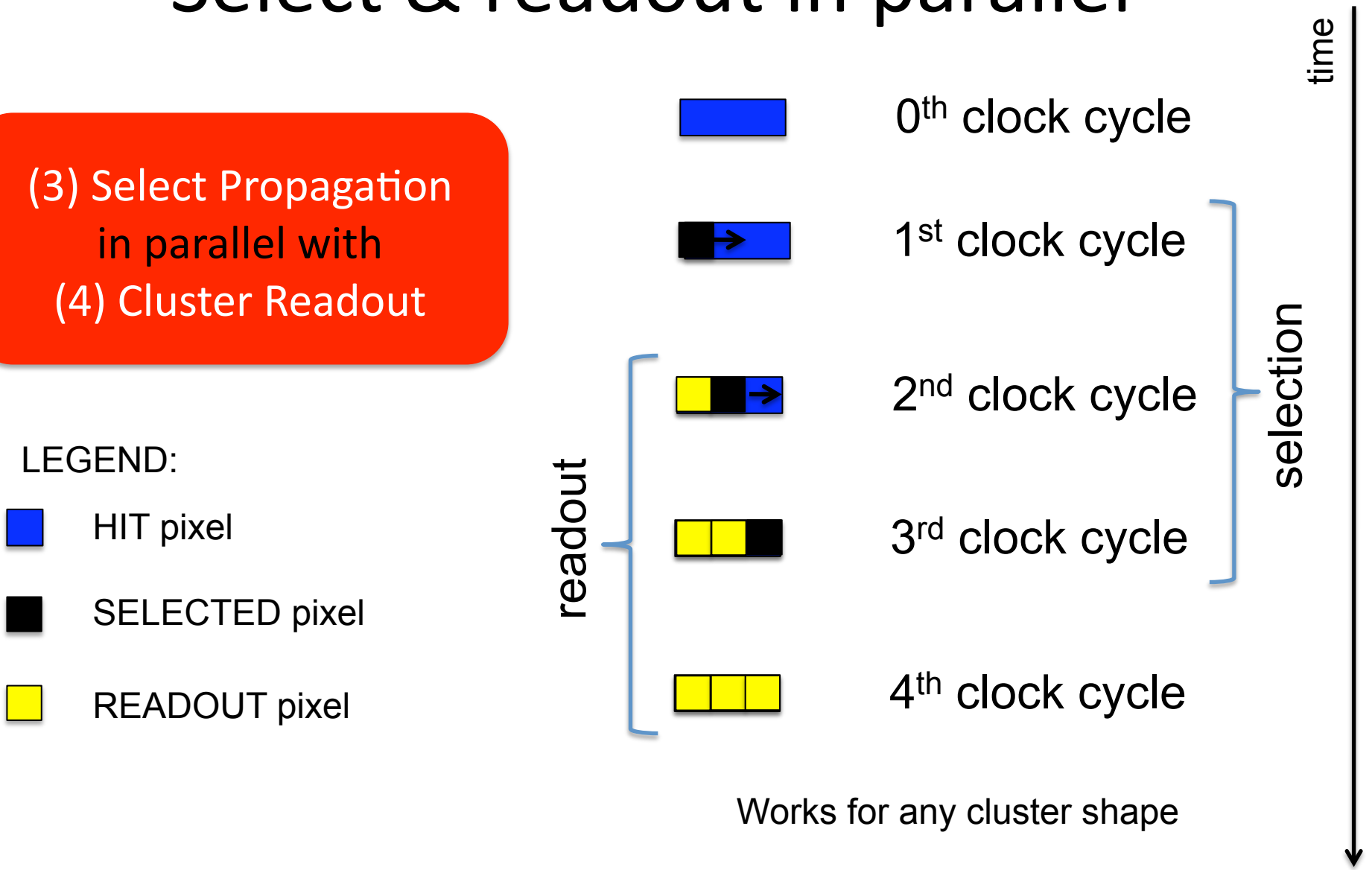
1. Load hits
2. Select left-top-most hit
3. Propagate “selected”
   - Black pixel
4. Readout cluster
   - Black pixels

(3) Select Propagation
in parallel with
(4) Cluster Readout

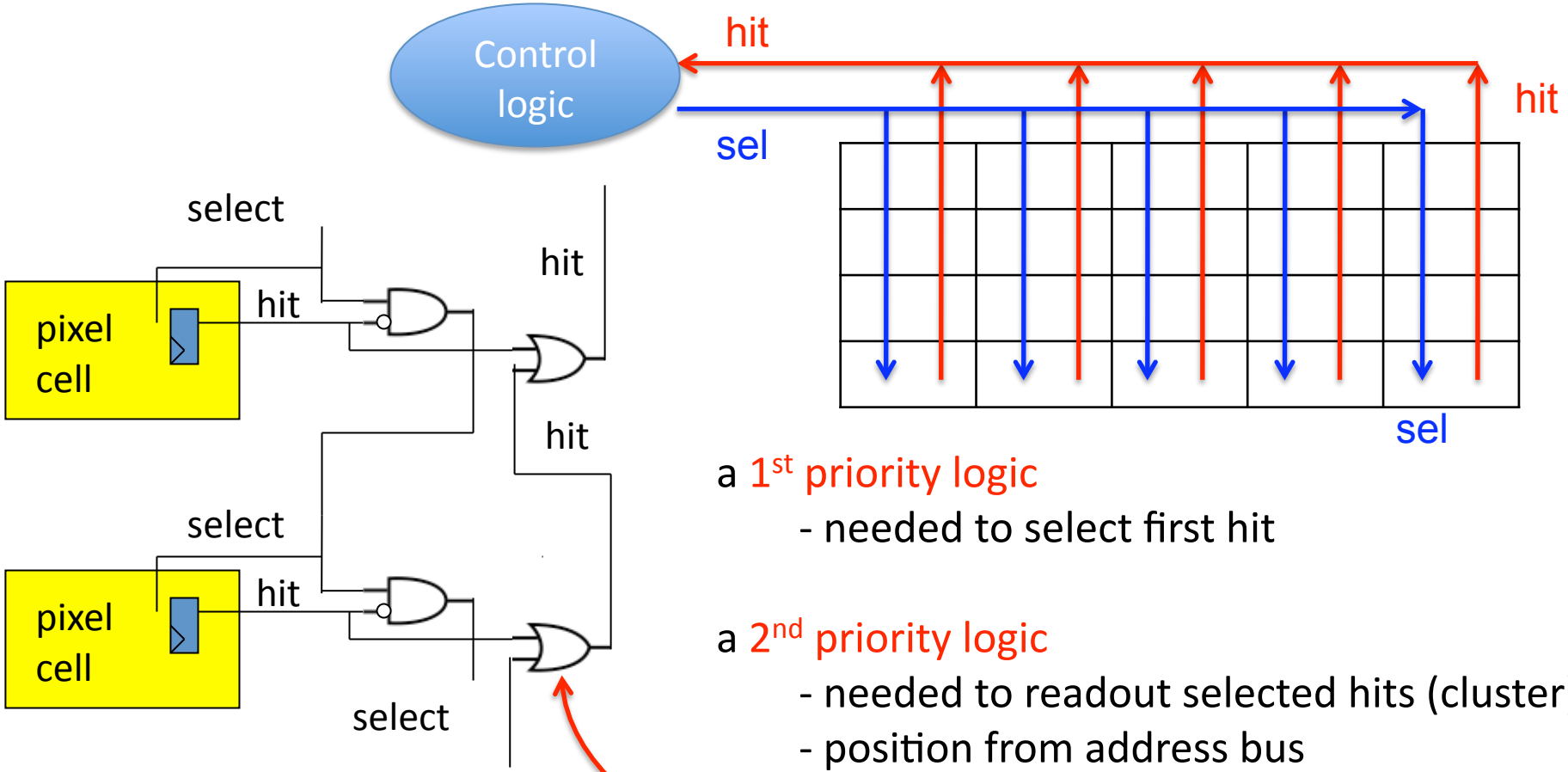


Select & readout in parallel

(3) Select Propagation in parallel with (4) Cluster Readout



Two priority logic chains



- X 328 pixels in a column
- and 144 columns

a 1st priority logic
- needed to select first hit

a 2nd priority logic
- needed to readout selected hits (cluster)
- position from address bus

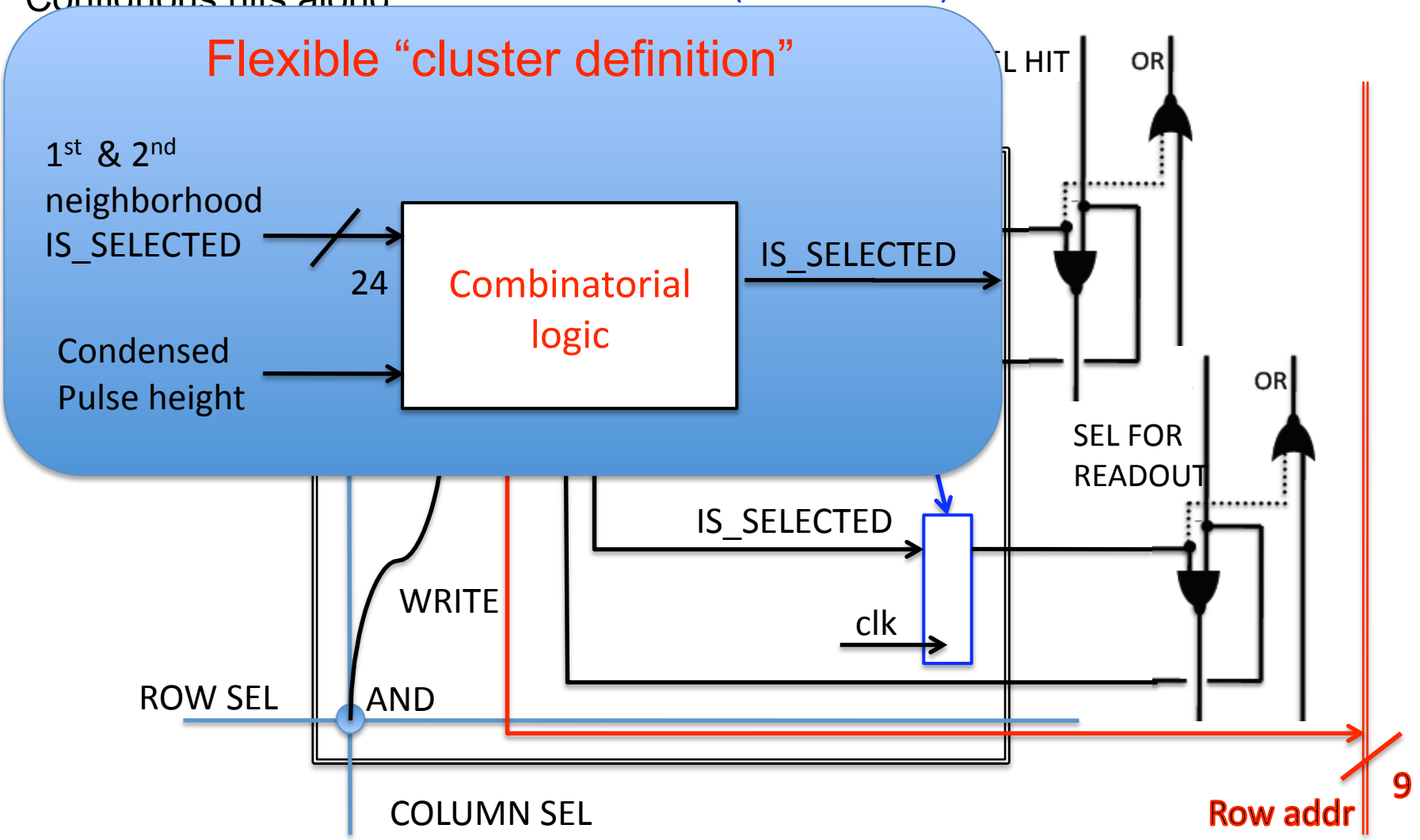
This logic selects the top most pixel.
Similar logic to select the left most column with a hit.

The elementary cell

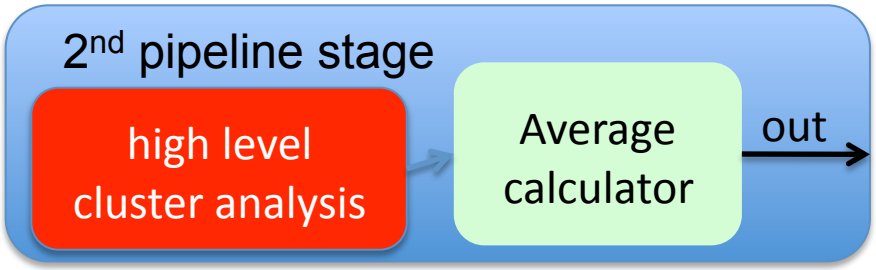
Cluster definition:

Contiguous hits along

3 STATES (2 FLIP-FLOPS):



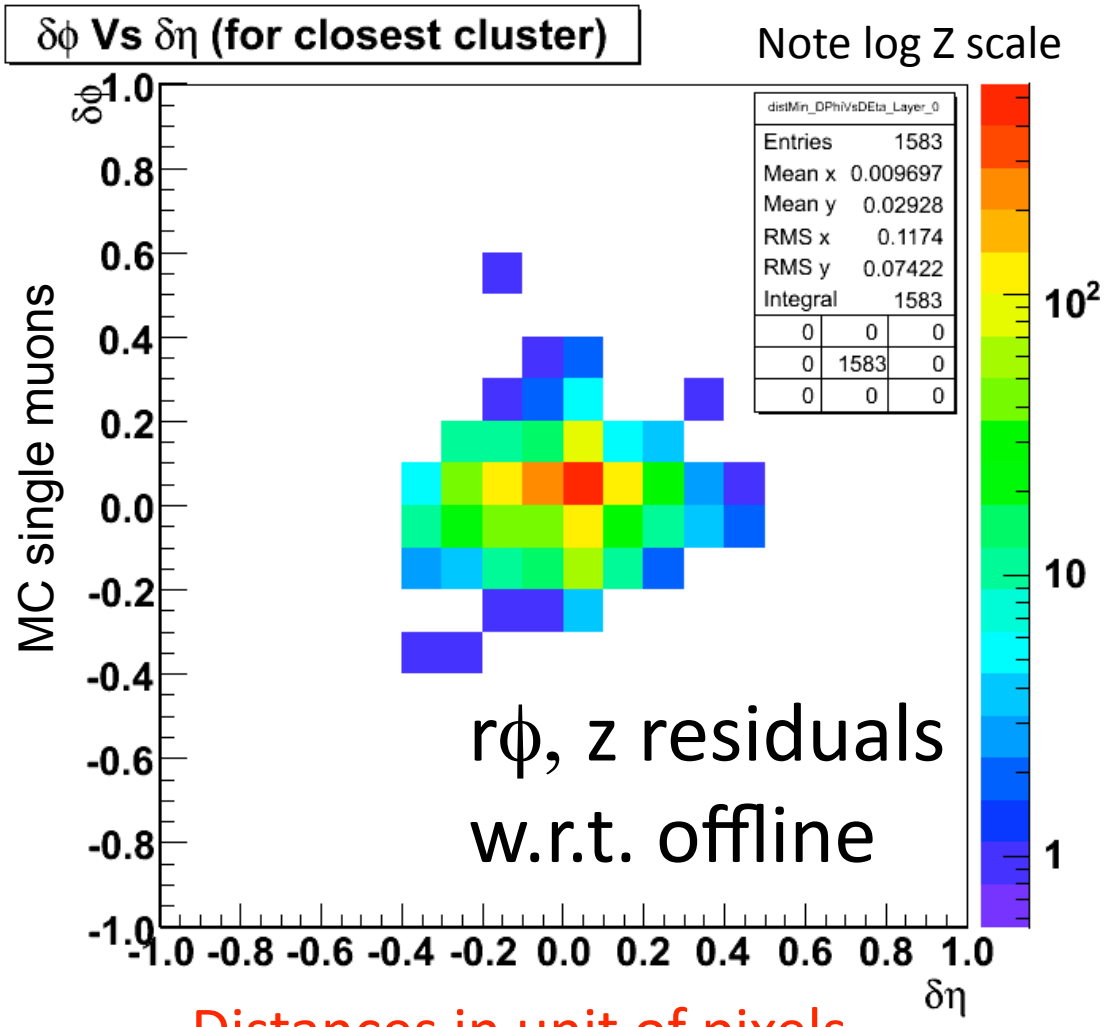
Clustering resolution w.r.t. offline



Including collected charge average:
RMS rϕ ~ 0.002 pixels (~0.1μm)
RMS z ~ 0.05 pixels (~20μm)
Offline accounts for track angle.

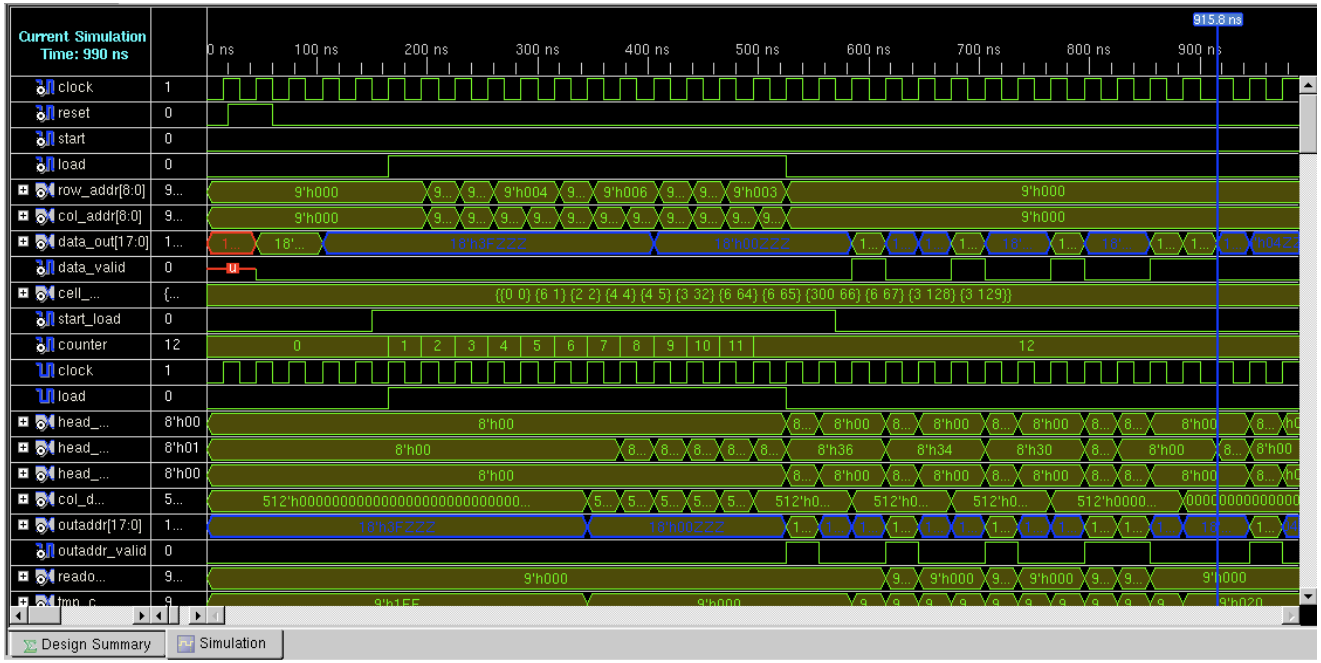
Without collected charge (digital):
Clusters are found close to offline
RMS ~0.1 pixels
RMS rϕ ~ 0.07 pixels (~4μm)
RMS z ~ 0.12 pixels (~50μm)

Pixels are 50x400μm

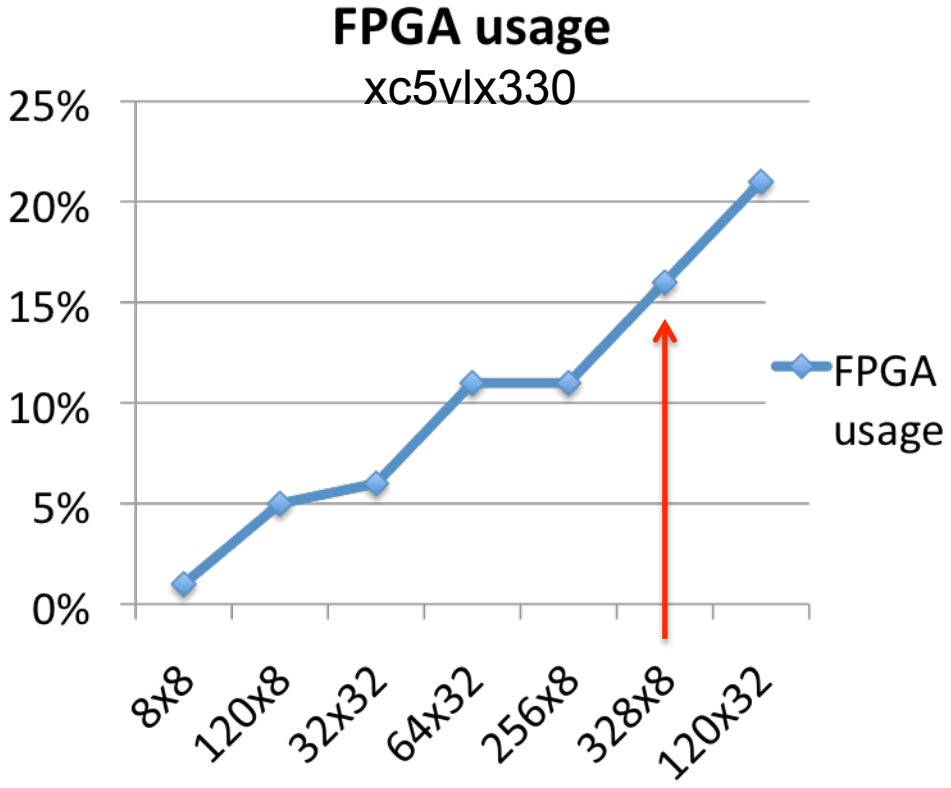


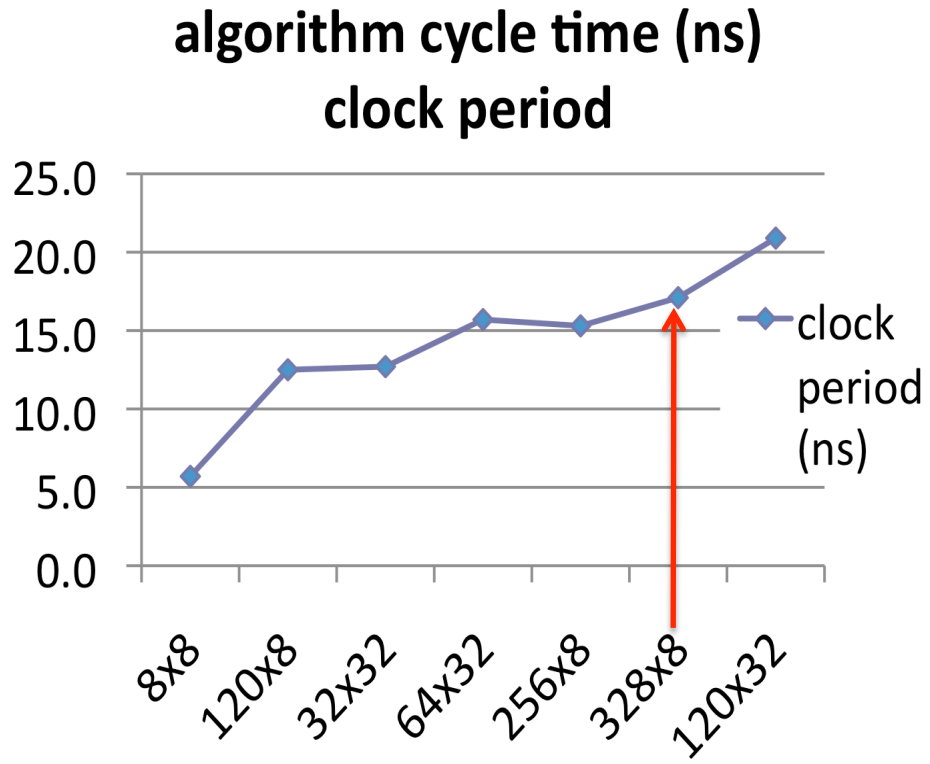
Implemented & simulated

- Using xilinx virtex5 (xc5vlx330)
- Timing (cycles): 2/hit + 2/cluster
  - could be reduced to: 2/hit



Resources and clock speed



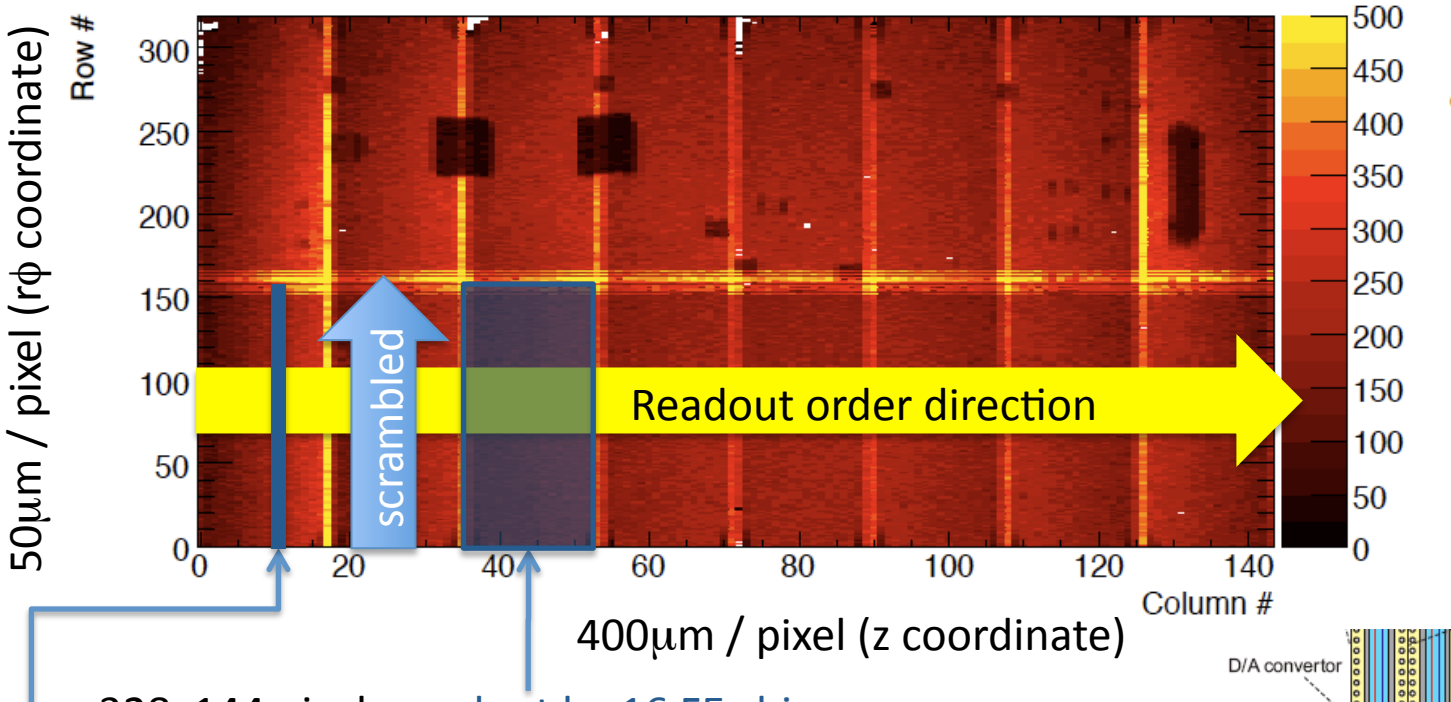


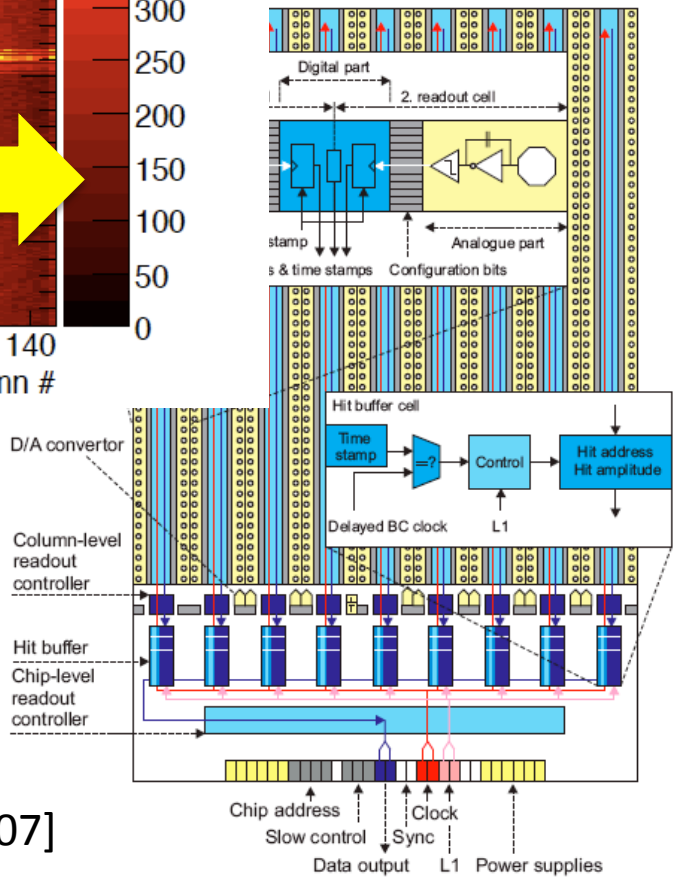
FPGA usage and clock period increase for large matrixes.

For a 328x144 matrix, area usage ~250%. Now what?

Take advantge of readout order (depend on actual detector).

ATLAS Pixel module





328x144 pixels readout by 16 FE chips
Partially sorted readout.
Hits within one double-column (half-legnth) are scrambled. Double-columns are readout in order.
FE chips are readout in order

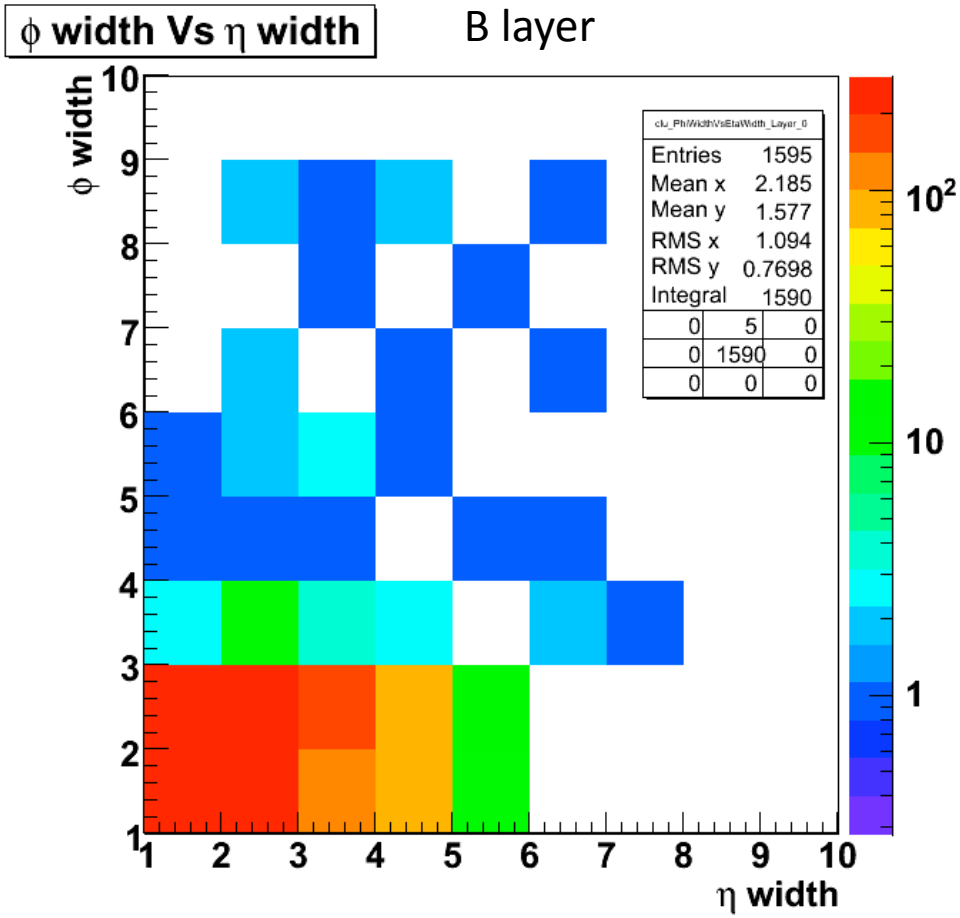
All details in [2008 JINST 3 P07007]

Cluster shapes

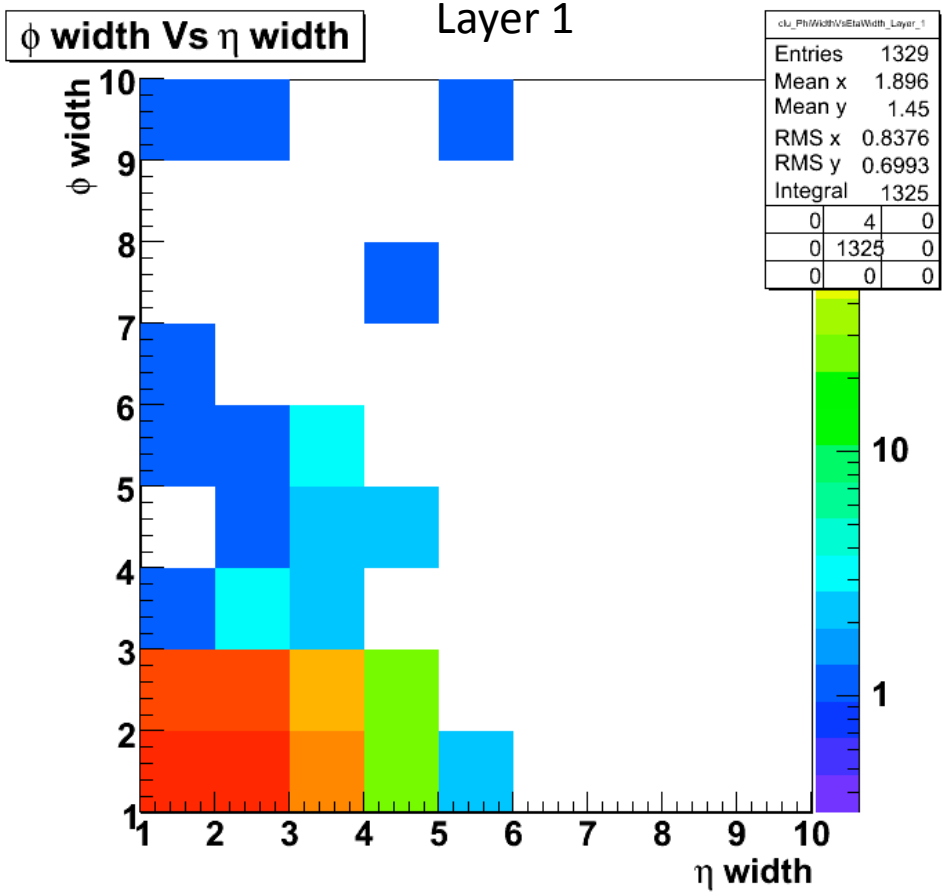
Most clusters are smaller than 5x3 pixels

Safely fit in a matrix 8 pixel wide along η

B layer

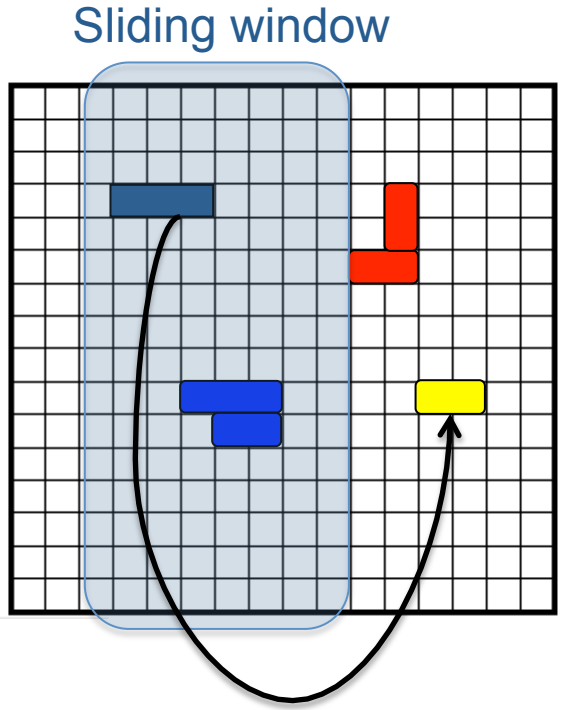


Layer 1



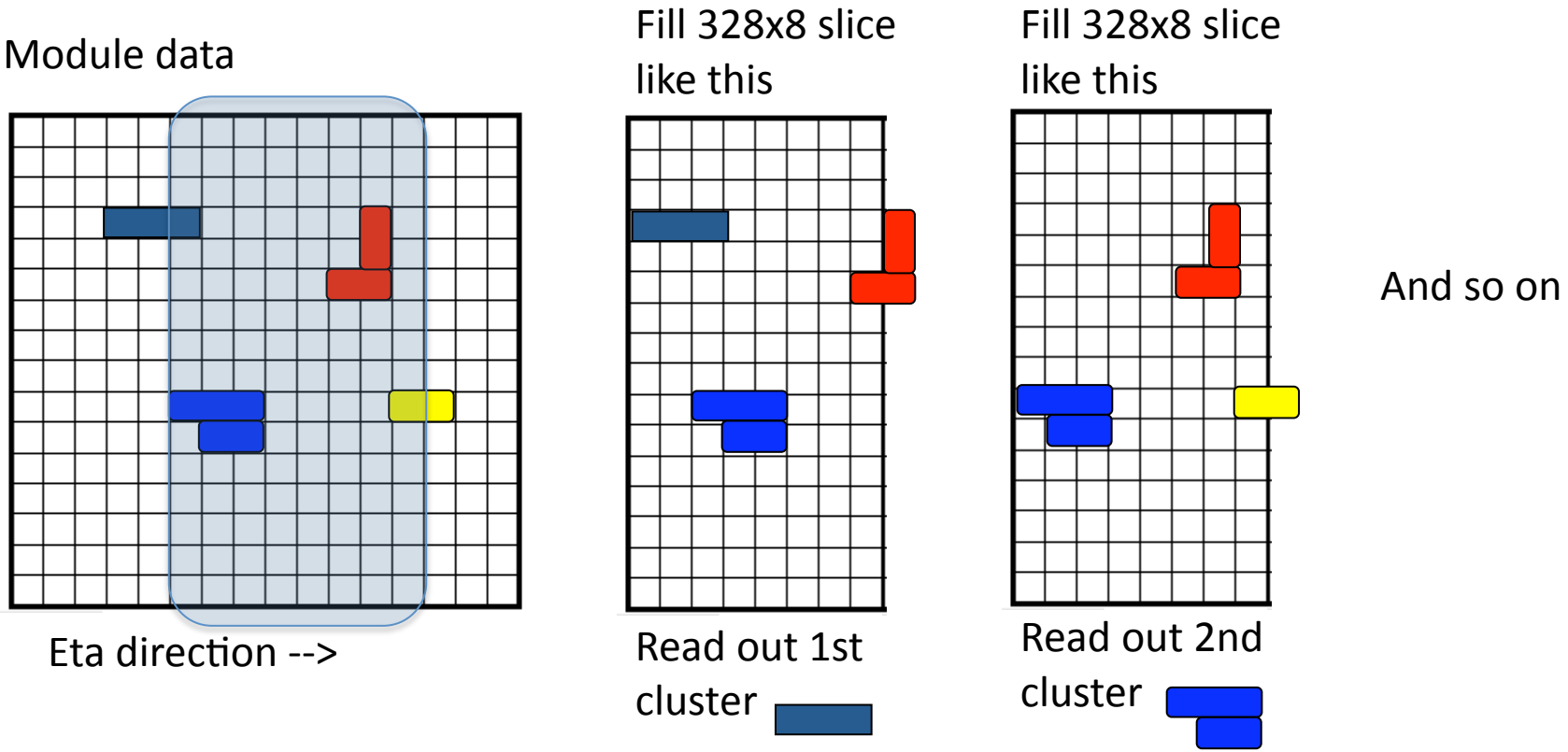
How to save logic resources?

- Take advantage of partial readout ordering
- Use a sliding window to process the complete module
- Use a 328x8 matrix
  - Full rϕ length (can be squeezed)
  - Larger than maximum cluster size (5)
  - Clock freq. 58MHz
    - 3 cycles/hit → ~ 20MHz hit proc. rate
  - Area usage 32% (xc5vlx155)
  - Use 2 matrixes 64% of xc5vlx155 safely process one 40MHz Slink



Far away clusters are unrelated

Clustering by 328x8 slices?

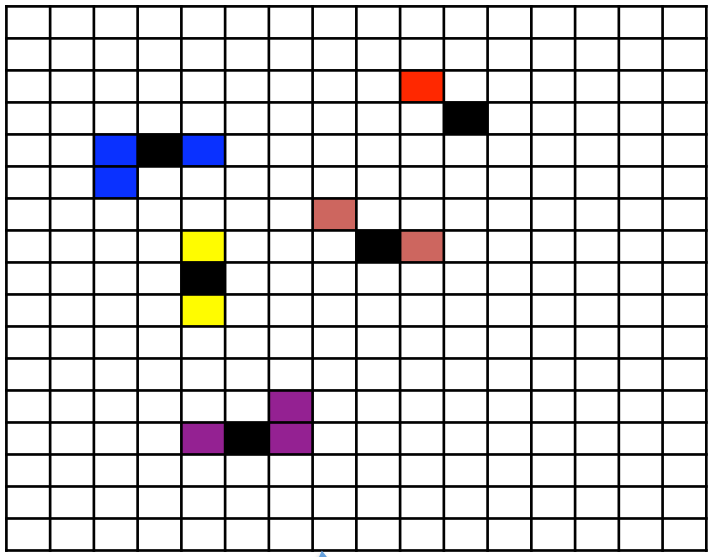


Shift of hits comes for free (no extra time)! Just use the slice as a circular buffer in the eta direction. Then hits are shifted by redefining the first column.

SLIDING WINDOW: with one xc5vlx155 process one S-Link
Implement 2 processing matrixes.
Process hits at 40MHz rate.

Another clustering algorithm

Logic matrix

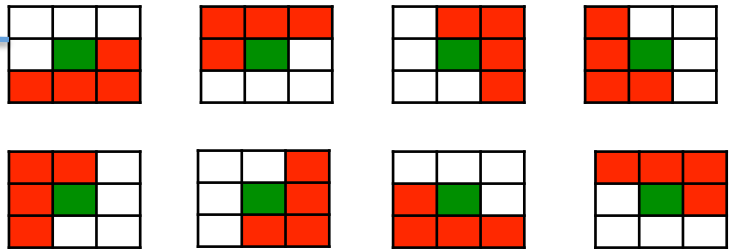


LEGEND:

Center of cluster

Apply mask to all pixels in parallel

List of rules/masks can be changed



Make cluster and find center @ once

No need for long priority chains.

Caveats: center is approximate, rarely splits clusters.

Clustering executed in parallel:

- ideal for level-1 applications
- ideal for on detector applications

ALGORITHM RULES:

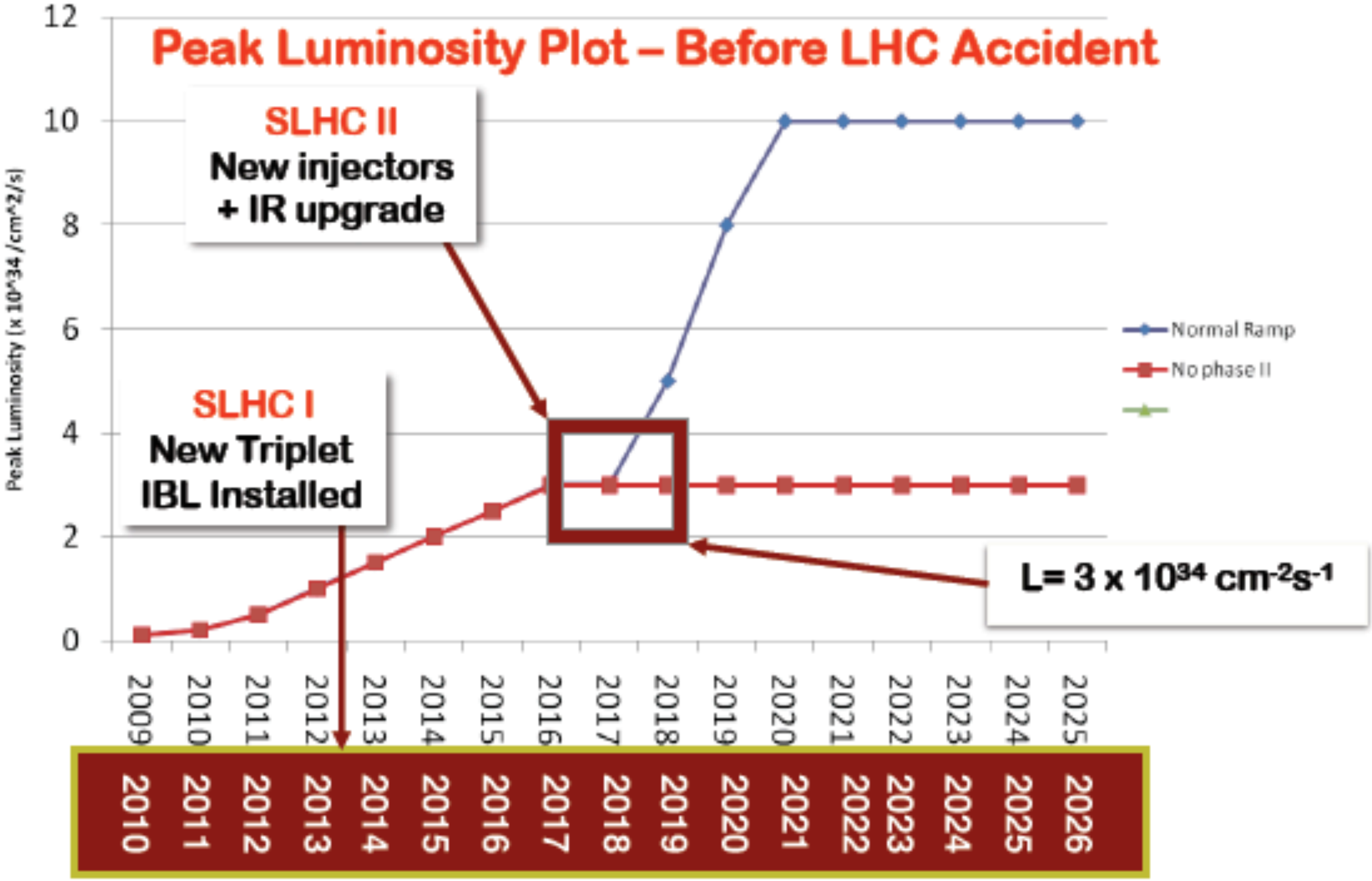
- turn off "external hits"
- for each hit delete rule is : turn off "hit" IF
  - one of "red cell is hit" AND
  - all "white cells" are off
- looping over this list of "rules/masks" until only single hits are left

Conclusions

- Developed a clustering algorithm for the Fast Tracker
  - Full resolution with linear processing time
- The algorithm is fully general
  - Could do 3D clustering
  - Re-usable for other applications
  - Can employ flexible “contiguity/cluster” definition
  - For the Fast Tracker contiguity is by side or corner
- Proposing also a fully parallel algorithm
  - Level1, On detector applications

BACKUP SLIDES

LHC: Luminosity Plot



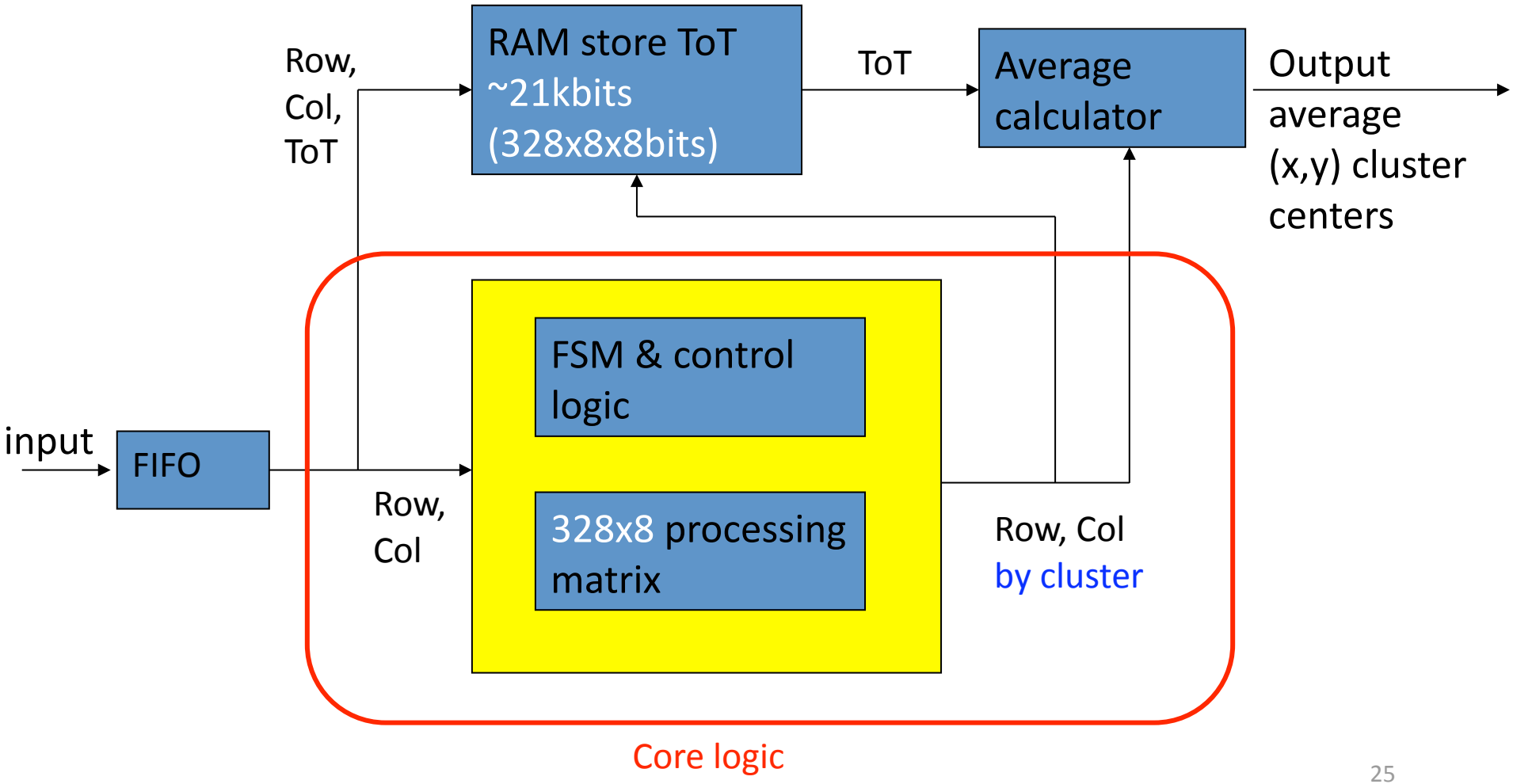
Shifted by one year
Ref. F. Zimmermann - AUW

SLHC Phase I delayed by one year 2014
Ref.: M. Nessi (AUW) from L. Evans

Luminosity Plot
Ref. R. Garoby – LHCC 1/7/2008 –
http://indico.cern.ch/conferenceDisplay.py?confId=36149

Firmware diagram

ToT = time over threshold



FPGA cost (feb 2009)

- Today avnet quotes for 1 piece
- xc5vlx330 --> 9600$ up (used for implementation)
- xc5vlx220 --> 3900$ up
- xc5vlx155 --> 2300$ up
- xc5vlx110 --> 1500$ up
- Logic proportional to last number
  - e.g. 330 --> 330000 logic cells
- In order to process 1 S-Link at 40MHz
  - Need two grids (328x8)
    - Equivalent to ~ 110000 logic cells
  - Plus surrounding logic and safety margin
- Example: choose 1 xc5vlx155 per S-Link
  - Area usage > 66% from the 2 grids
  - Need 120 FPGAs --> 276k$ at today price