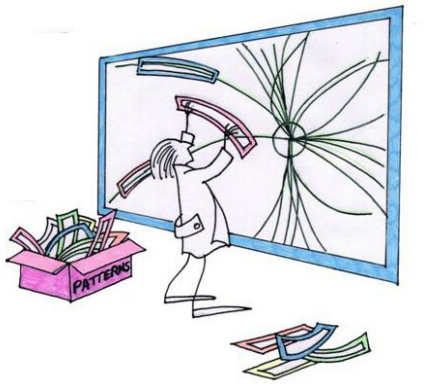
Intelligent triggering: pattern recognition with Associative Memories & other tools (FPGAs)

Fast **T**rac**K**er & **H**ardware **T**racking for the **T**rigger (HTT) in ATLAS and other examples



Kostas Kordas

Aristotle University of Thessaloniki

ARISTOTLE UNIVERSITY OF THESSALONIKI

ISOTDAQ

International School of Trigger and Data Acquisition

ISOTDAQ2023

Istanbul, Turkiye

June 21, 2023

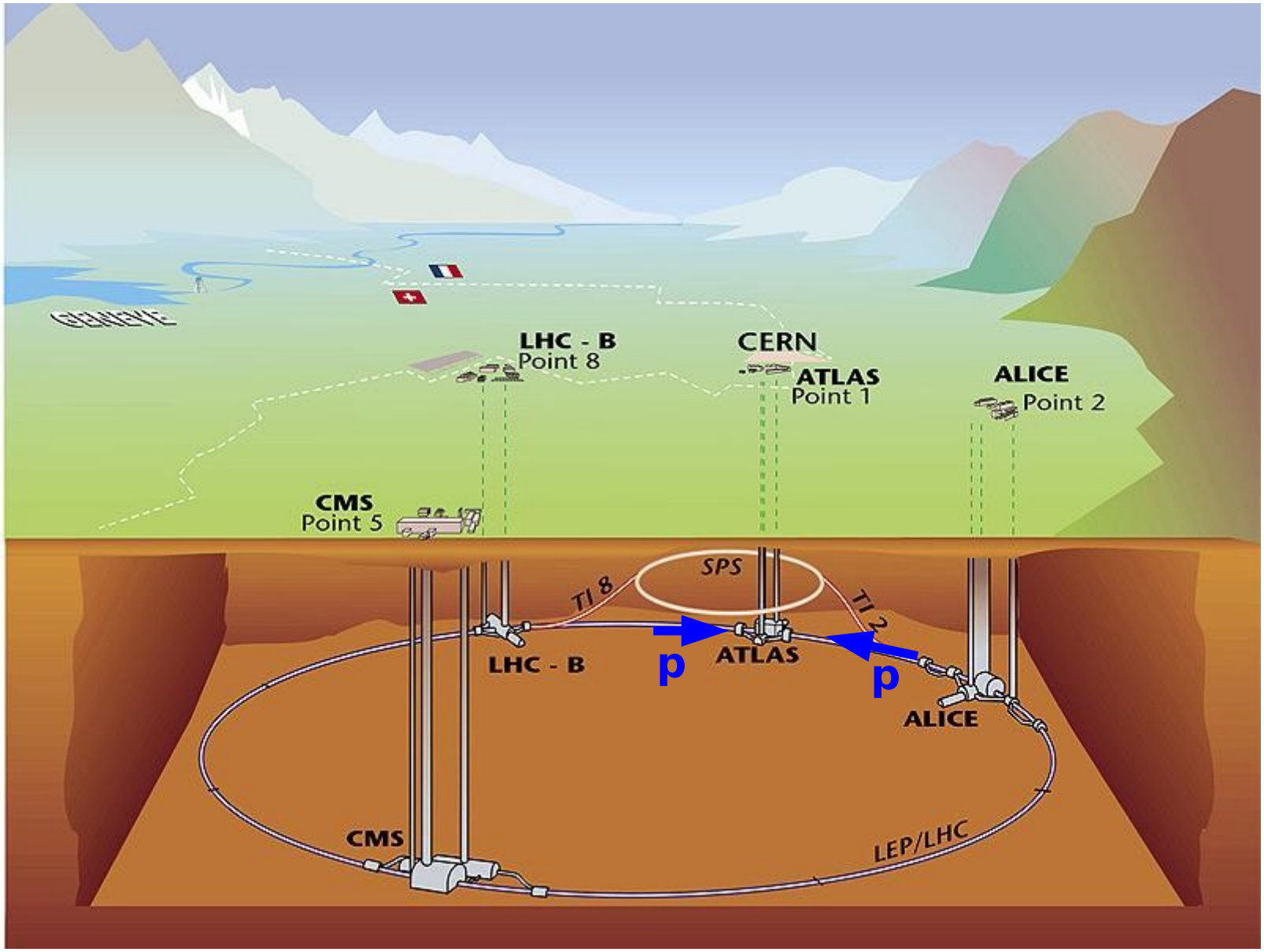
So, here we came
"To the City"
"Εις την Πόλη"
(Is tin Poli → Istanbul)

Overview

- The need for **tracking information at the Trigger** of High Energy Physics experiments and **how to do it fast**
- We'll split the problem into **"track finding"** (define fast a "road" where a track can be) and **"track fitting"** (determine the track characteristics)
- The specific examples from ATLAS (FTK and HTT) will be discussed in more detail to see all aspects of the problem:
  - **1) Track finding** with **Pattern matching** in Associative Memories , and **2) Track fitting** in FPGAs
- You'll see that: if you want to avoid or cannot afford calculating something time consuming, split the problem and use pre-calculated patterns and quantities.
- **We'll see also examples of other approaches, with both steps done in FPGAs.**
- We'll also see examples beyond High Energy Physics

A. Introduction

Experiments at the LHC - pp collisions



The energy frontier - interested in **relatively rare processes**

Probability of interaction ↔ cross section:

In order to have a reasonable number of interesting events produced, we need high luminosity colliders:

σ ~80 mb

>10 orders of magnitude!

μb

nb

pb

GHz

MHz

kHz

Hz

mHz

Rate of process ("pp → X") = L * σ

$$LHC\ Design: L = 10^{34}\ cm^{-2}\ s^{-1}$$

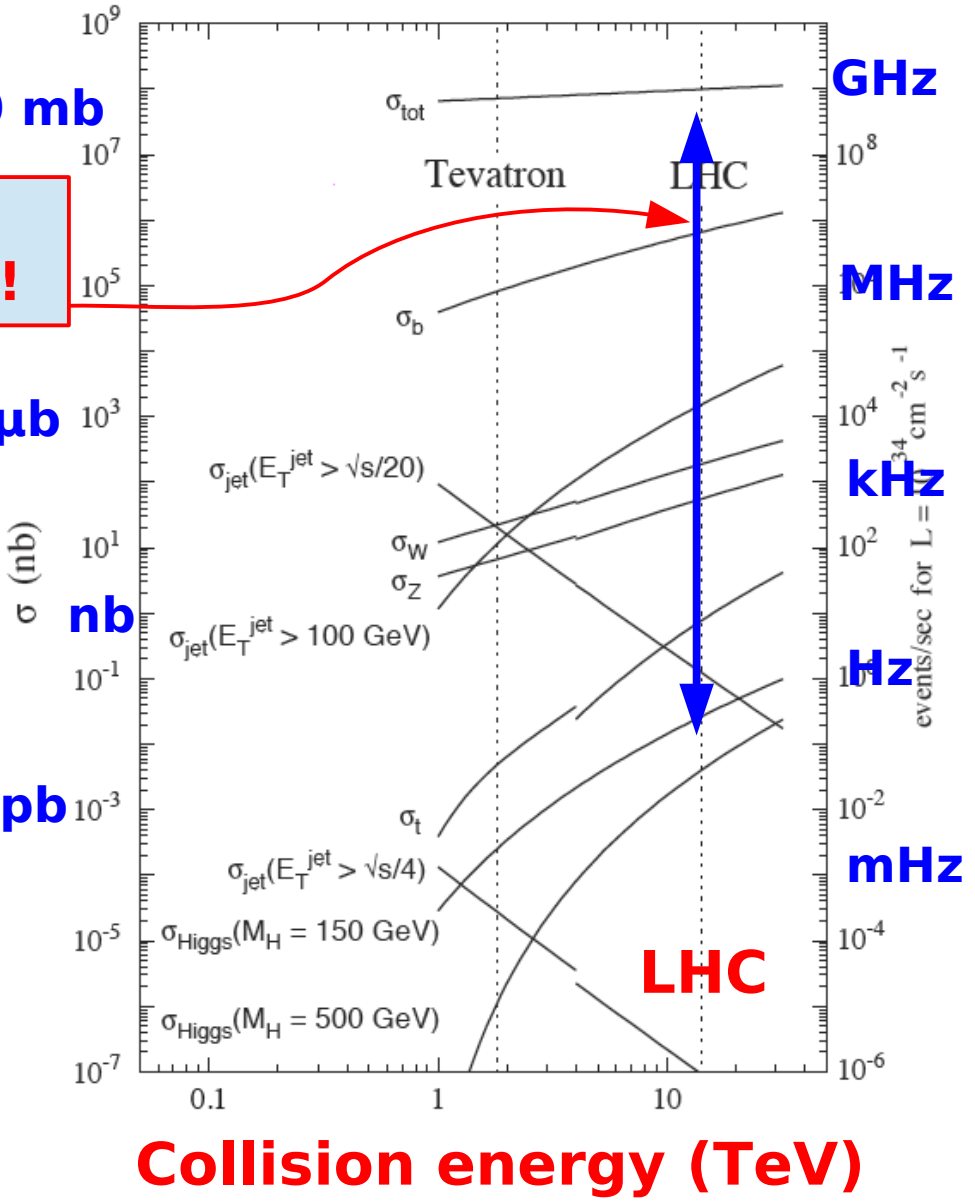
For proton bunch spacing of 25ns:

pp interactions/bunch crossing: ~25

(called "pile-up" events)

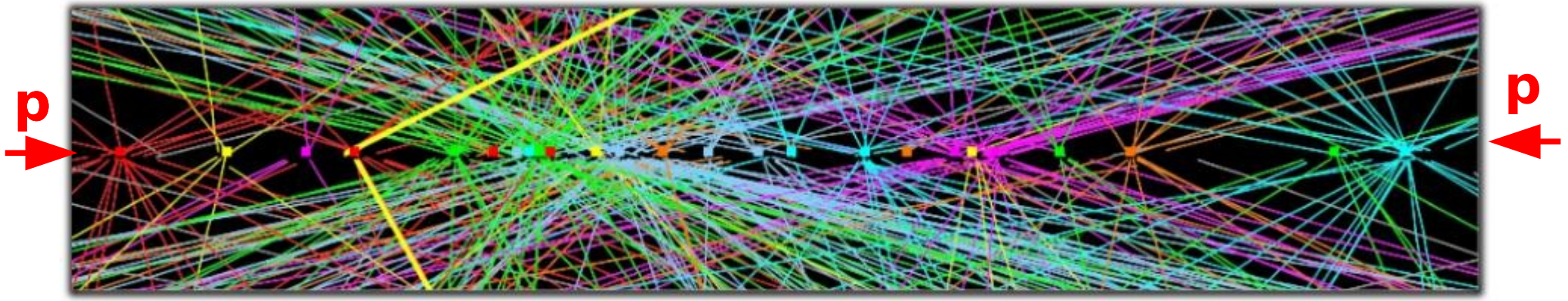
Luminosity has already reached ~2 times more (~3 next Run)

To reach 6-8 times more in High Lumi LHC (year 2026+) → 140-200 pile-up



Looking at many & complex events

every 25ns two proton bunches cross each other
→ a superposition of >25 pp collisions



Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.

The Trigger and Data Acquisition system,

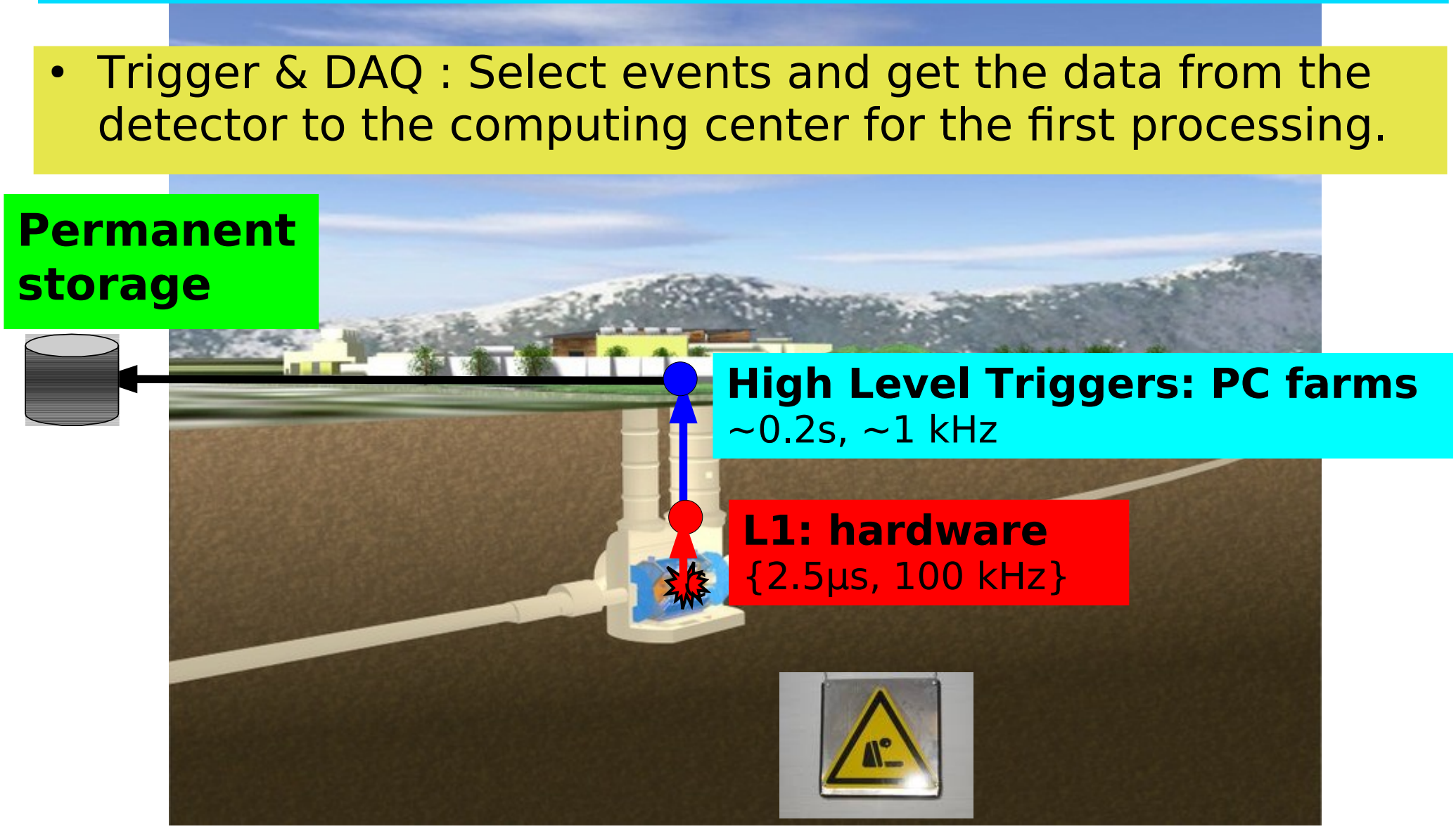
* **watches 40M such "events" (bunch crossings) / sec**
  → O(1) billion pp interactions per second

* **select online "the most interesting" O(1k) events/sec**
  (1 : 1 Million pp interactions deemed interesting enough to keep)

* **and log them for offline use with a resolution of a ~100 Mpixel camera (100M channels: total ~1.5 MB/event)**

Trigger at 2 stages:
Level1 (L1: fast, no detailed info) &
High Level Trigger (HLT: slower, using detailed info)

- Trigger & DAQ : Select events and get the data from the detector to the computing center for the first processing.

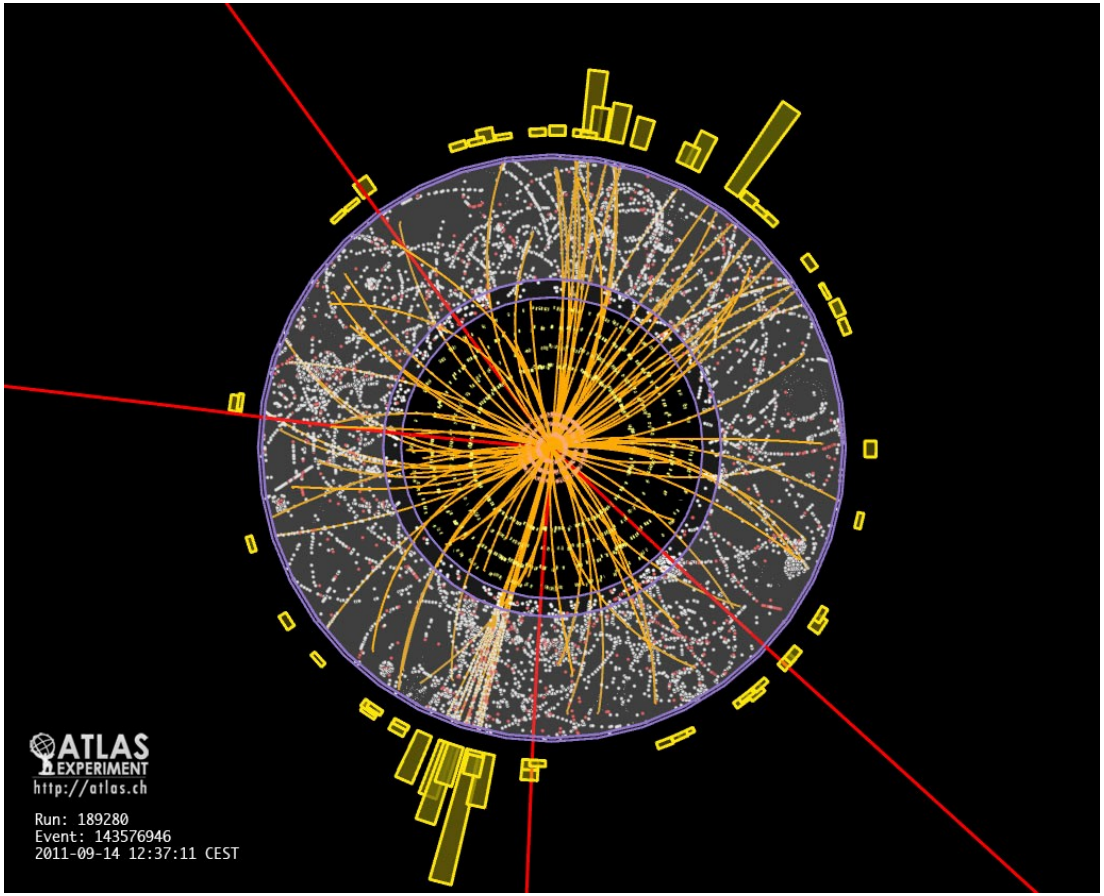


Example: Looking for Higgs

- **How do we see the Higgs?**

  → from its children!

  E.g.., 4 muons traversing the detector (red lines here)

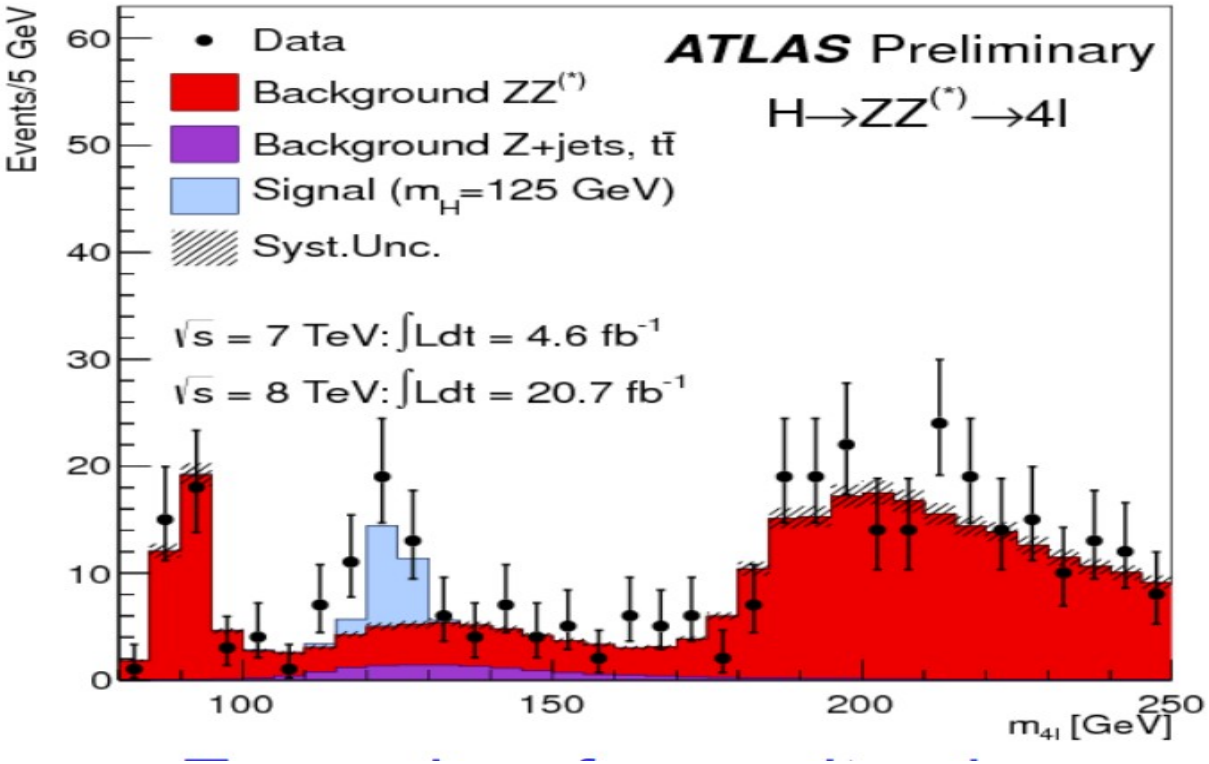


$E = mc^2 \rightarrow E^2 = m^2c^4 + p^2c^2 \rightarrow E^2 = m^2 + p^2 \rightarrow m = \sqrt{(E^2 - p^2)}$

Selection of the right events is essential (and selection of many of them too!) e.g., ATLAS: H → ZZ → 4l

./4l-FixedScale-NoMuProf2_400x300.gif

./4l-FixedScale-NoMuProf2_238x231.gif



https://twiki.cern.ch/twiki/pub/AtlasPublic/HiggsPublicResults//4l-FixedScale-NoMuProf2.gif

The more you know about the events, the easiest you select the “signal” and reject the “background”

When there is limited time budget (L1 trigger): typically, decide based only on the muon and calorimeter systems

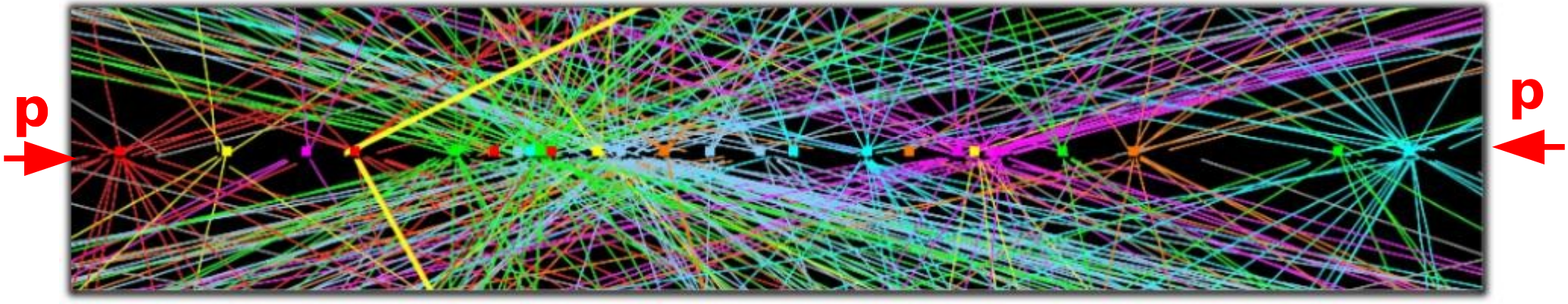
But may need information from the inner tracker as early as possible to make an “educated” decision and keep as much signal as possible

e.g., 2 “jets” of tracks, which are usually boring, they could actually be

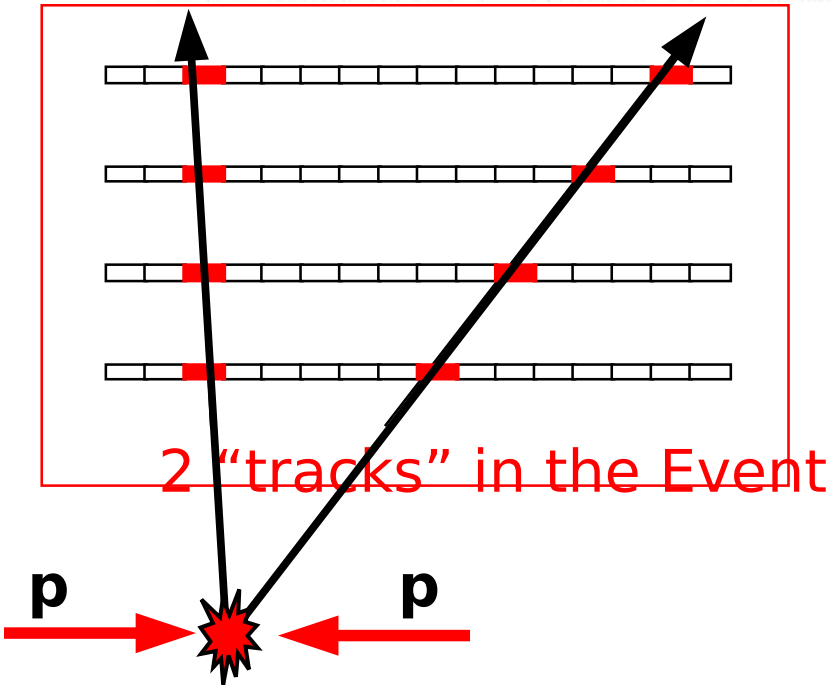
$$H \rightarrow b\bar{b} \quad \text{or} \quad H \rightarrow \tau^{+}\tau^{-}$$

You will hear on the last day from **Francesca Pastore** **the way various experiments trigger on the “interesting events”**

Each charged particle leaves a trace (“a track”) in the detector as it moves outwards

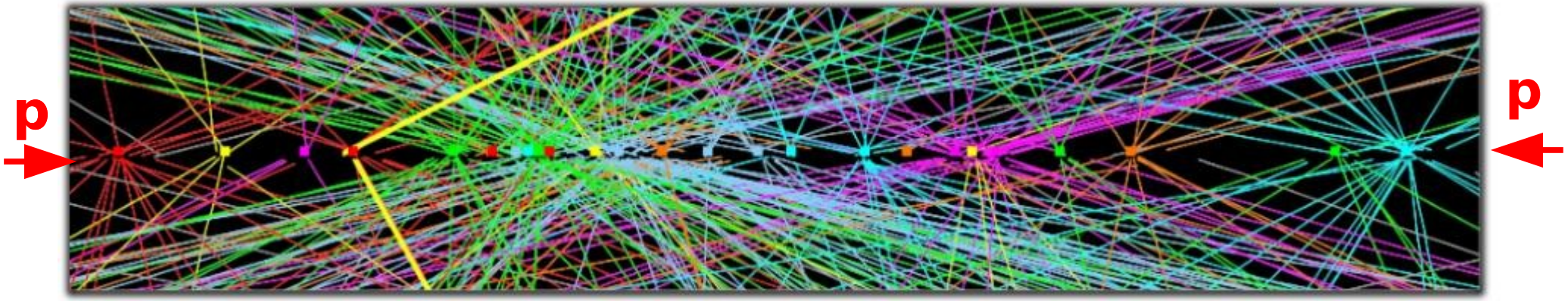


Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.

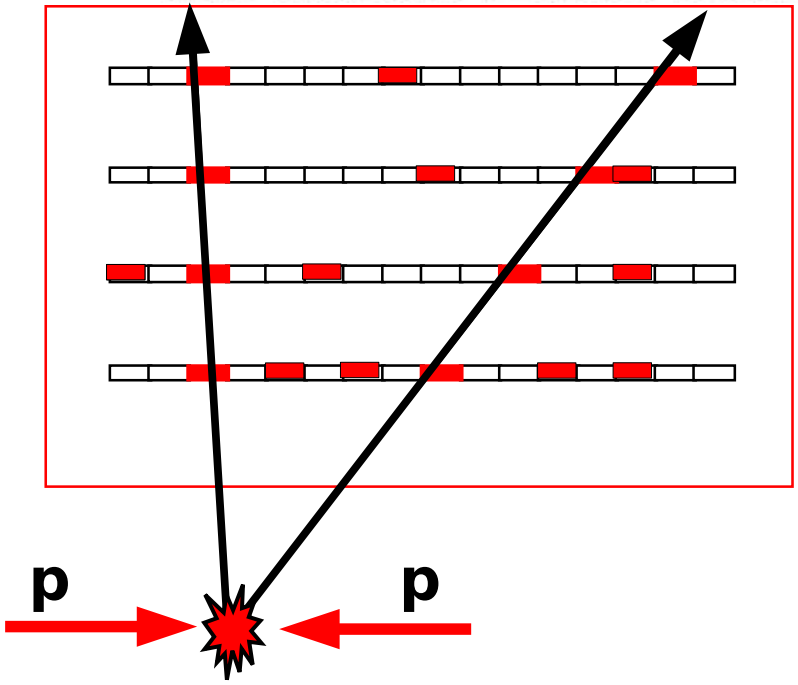


- A “track”
  traces the charged particle's path as it moves radially outward and its' position is measured in each detector layer
- Find the “track” by
  associating the relevant “hit” cells from one detection layer to the other
- Measure the track parameters
  by fitting these hit positions

You have also noise and irrelevant hits on the same “event”/“picture”

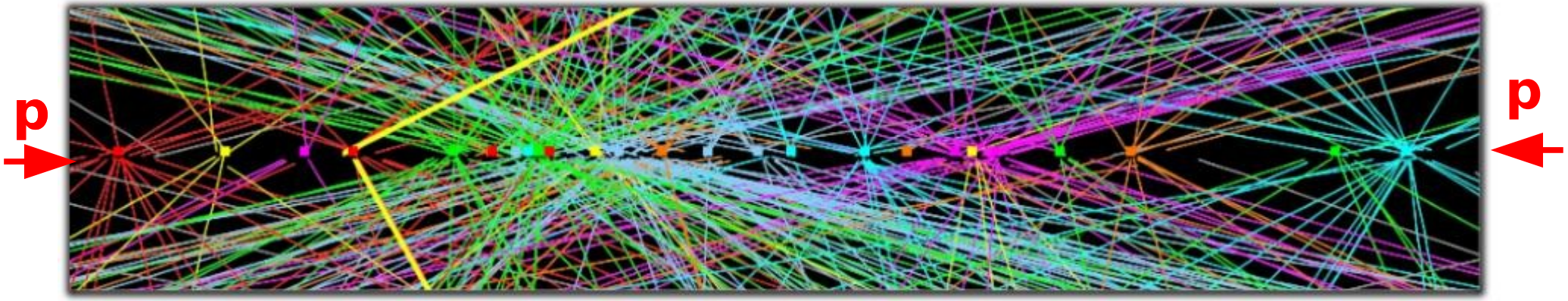


Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.

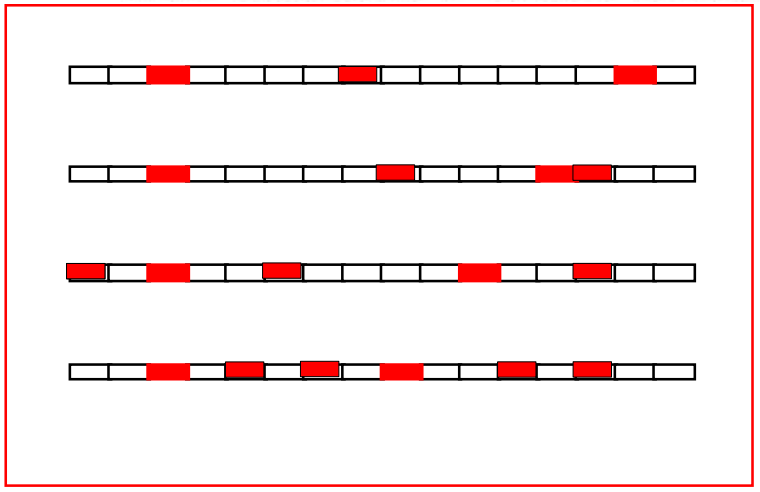


2 “real tracks” + extra hits

Tracking is a combinatorics problem: which combinations of hits fit track hypothesis?



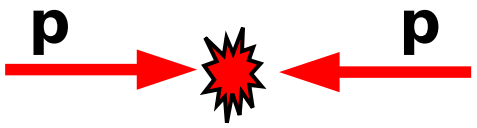
Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.

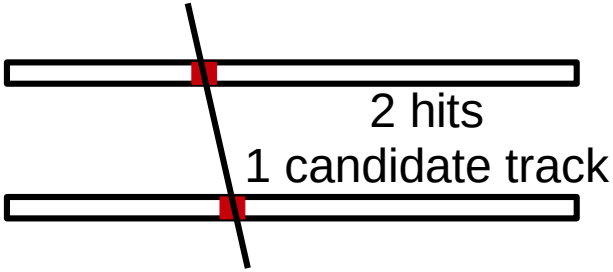


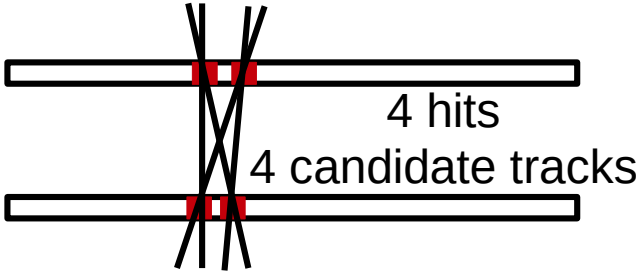
But when you look
at this event/picture,
you just see hits!

You have to find the tracks...

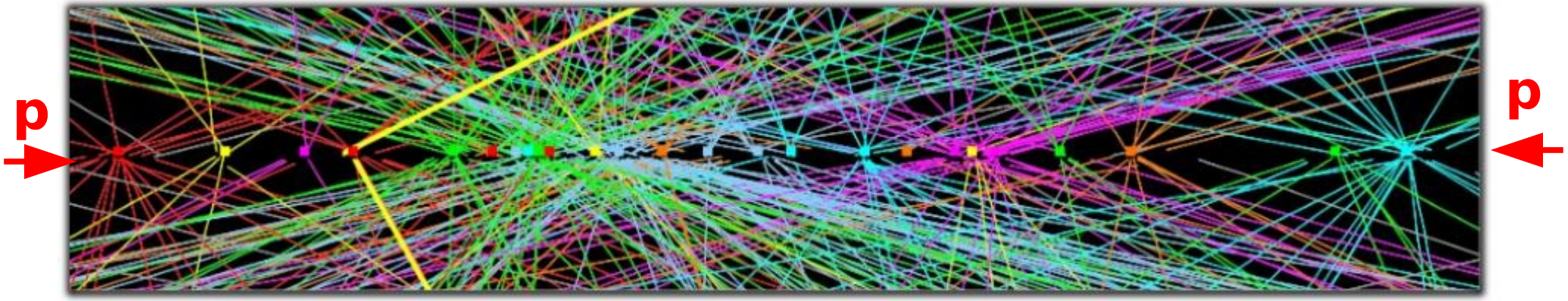
Number of possible tracks do not scale linearly with number of hits. e.g.:



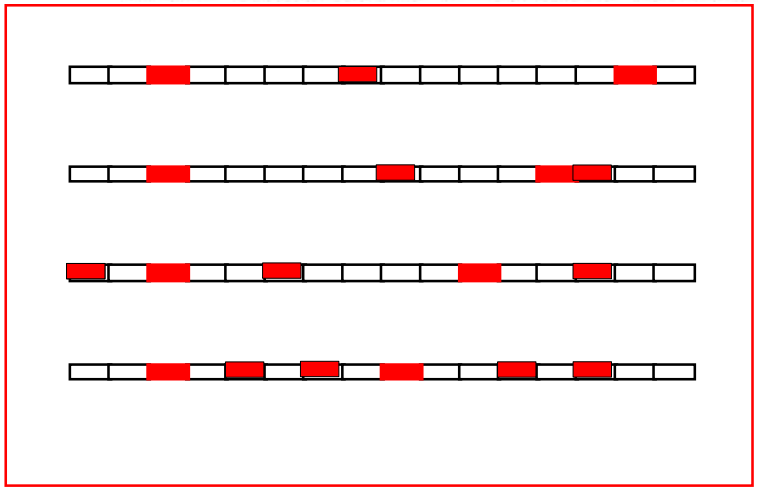


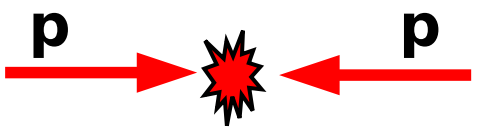


Tracking is a combinatorics problem: which combinations of hits fit track hypothesis?



Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.



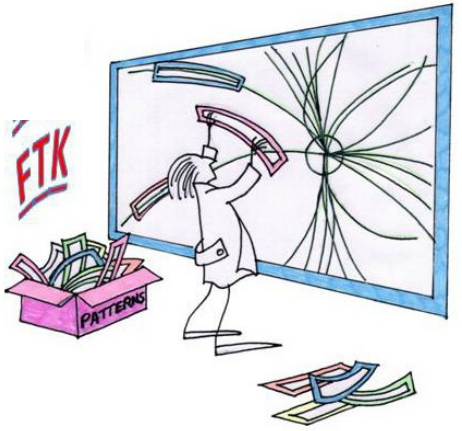


But when you look
at this event/picture,
you just see hits!

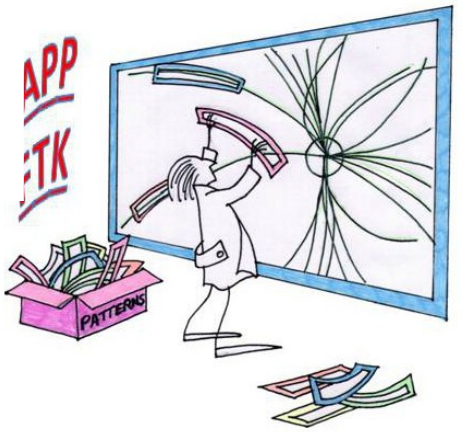
You have to find the tracks...

→ Lots of hit combinations to try
→ a huge combinatorics problem
→ becoming worse and worse
as luminosity increases
→ a big burden on CPUs

B. Track finding with Pattern Matching in Associative Memories
&
track fitting with linearized track fitting in FPGAs

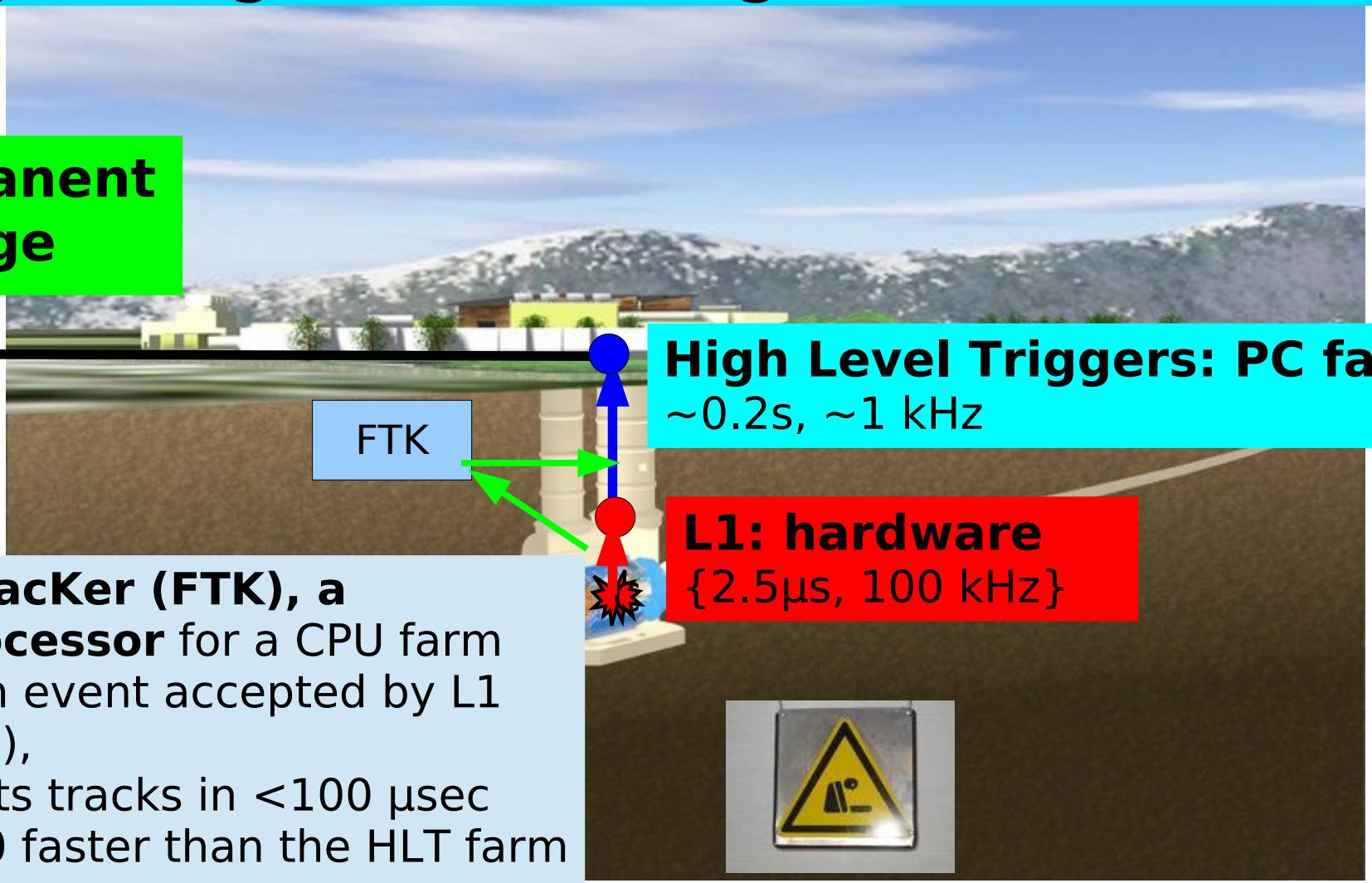


B. Track finding with Pattern Matching in Associative Memories & track fitting in FPGAs



- I give solutions adopted for the Fast TracKer (FTK) and the Hardware Tracking for the Trigger (HTT) projects of ATLAS as an example
  - FTK: a hardware *pre-processor* finding tracks and storing them for further usage by the trigger
  - HTT was born out of FTK, but is a hardware *co-processor* who is ordered by other components to the find tracks for them

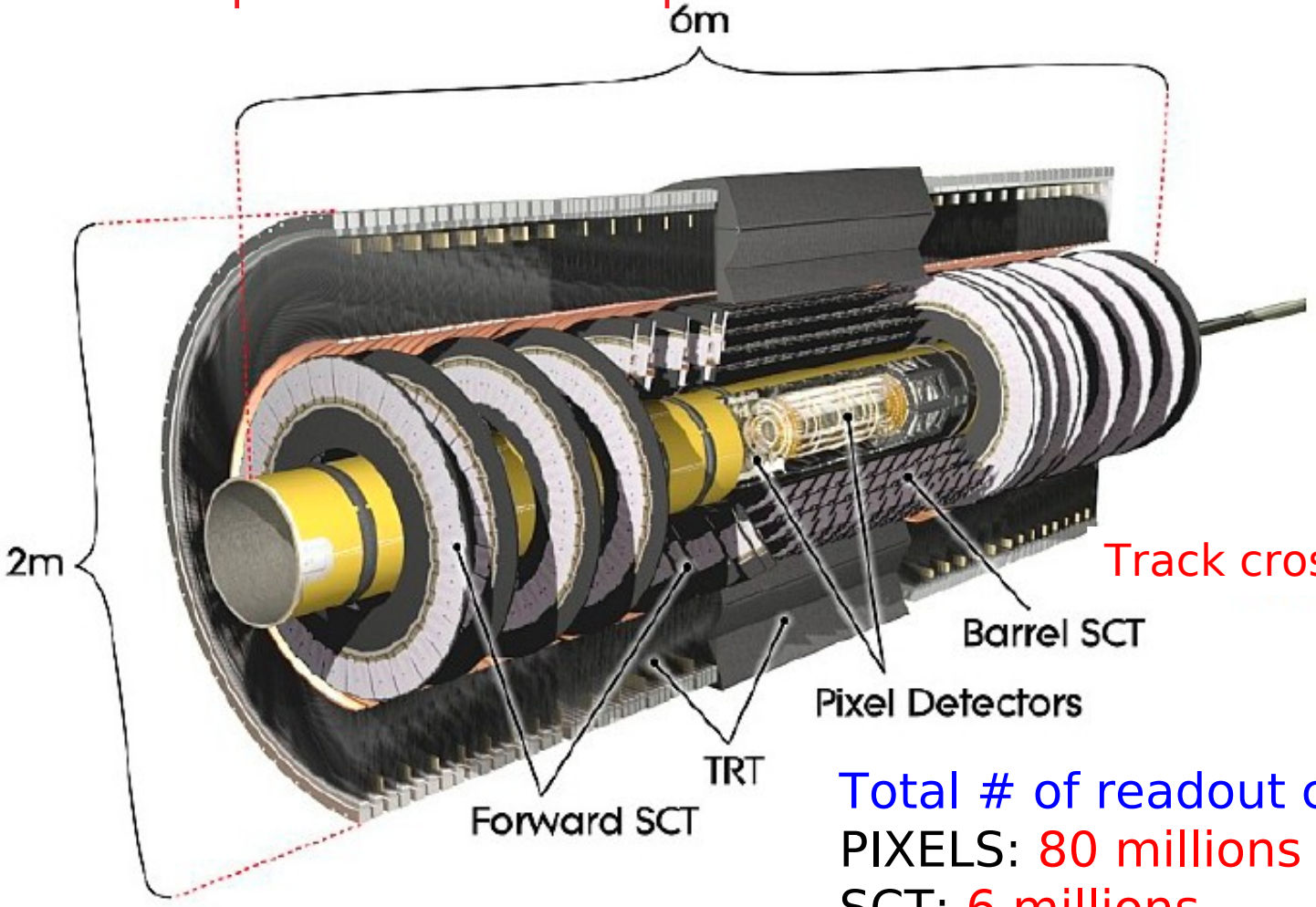
FTK (Fast TracKer): dedicated hardware helping the HLT, by doing the tracking before the HLT



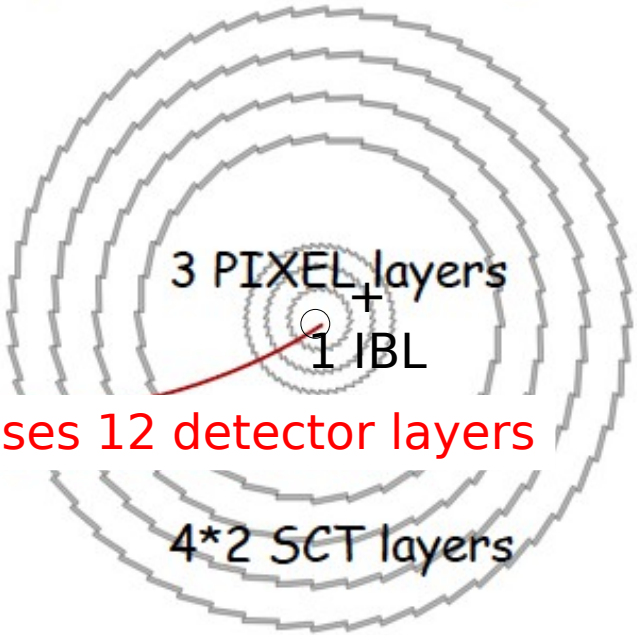
Fast TracKer (FTK), a pre-processor for a CPU farm
For each event accepted by L1 (100kHz),
find all its tracks in <100 µsec
→ x1000 faster than the HLT farm of PCs

FTK: Tracking particles in the Silicon Detectors

ATLAS' Fast TracKer (FTK) processes all Level-1 accepted events (100kHz)
Output: all tracks w/ pT>1 GeV available to HLT. Typical FTK latency ~100μs, compared to O(50ms) HLT
*** high-bandwidth connections with detector
*** HW optimized for the specific tasks



Example:
R-phi view of Barrel region:

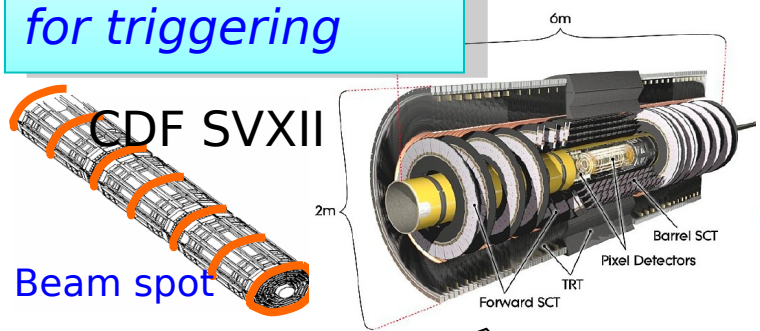


Track crosses 12 detector layers

Total # of readout channels: 98M
PIXELS: 80 millions + IBL: 12M
SCT: 6 millions

From hits to tracks in < 100 μs

Detector design for triggering



For L2: SVT trigger at CDF
For HLT: FTK in ATLAS
For HLT & L1 : HTT in ATLAS

Data transfer

1. Data formatting & clustering

1. Here **FPGAs** cluster hits and get their centroid as the hit position.
They forward these data to proper Processing Units

2a. Track Finding

2a. Associative Memories (pattern matching)

2b. Track Fitting

2b. FPGAs

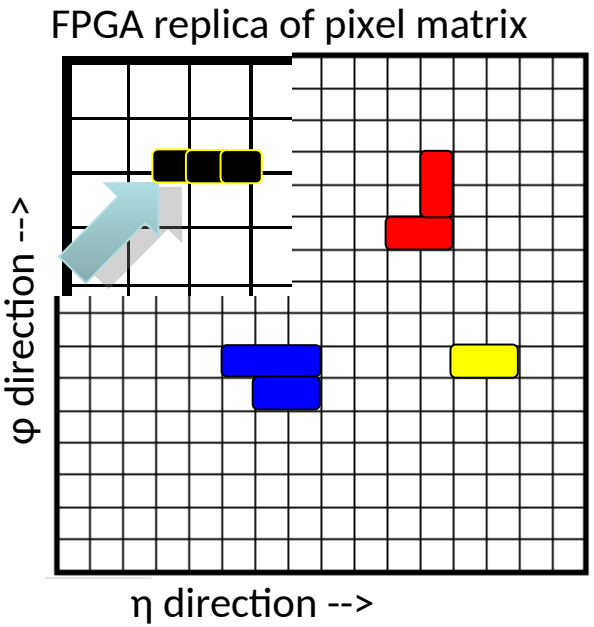
HLT

2. Processing Units (PUs) made of these two steps

Each PU, takes care of a given detector slice ("η-φ tower")
In FTK: 64 towers

1. Input & Data "Formatting":

cluster adjacent hits,
find the position of each cluster,
forward them to the Processing Unit
responsible for this geometrical η-φ region
(64 η-φ towers)



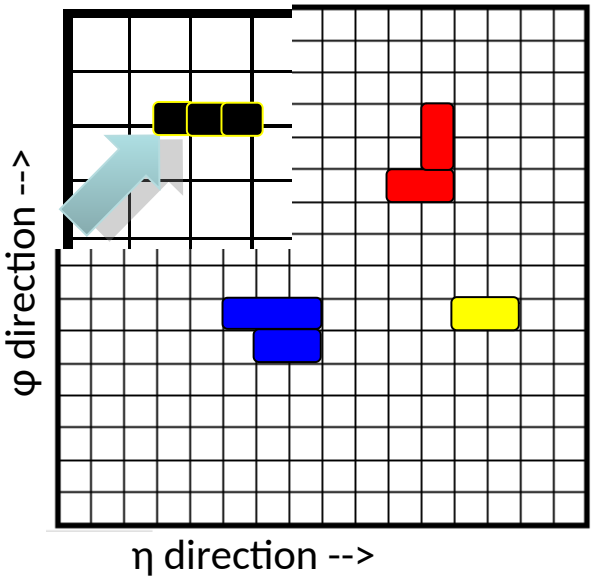
Significant **data reduction** by using hereafter only the position of each cluster

(in the example: from now on, instead of working with information from 14 cells, we work with information from 4 clusters)

Detail: Clustering algorithm how-to

NIM A617:254-257,2010
IEEE TNS, vol. 61, no.6, pp.3599-3606, 2014
doi: 10.1109/TNS.2014.2364183

FPGA replica of pixel matrix

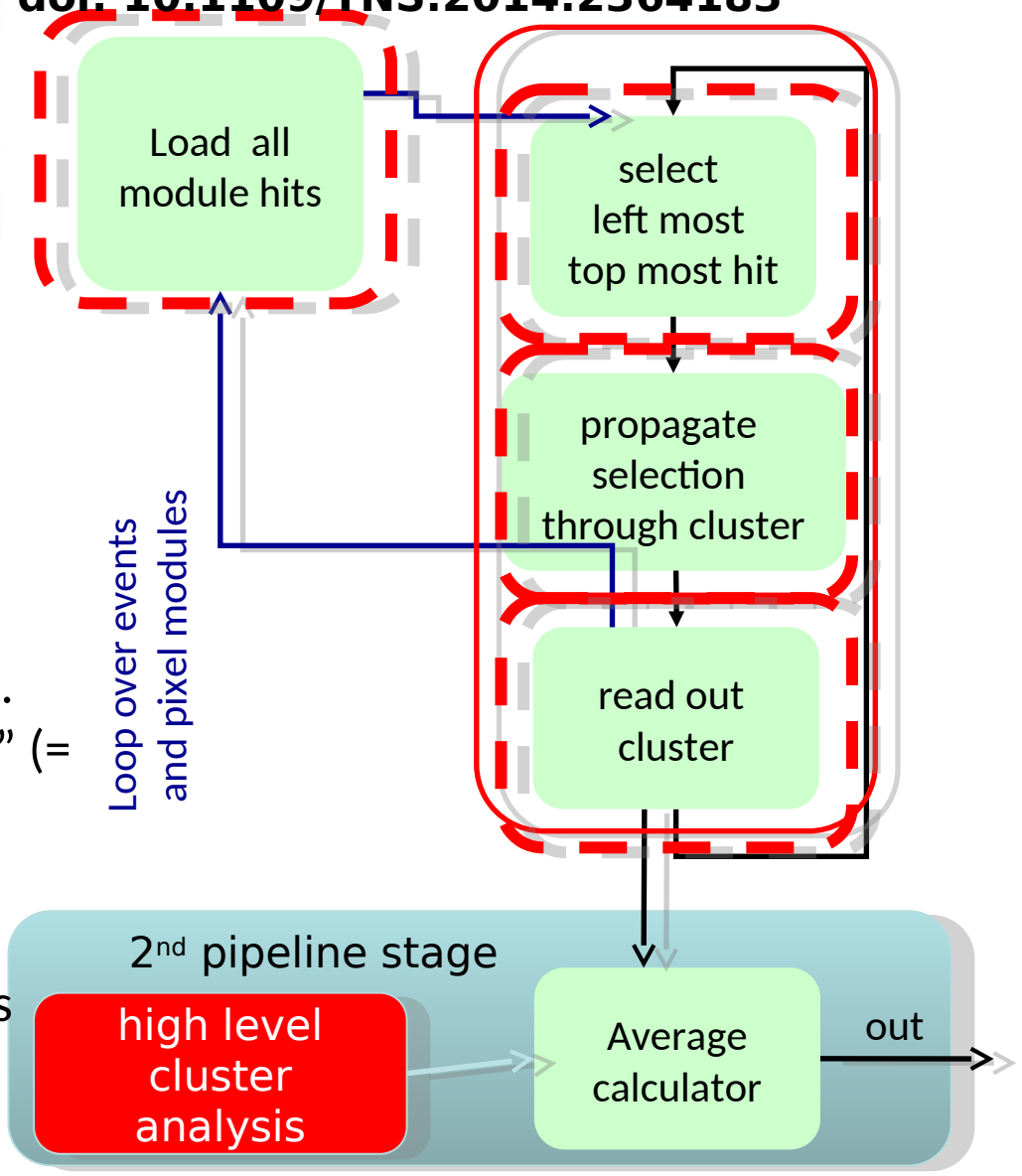


1st phase:

- Pixel module: a 328x144 matrix.
- Replicate a part of it (8x164) in hw matrix.
- Matrix identifies hits in the same "cluster" (= adjacent pixels)

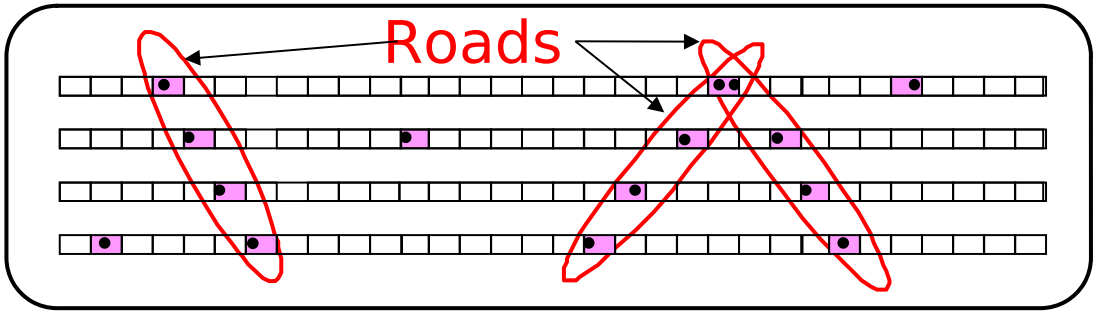
2nd phase:

- Hits in cluster analyzed (averaged) to get "the hit position", used in all next steps
- Flexibility to choose algorithm!



2. Processing Unit: tracking in 2 steps (see analogy to the Trigger doing L1 & HLT)

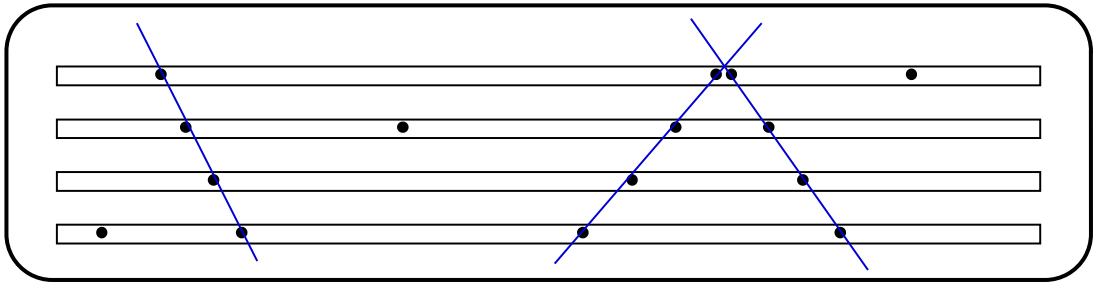
1. Find low resolution track candidates called "roads". Solve most of the combinatorial problem.



Pattern recognition w/ Associative Memory

Originally:
M. Dell'Orso, L. Ristori, NIM A 278, 436 (1989)

2. Then track fitting inside the roads. Thanks to 1st step, this is much easier.



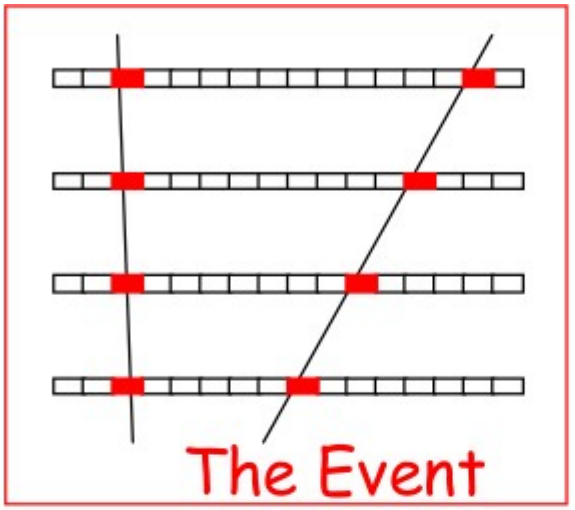
http://www.pi.infn.it/~orso/ftk/IEEECNF2007_2115.pdf

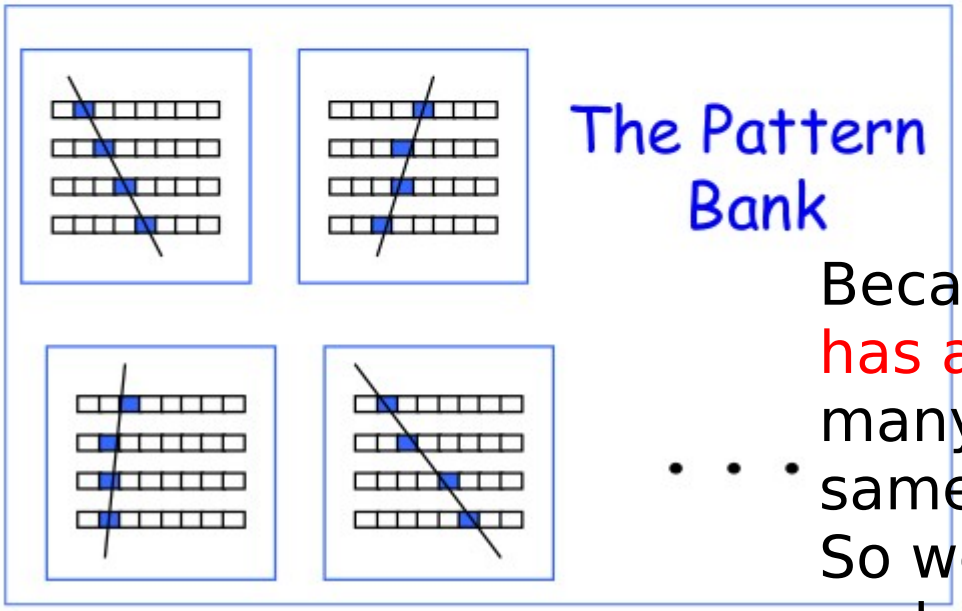
Excellent results with linear approximation!

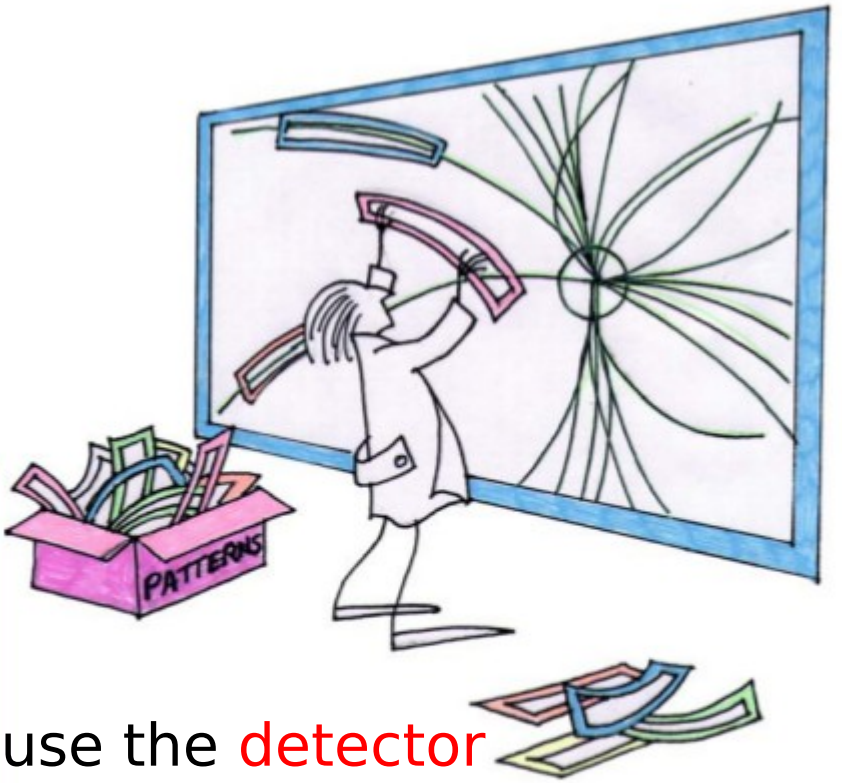
2a.
The coarse pattern matching first

In SVT, FTK and HTT: use 8 layers of the tracker

The detector's finite resolution makes it "binned"→ finite number of "hit patterns"

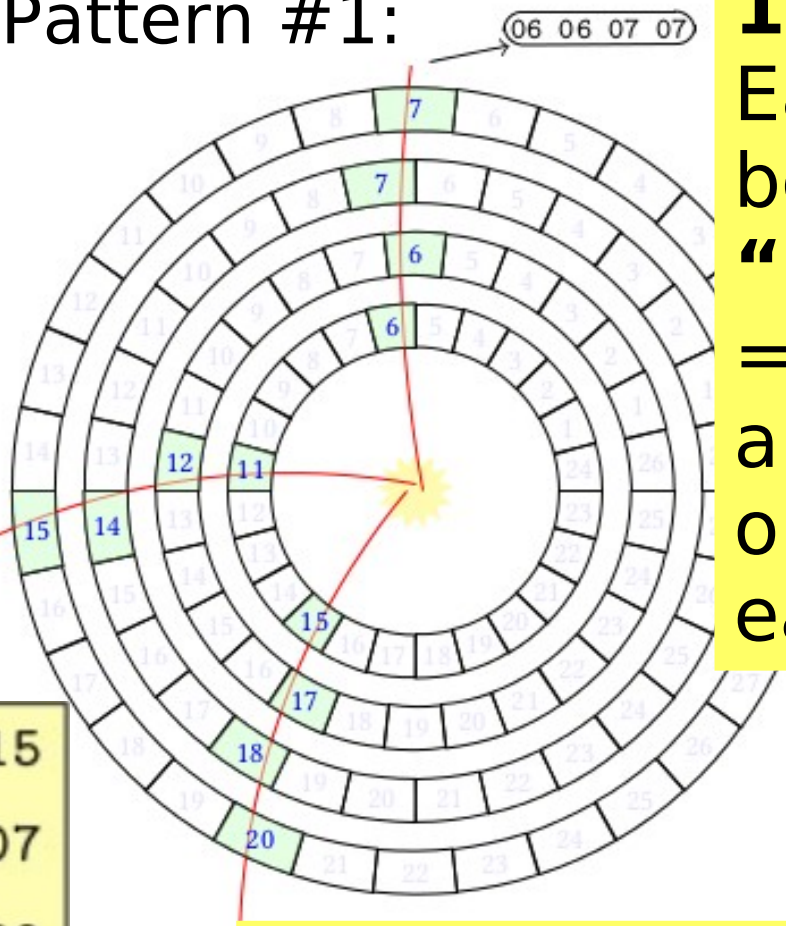






Because the detector has a finite resolution ("bin size"), many different tracks generate the same hit pattern,
So we have a finite number of patterns and a finite-size pattern-bank.

Training: simulated tracks to find possible patterns



Pattern Bank:

Patt0	11	12	14	15
Patt1	06	06	07	07
Patt2	15	17	18	20

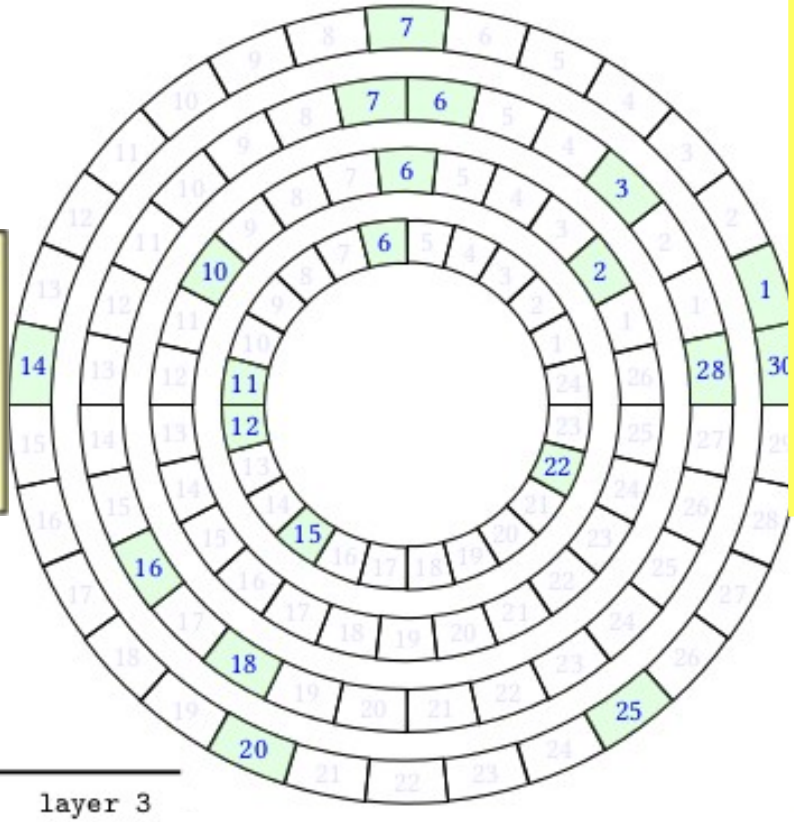
1.
Each possible track becomes a **“pattern”**
=
a series of numbers: one coordinate for each detector layer

2.
All patterns are stored in a **“pattern bank”**

Coarse track finding = pattern matching: does your event contain any of these patterns?

Pattern Bank:

Patt0	11	12	14	15
Patt1	06	06	07	07
Patt2	15	17	18	20



3.
Compare the hits in your event with the stored patterns

The "event"

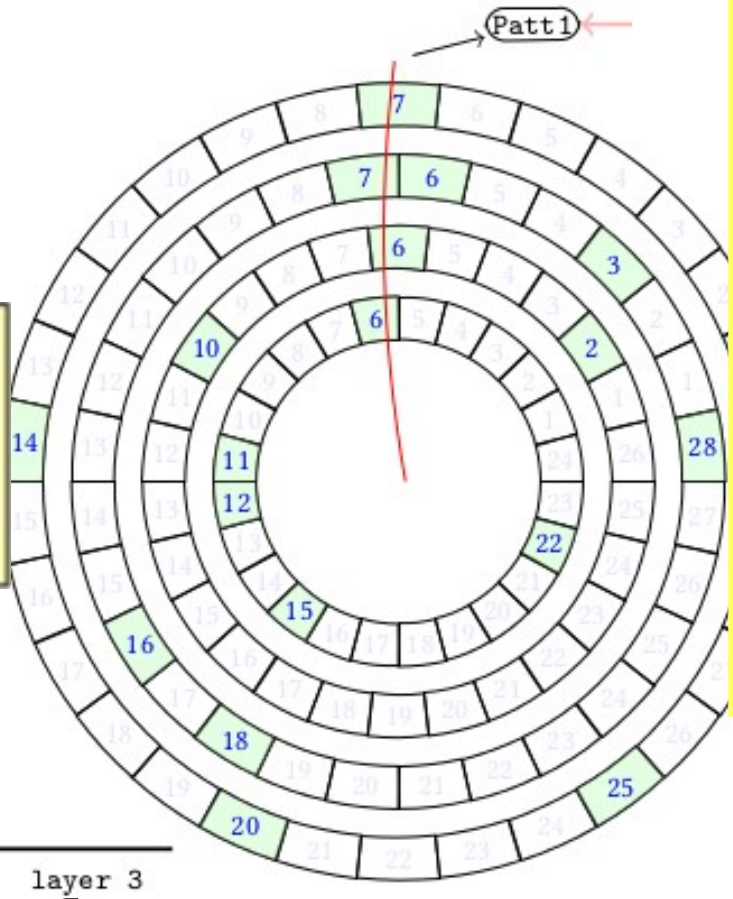
layer 0	layer 1	layer 2	layer 3
6	2	3	1
11	6	6	7
12	10	7	14
15		16	20
22		18	25
		28	30

The "event" is a list of hits in each detector **layer**

Compare ALL the hits in each event with ALL the stored patterns.

Pattern Bank:

Patt0	11	12	14	15
Patt1	06	06	07	07
Patt2	15	17	18	20



The event

layer 0	layer 1	layer 2	layer 3
6	2	3	1
11	6	6	7
12	10	7	14
...			

4.
After all comparisons are done, we have the **list of matched patterns** in the event =
the **list of "roads"** to perform refined tracking after

How to match data to patterns?

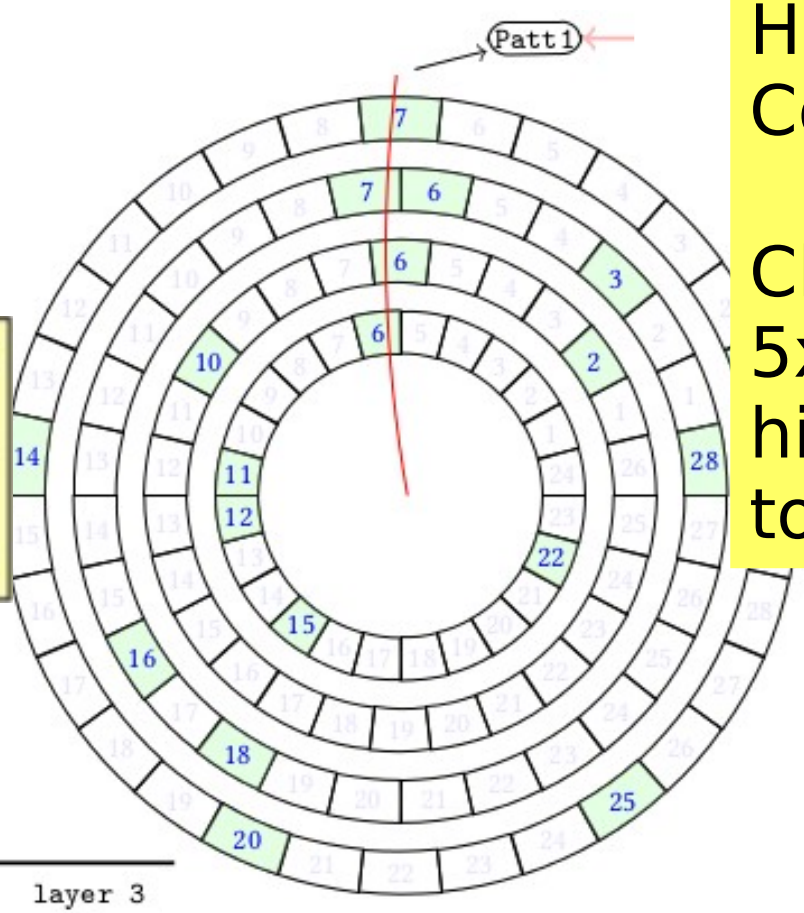
Pattern Bank:

Patt0	11	12	14	15
Patt1	06	06	07	07
Patt2	15	17	18	20

?

The event

layer 0	layer 1	layer 2	layer 3
6	2	3	1
11	**6**	6	**7**
12	10	**7**	14
15		16	20
22		18	25
		28	30



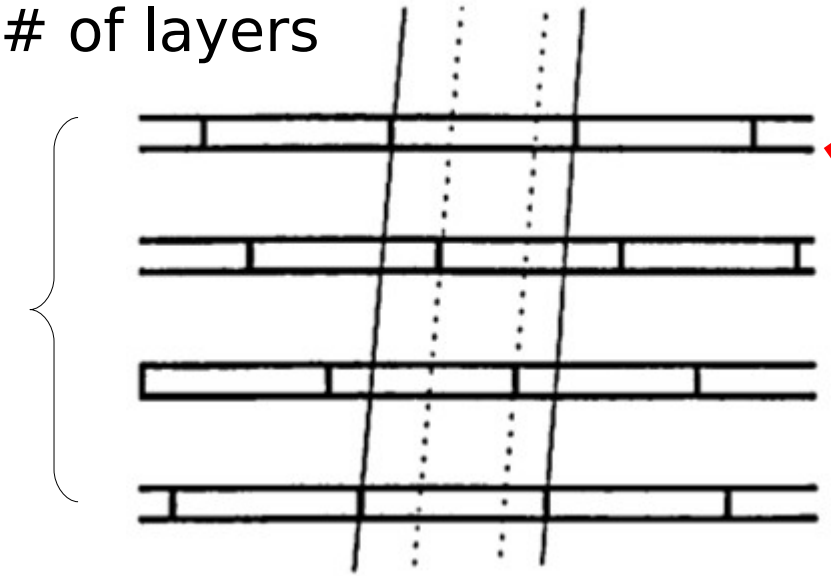
How to do the Comparison?

Check each of the 5x3x6x6 = 540 hit combinations to each pattern?

Number of patterns in your bank gets big easily

N_p = number of **straight lines** crossing the detector layers

m = # of layers



n= #bins per layer

$$N_p \simeq (m-1)\, n^2$$

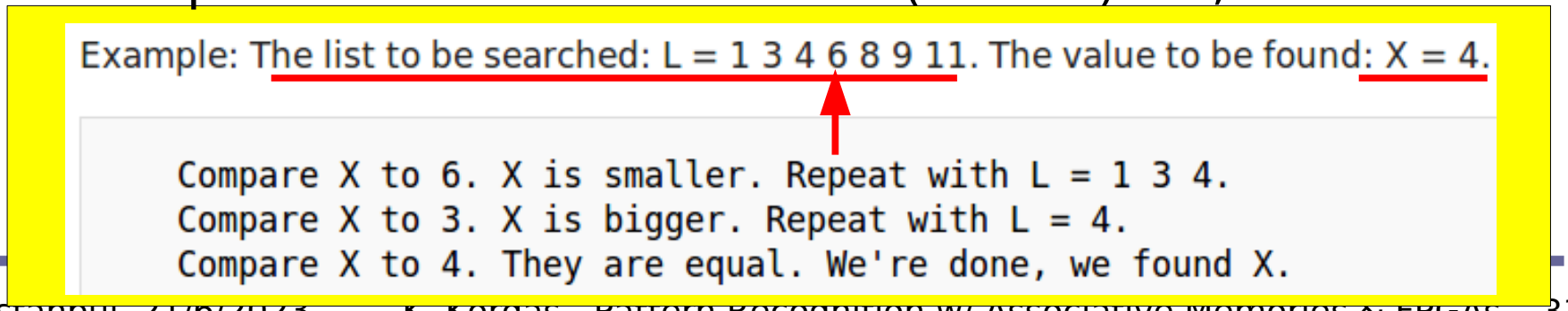
Can convince yourselves about this, with m=4 in the above drawing

For a detector with 8 layers, with 1M channels/layer, $N_p = 7\ 10^{12}$ **!!!**

(Re-bining with 2-channels per bin: n → n/2 means Np → ¼ Np)

$N_{patterns}$ and search time are critical

- Need a lot of memory for the patterns:
  - OK, can use larger (“coarser”) bins for 1st pattern matching (will come back to this later).
- But still, **you have to match hits with patterns fast**:
  - Linear search, of the pattern-table (“brute force”) is the slowest.
  - If list of patterns is ordered, can do “binary” search:
    - Pick the middle element in the list,
    - Compare the data to the pattern to find the good half of the list,
    - pick the middle of the new (halved) list, and so on.



Speed is extremely important at triggering.

Find tracks at ultimate speed

→ use "**Associative Memories**"

Ultimate speed for pattern matching:
do it during the I/O, as the data go through the system
→ no "processing time"

Associative Memory in a VLSI

436 Nuclear Instruments and Methods in Physics Research A278 (1989) 436–440
North-Holland, Amsterdam

October 24, 1988

VLSI STRUCTURES FOR TRACK FINDING

M. Dell'Orso, L. Ristori, NIM A 278, 436 (1989)

Mauro DELL'ORSO
Dipartimento di Fisica, Università di Pisa, Piazza Torricelli 2, 56100 Pisa, Italy

Luciano RISTORI
INFN Sezione di Pisa, Via Vecchia Livornese 582a, 56010 S. Piero a Grado (PI), Italy

Received 24 October 1988

We discuss the architecture of a device based on the concept of *associative memory* designed to solve the track finding problem, typical of high energy physics experiments, in a time span of a few microseconds even for very high multiplicity events. This "machine" is implemented as a large array of custom VLSI chips. All the chips are equal and each of them stores a number of "patterns". All the patterns in all the chips are compared in parallel to the data coming from the detector while the detector is being read out.

1. Introduction

The quality of results from present and future high energy physics experiments depends to some extent on the implementation of fast and efficient track finding algorithms. The detection of *heavy flavor* production, for example, depends on the reconstruction of secondary vertices generated by the decay of long lived particles, which in turn requires the reconstruction of the majority of the tracks in every event.

Particularly appealing is the possibility of having detailed tracking information available at trigger level even for high multiplicity events. This information could be used to select events based on impact parameter or secondary vertices. If we could do this in a sufficiently short time we would significantly enrich the sample of events containing heavy flavors.

Typical events feature up to several tens of tracks each of them traversing a few position sensitive detector layers. Each layer detects many hits and we must correctly correlate hits belonging to the same track on different layers before we can compute the parameters

2. The detector

In this discussion we will assume that our detector consists of a number of layers, each layer being segmented into a number of *bins*. When charged particles cross the detector they *hit* one bin per layer. No particular assumption is made on the shape of trajectories: they could be straight or curved. Also the detector layers need not be parallel nor flat. This abstraction is meant to represent a whole class of real detectors (drift chambers, silicon microstrip detectors etc.). In the real world the coordinate of each hit will actually be the result of some computation performed on "raw" data: it could be the center of gravity of a cluster or a charge division interpolation or a drift-time to space conversion depending on the particular class of detector we are considering. We assume that all these operations are performed upstream and that the resulting coordinates are "binned" in some way before being transmitted to our device.

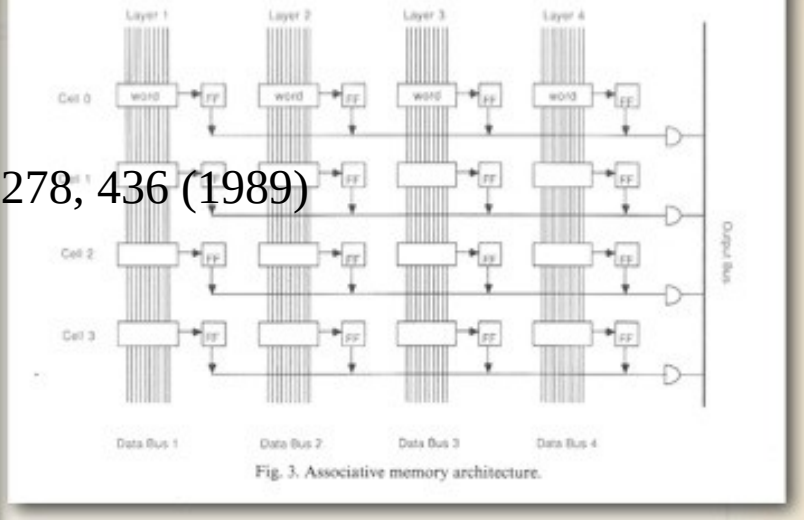


Fig. 3. Associative memory architecture.

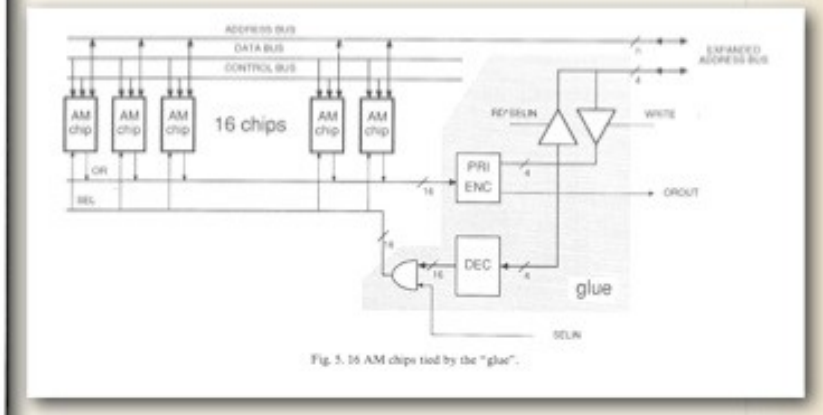


Fig. 5. 16 AM chips tied by the "glue".

We discuss the architecture of a device based on the concept of *associative memory* designed to solve the track finding problem, typical of high energy physics experiments, in a time span of a few microseconds even for very high multiplicity events. This "machine" is implemented as a large array of custom VLSI chips. All the chips are equal and each of them stores a number of "patterns". All the patterns in all the chips are compared in parallel to the data coming from the detector while the detector is being read out.

14

Associative Memory (AM) = a kind of Content Addressable Memory (CAM)

- CAM = a memory that is accessed by its **contents,** not its **location.**

- E.g., while in a RAM we ask:
  - what do you have in location ***xyz***?

- In a Content Addressable Memory (CAM) we ask:
  - Are there any locations holding the value ***abc***?

Binary CAMs

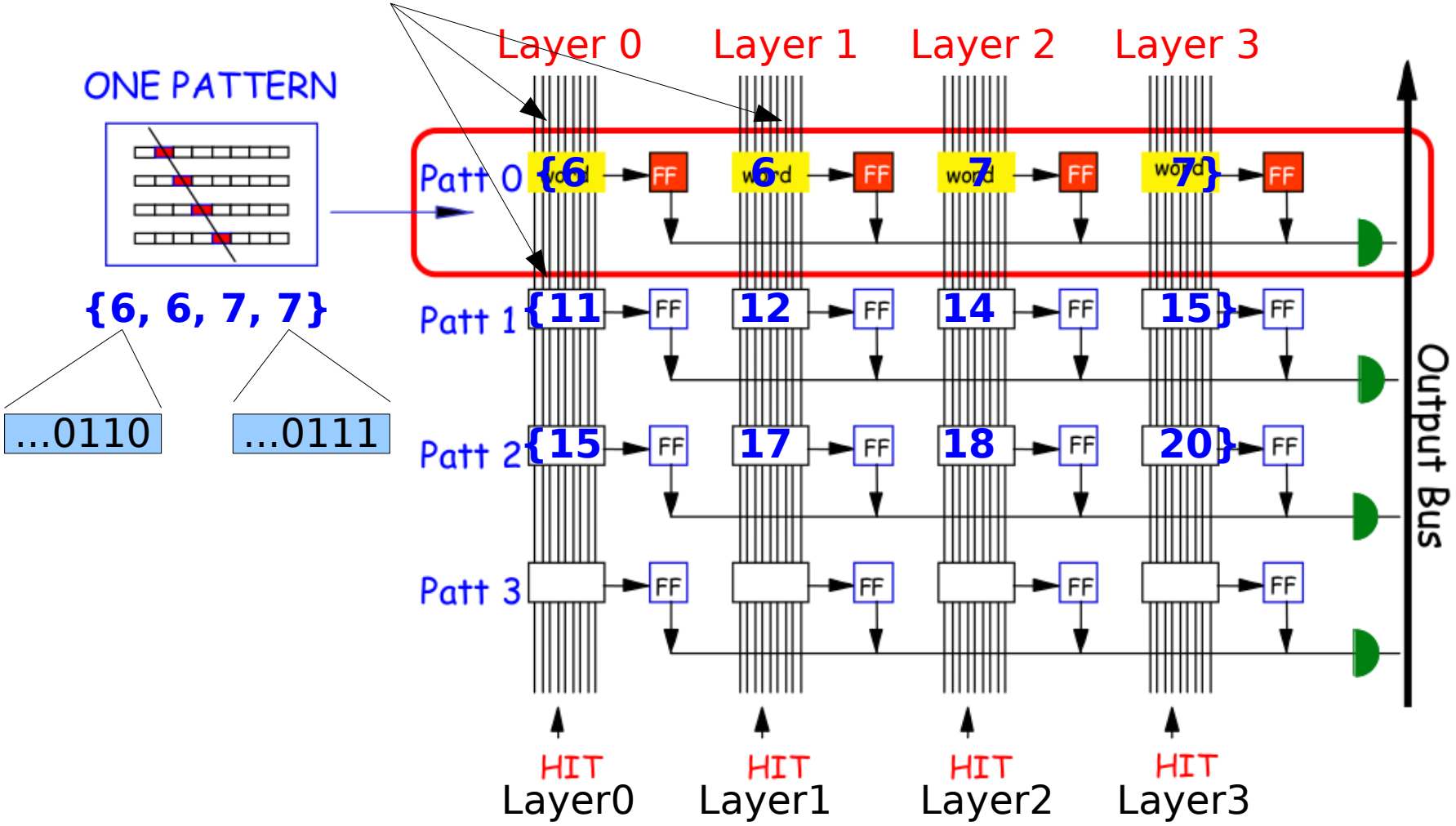
- **Binary CAM** (simplest):
  - uses search words consisting entirely of “1” and “0”

Example:

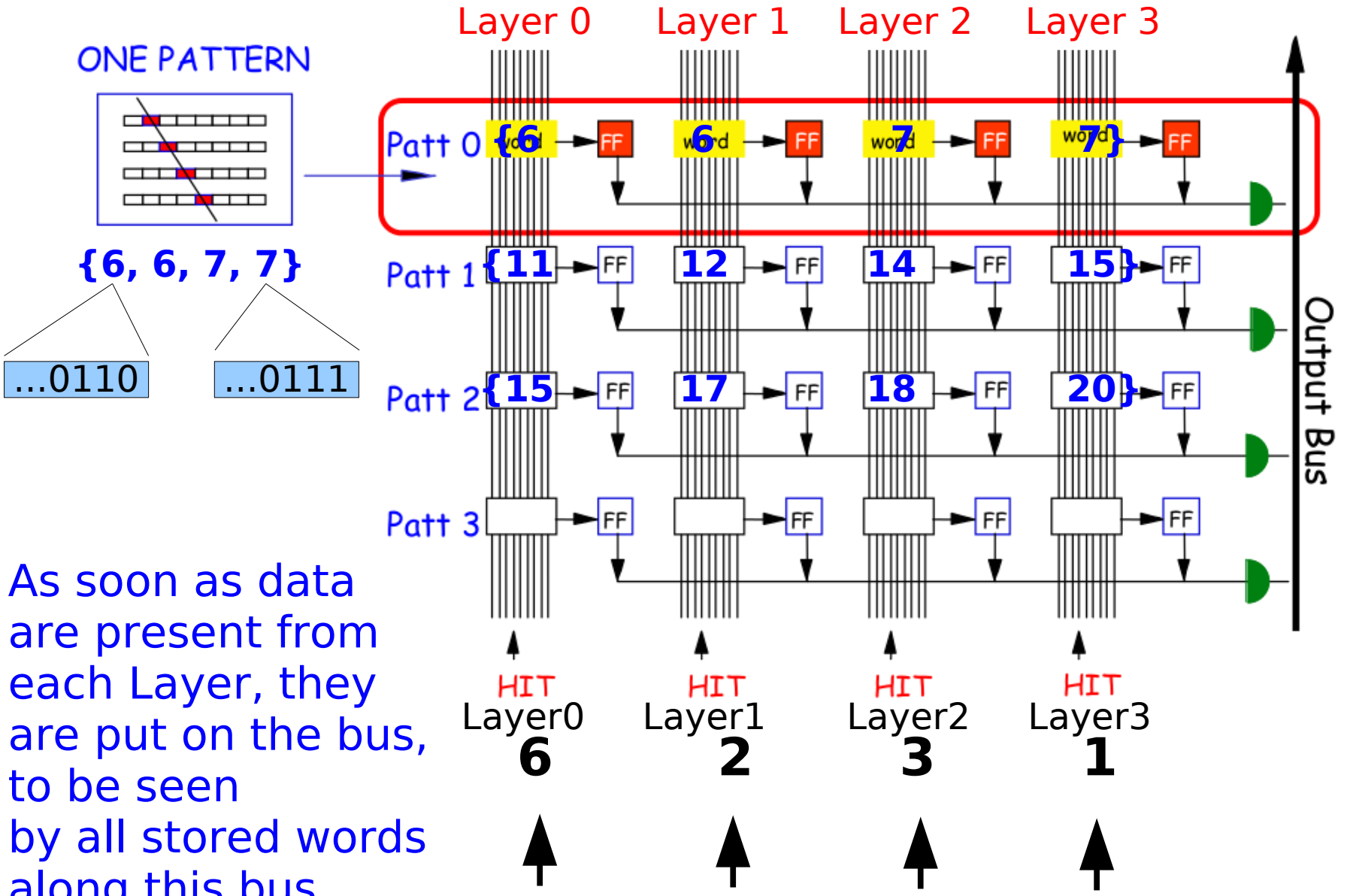
stored word of ----------> "**10110**" (“one pattern”)

It will be matched by the search word: "10110" (“the data”)

Associative Memory: CAM cells contain pattern bank. CAM cells of same Layer are on a common bus

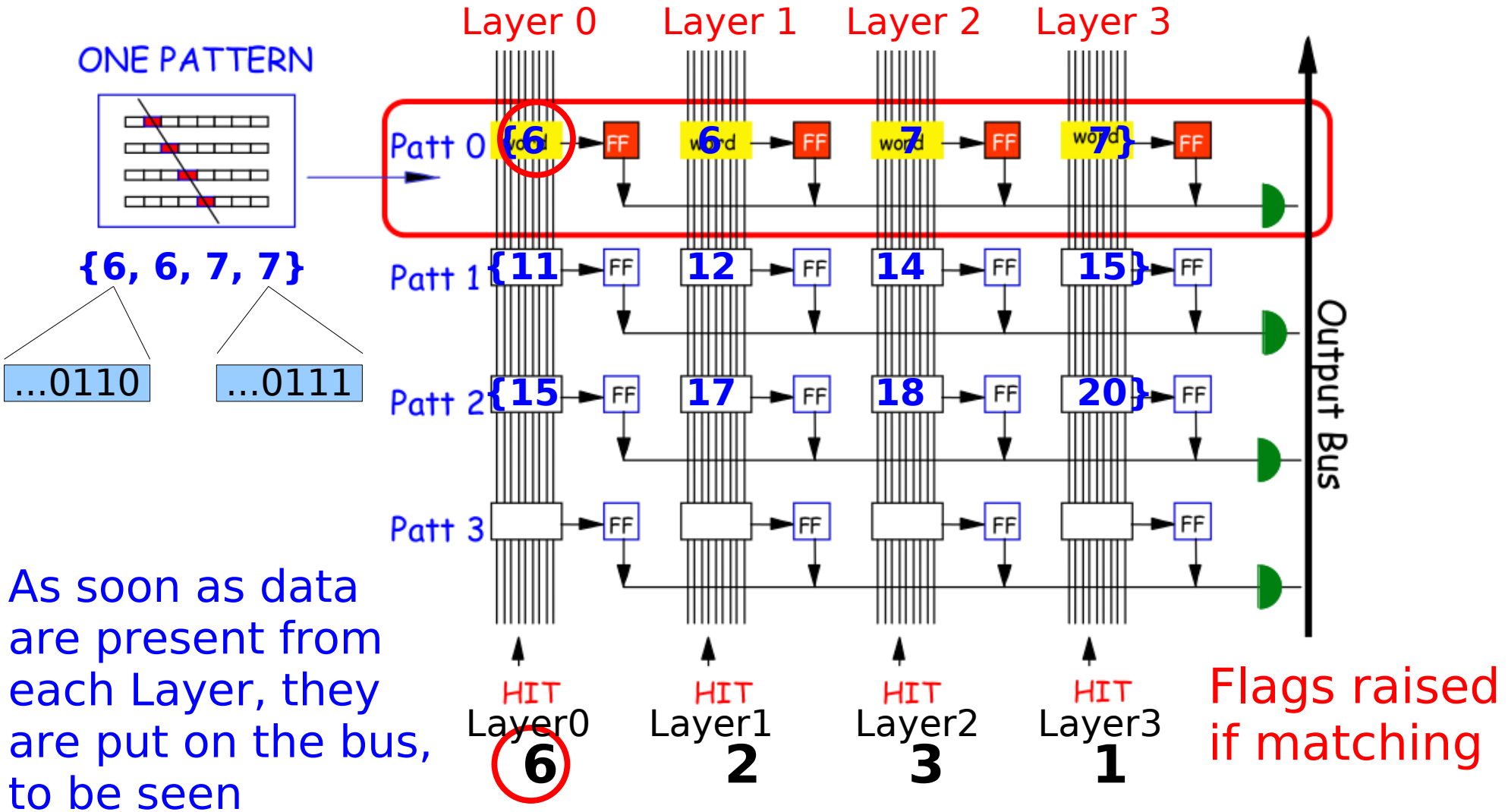


Associative Memory: CAM cells check the macthing of each hit independently



As soon as data are present from each Layer, they are put on the bus, to be seen by all stored words along this bus

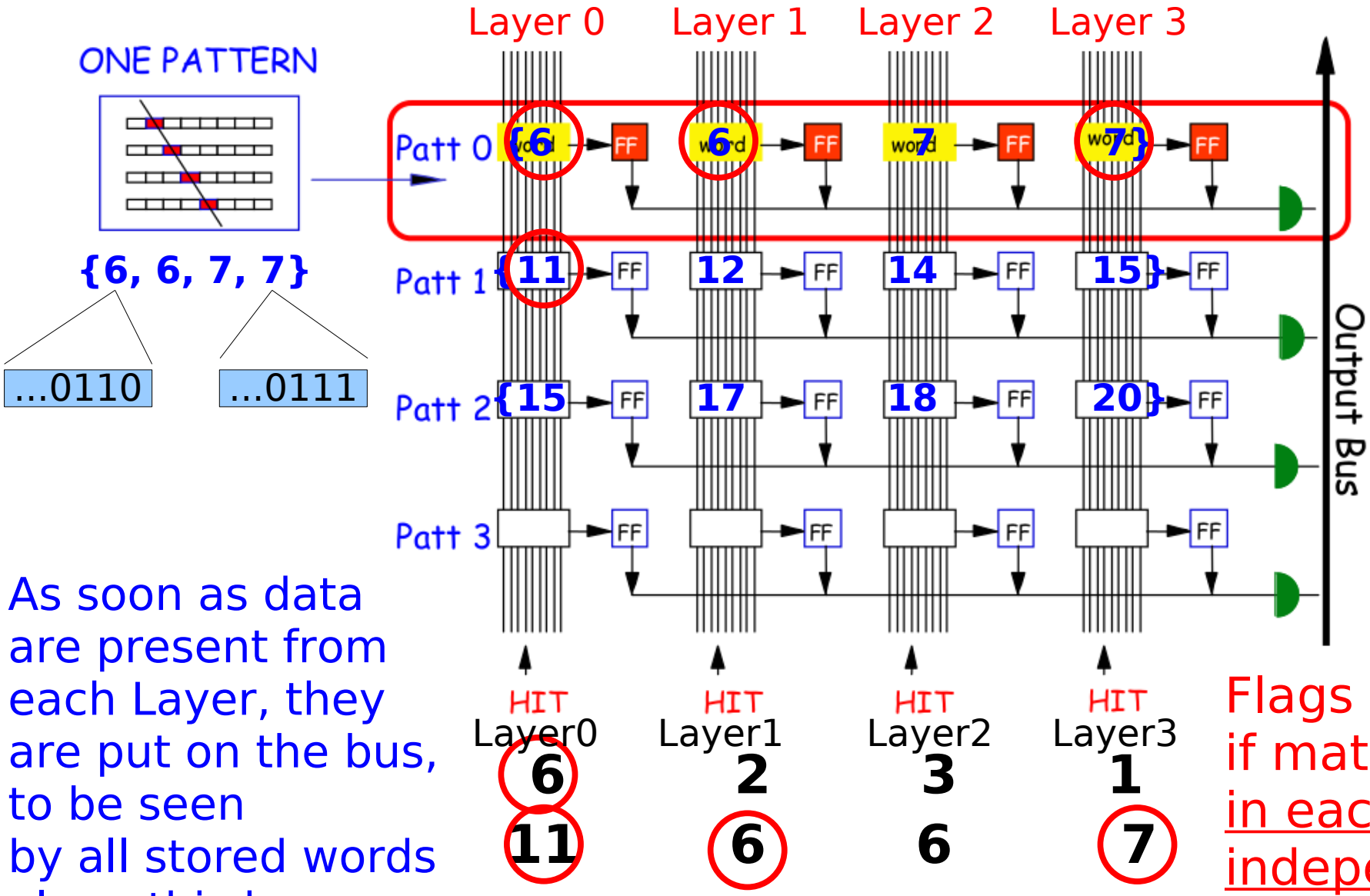
Associative Memory: CAM cells check the macthing of each hit independently



As soon as data are present from each Layer, they are put on the bus, to be seen by all stored words along this bus

Flags raised if matching

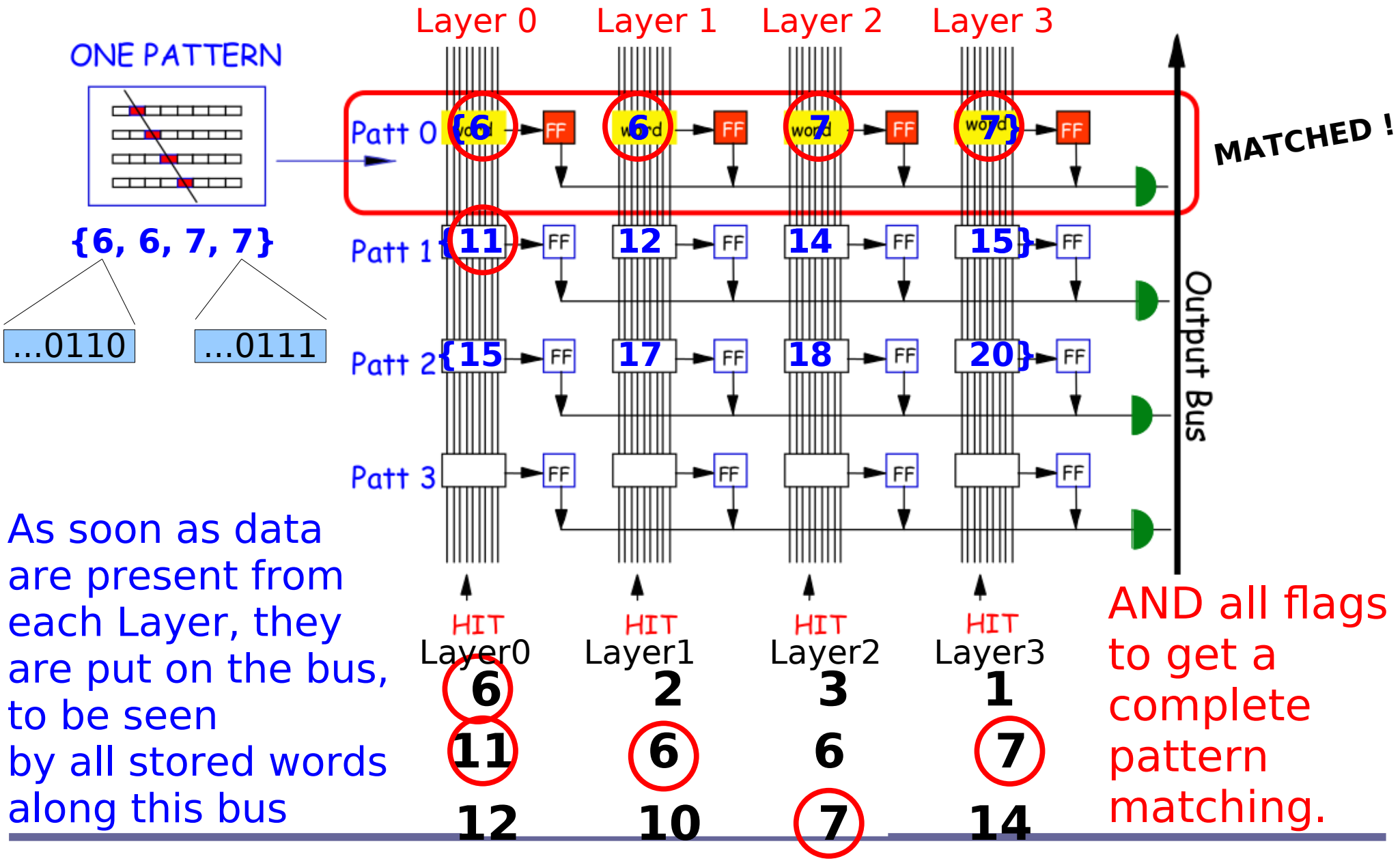
Associative Memory: CAM cells check the macthing of each hit independently



As soon as data are present from each Layer, they are put on the bus, to be seen by all stored words along this bus

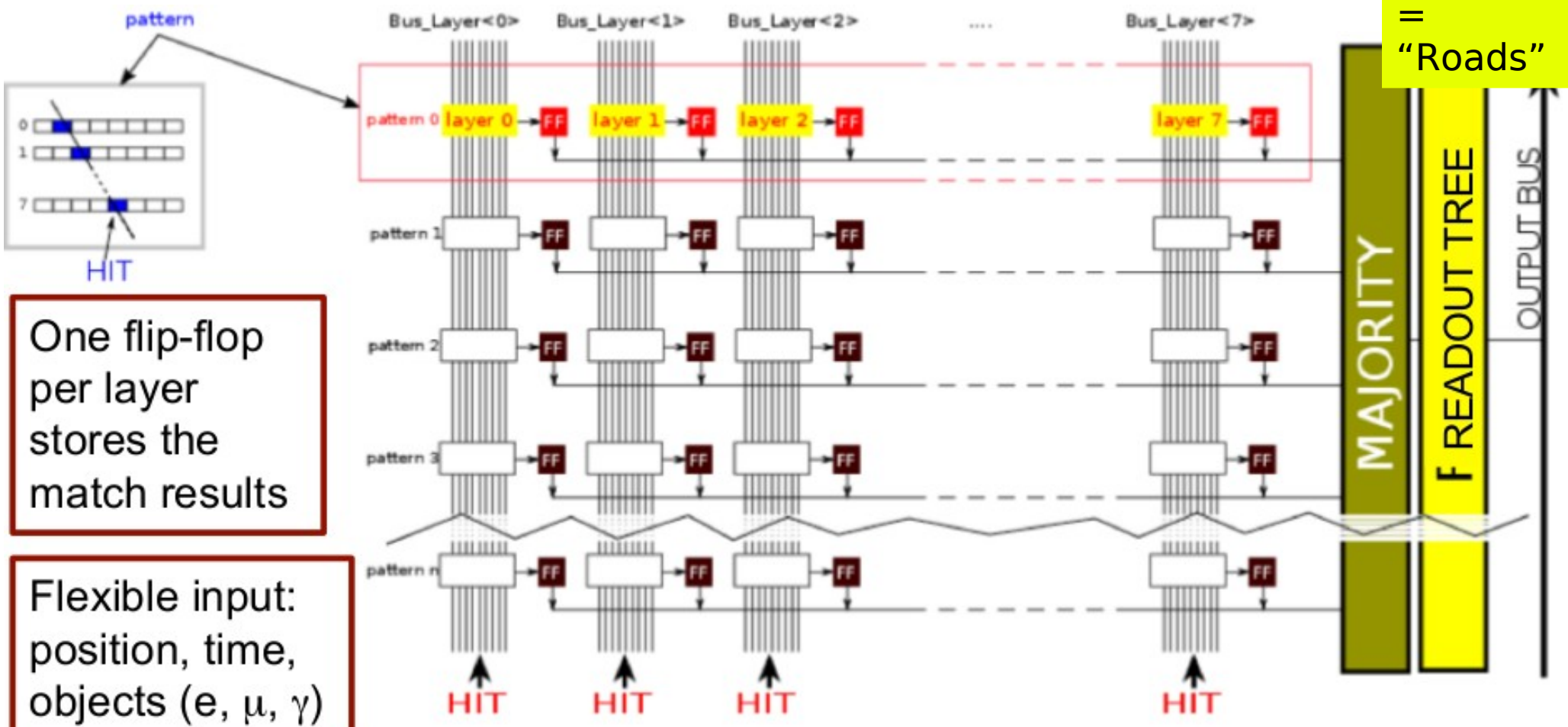
Flags raised if matching in each hit independently

Associative Memory: CAM cells check the matching of each hit independently



Track trigger w/ pattern matching AM

Result: Matched Patterns = "Roads"



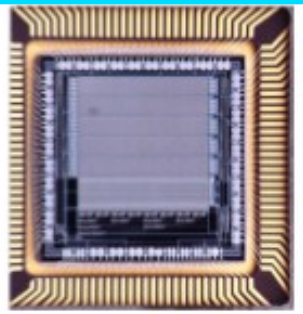
One flip-flop per layer stores the match results

Flexible input: position, time, objects (e, μ, γ)

Pattern matching is completed as soon as all hits are loaded.
Data arriving at different times is compared in parallel with all patterns.
Unique to AM chip: look for correlation of data received at different times.

AM evolution: ASICs (mainly)

SVT AM chip



- (90's) **Full custom VLSI chip -** 0.7μm (INFN-Pisa)
- **128 patterns, 6x12bit words each, 30MHz**

F. Morsani et al., IEEE Trans. on Nucl. Sci., vol. 39 (1992)

Alternative **FPGA** implementation of SVT AM chip

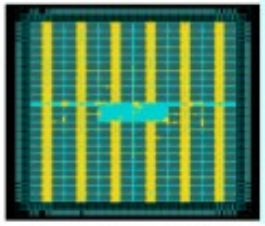
P. Giannetti et al., Nucl. Intsr. and Meth., vol. A413/2-3, (1998)

G Magazzù, 1st std cell project presented @ LHCC (1999)

SVT upgrade

Standard Cell 0.18 μm → 5000 pattern/AM chip
SVT upgrade total: 6M pattern, 40MHz
A. Annovi et al., **IEEE TNS,** Vol 53, Issue 4, Part 2, **2006**

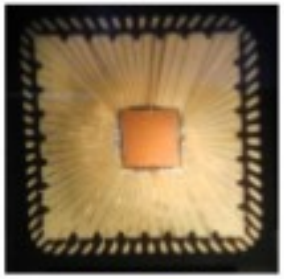
FTK R&D

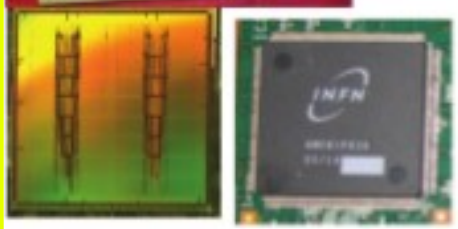


AMchip04 –65nm technology, std cell & full custom, 100MHz
Power/pattern/MHz ~30 times less. Pattern density x12.
First variable resolution implementation!
F. Alberti *et al 2013 JINST* ***8 C01040,*** *doi:*10.1088/1748-0221/8/01/C01040

ASICs: saw on Saturday from Alessandro Marchioro that it's a huge effort to design and make such a thing

AM chip for FTK: AMchip06

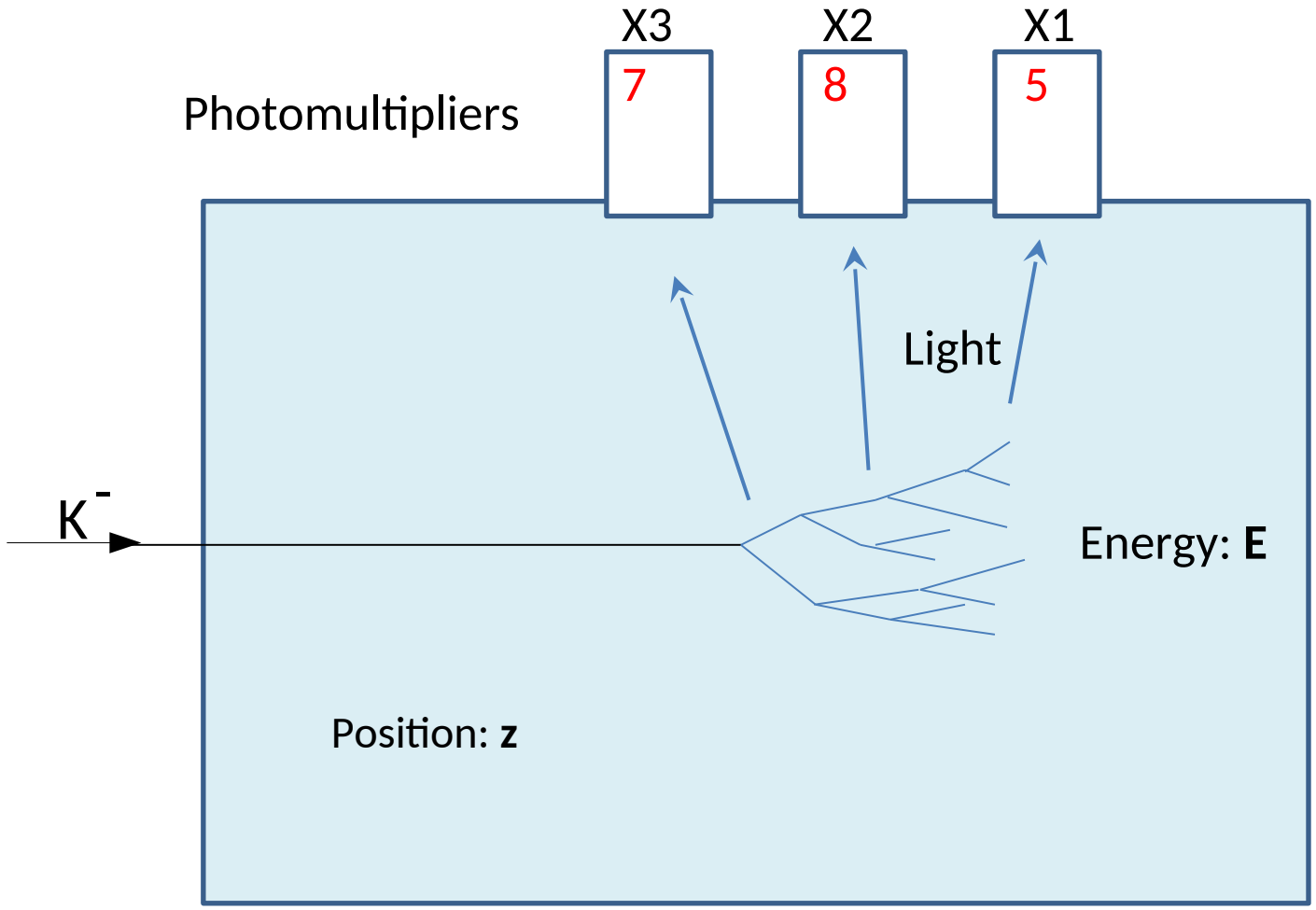




- 90's Full custom VLSI chip - 0.7mm (INFN-Pisa) 128 patterns, 6x12bit words each (F. Morsani et al., The AMchip: a Full-custom MOS VLSI Associative memory for Pattern Recognition, IEEE Trans. on Nucl. Sci.,vol. 39, pp. 795-797, (1992).)
- 1998 FPGA for the same AMchip (P. Giannetti et al. A Programmable Associative Memory for Track Finding, Nucl. Intsr. and Meth., vol. A413/2-3, pp.367-373, (1998)).
- 1999 G. Magazzù, first standard cell project presented at LHCC
- 2006 Standard Cell UMC 0.18 μm 5000 pattern/AMchip for CDF SVT upgrade total: 6M patterns (L. Sartori, A. Annovi et al., A VLSI Processor for Fast Track Finding Based on Content Addressable Memories, IEEE TNS, Vol 53, Issue 4, Part 2, Aug. 2006)
- **2012 AMchip04** 8k patterns in 14mm2, TSMC 65nm LP technology Power/pattern/MHz 40 times less. Pattern density x12. First variable resolution implementation. (F. Alberti et al 2013 JINST 8 C01040, doi:10.1088/1748-0221/8/01/C01040)
- **2013-2014 AMchip MiniAsic and AMchip05**
  a further step towards final AMchip version. Serialized input and output buses at 2 Gbs, further power reduction approach. BGA 23 x 23 package.
- **2014-2015 AMchip06**: final FTK version of the AMchip for the ATLAS experiment .

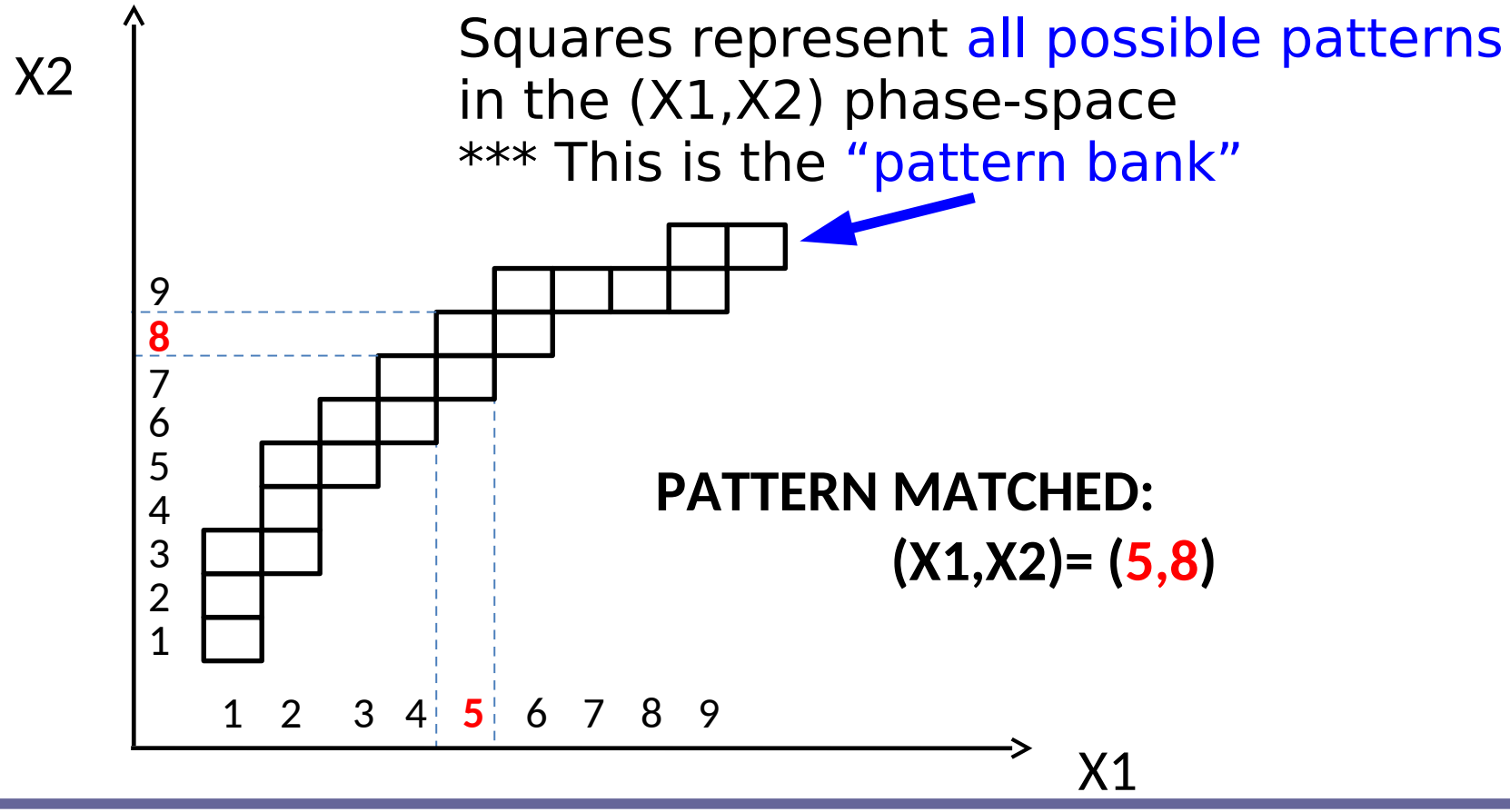
AMchip06: the FTK AM chip has 128k patterns/chip
AMchip08→09: the AM chip for HTT: ~400k pat./chip

Pattern matching not restricted to trackers. Applies everywhere there is correlated behaviour. e.g:



The values in X1 and X2 are correlated: their values define the possible {X1,X2} patterns

For example, task = Associate the measured X1 and X2:
e.g., X1 = 5 with X2 = 8

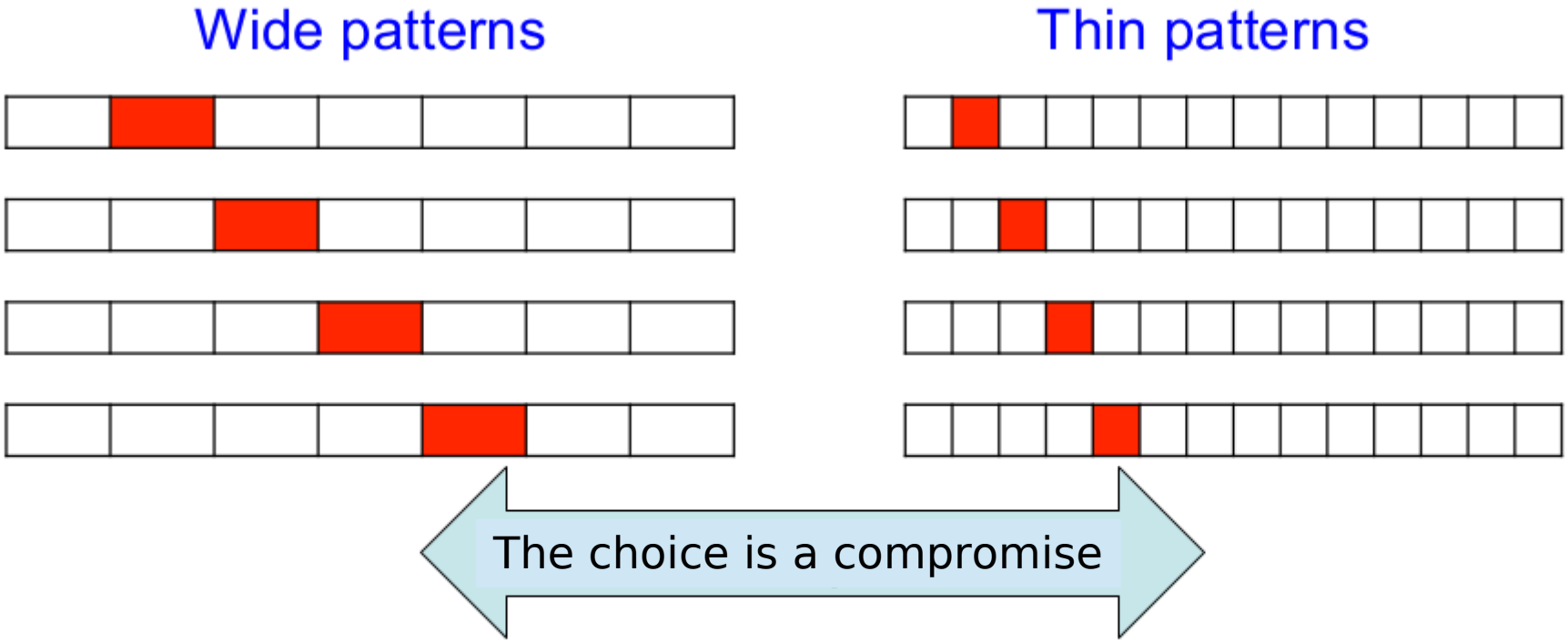


2b.

Now that we have a system that does pattern matching as the data are coming in,

how do we deal with the number of patterns which can be big in the high-granularity detectors?

$N_{patterns}$ depends on the "bin size", i.e, the granularity with which we want to look at the detector



High efficiency
with less patterns (hardware)
BUT more fakes

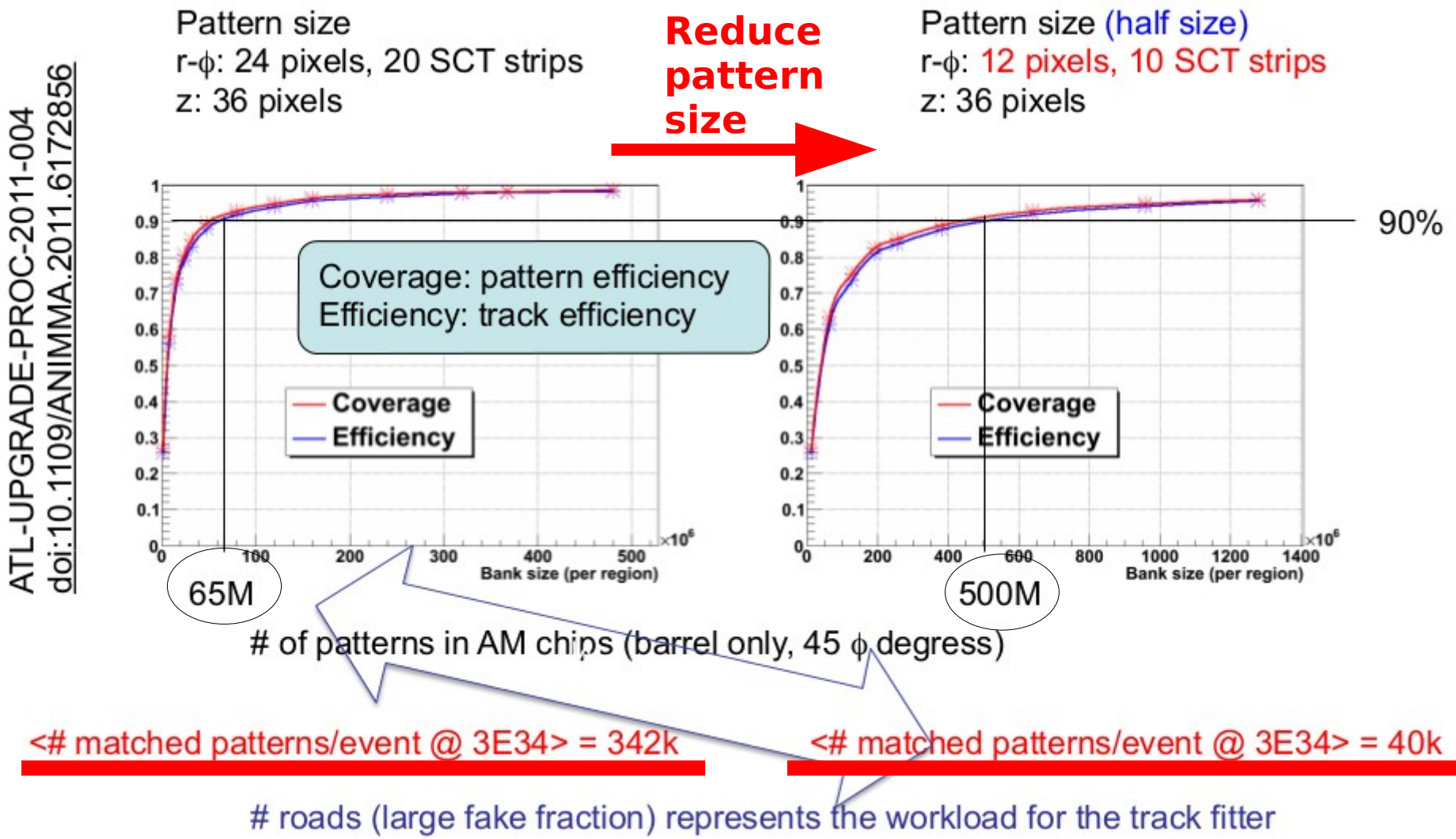
More patterns (hardware)
for same efficiency less fakes
Fakes are workload for track fitter

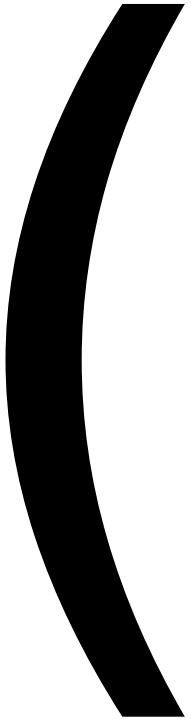
!

Recall: the number of patterns **N_p**, with **m** layers, of **n** bins each, is $N_p \simeq (m-1)n^2$

High efficiency with large bins (small number of patterns) → but, lots of fake tracks found → lots of work for detailed track fitting!

We want "few patterns & few fakes" scenario





Use the feature of "ternary CAMs"

- **Ternary CAM**: added flexibility to the search
  - allows a third matching state of "X" or "Don't Care" for one or more bits in the stored pattern word: one pattern matches various data words
- **Example:** a ternary CAM might have a

  stored word of ----------> "**10XX0**" ("one pattern")

  This will match any of 4 search words: "10000" ("the data")

  "10010" ("the data")

  "10100" ("the data")

  "10110" ("the data")

  The added flexibility comes at additional cost:
  - the internal memory cell must now encode *three possible states instead of the two of binary CAM*. This additional state is typically implemented *by adding a mask bit ("care" or "don't care" bit) to every memory cell*.

Variable resolution (bin sizes) with "Don't Care" (DC) bits

Alberto Annovi
** ANIMMA - A new "Variable Resolution Associative Memory" for High Energy Physics ATL-UPGRADE-PROC-2011-004, doi:10.1109/ANIMMA.2011.6172856*
** "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013*

- For each layer: a "bin" is identified by a number with DC bits (X)
- Least significant bits of "bin" number can use 3 states (0, 1, X)
- The "bin" number is stored in the Associative Memory
- The DC bits can be used to OR neighborhood high-resolution bins, which differ by few bits, without increasing the number of patterns

Pixels:

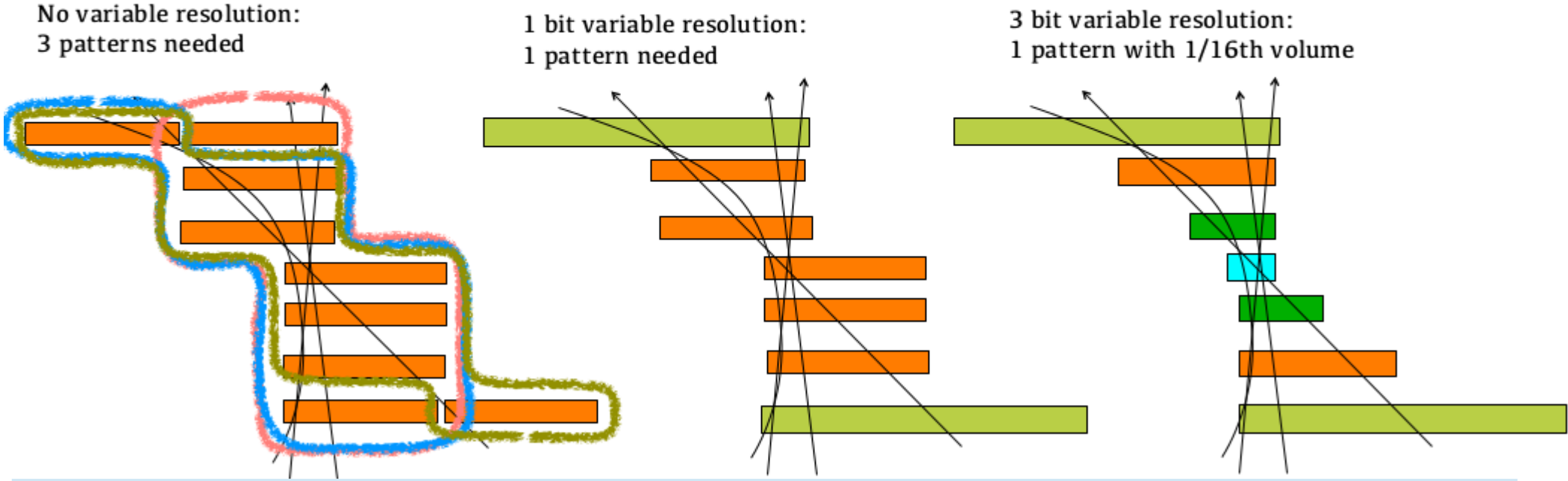
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

Using binary format
"01010" selects bin 10
"0001x" selects bins 2 or 3
"1x000" selects bins 16 or 24
"0x11x" selects bins 6,7,14, or 15
"111xx" selects bins 28 to 31

Refinements: majority & variable widths

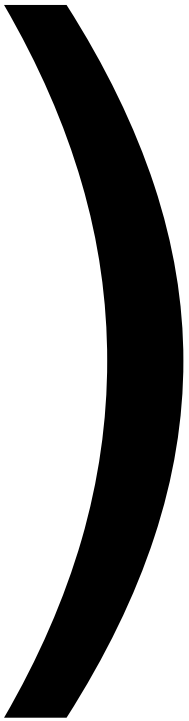
- Majority Logic: Only require N out of M layers have a match
  - Gains efficiency
- Variable Resolution Patterns (Don't Care Bits)

With 2 DC bits: Apart from reduction in fakes (factor 7), we save also a factor 5 in the size of the pattern bank!



Technique can be exploited by any coincidence based trigger!

Alberto Annovi
* *ANIMMA - A new "Variable Resolution Associative Memory" for High Energy Physics ATL-UPGRADE-PROC-2011-004, doi:10.1109/ANIMMA.2011.6172856*
* *"Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013*



3.

So, we have found possible tracks (the matched patterns)

Each matching pattern defines a “road” for the refined tracking

↓

fetch all the (few now) hits in the road

↓

fit them to a helical track to measure the track parameters precisely

Linearized track fitting

E.g., R,φ,z = hit positions of track with curvature ρ, starting at {R=0, φ_0, z_0}

Relations between hit coordinates and track parameters are not linear

e.g., hit coordinates φ, z

$$\phi = \phi_0 - \arcsin\left(\frac{R}{2\rho}\right), \text{ and}$$

$$z = z_0 + 2\rho \arcsin\left(\frac{R}{2\rho}\right) \cot\theta$$

e.g., track parametres

Linearized track fitting

E.g., R,φ,z = hit positions of track with curvature ρ, starting at $\{R=0, \varphi_0, z_0\}$

Relations between hit coordinates and track parameters are not linear

$$\phi = \phi_0 - \arcsin\left(\frac{R}{2\rho}\right), \text{ and}$$

$$z = z_0 + 2\rho \arcsin\left(\frac{R}{2\rho}\right) \cot\theta$$

track parametres p and χ^2 test of fit

hit coordinates x

linear fitter

= linear approximation

- get track parameters p_i as a linear function of hit coordinates x_j

$$p_i = \Sigma_j A_{ij}(x_j - \overline{x_j}) + \overline{p_i}$$

where $\overline{p_i}$ and $\overline{x_j}$ are the track parameters and hit positions about which the linear expansion is performed

- get $\chi_i^2 = \Sigma_i \chi_i^2$, where

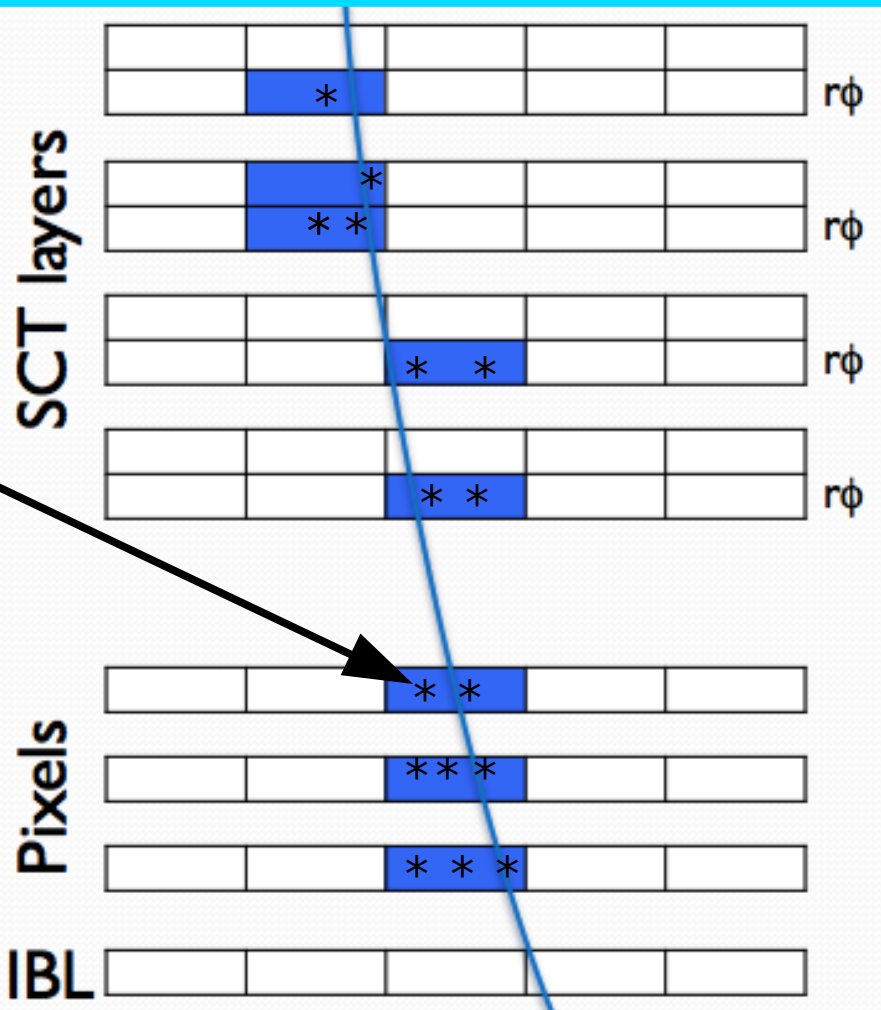
$$\chi_i = \Sigma_j B_{ij}(x_j - \overline{x_j})$$

where the coefficients A_{ij} and B_{ij} are obtained from the PCA.

Linear approximation holds for small detector regions → specific constants for each region → can have many constants

Track fitting in FPGAs: 1st stage, 8 layers

- Pattern recognition layers
- 8 layers track fit
  - full resolution hits
  - reject most fakes



5 parameters & χ^2
d0, z0, eta, phi, PT, χ^2

Full resolution hits

Hit coordinate (local to each detector module)

$$\tilde{p}_i = \sum_{l=1}^{N} C_{il} x_l + q_i$$

Track fitting in FPGAs w/ many Digital Signal Processors (DSPs)

BUT: Linear approximation: get a set of linear equations

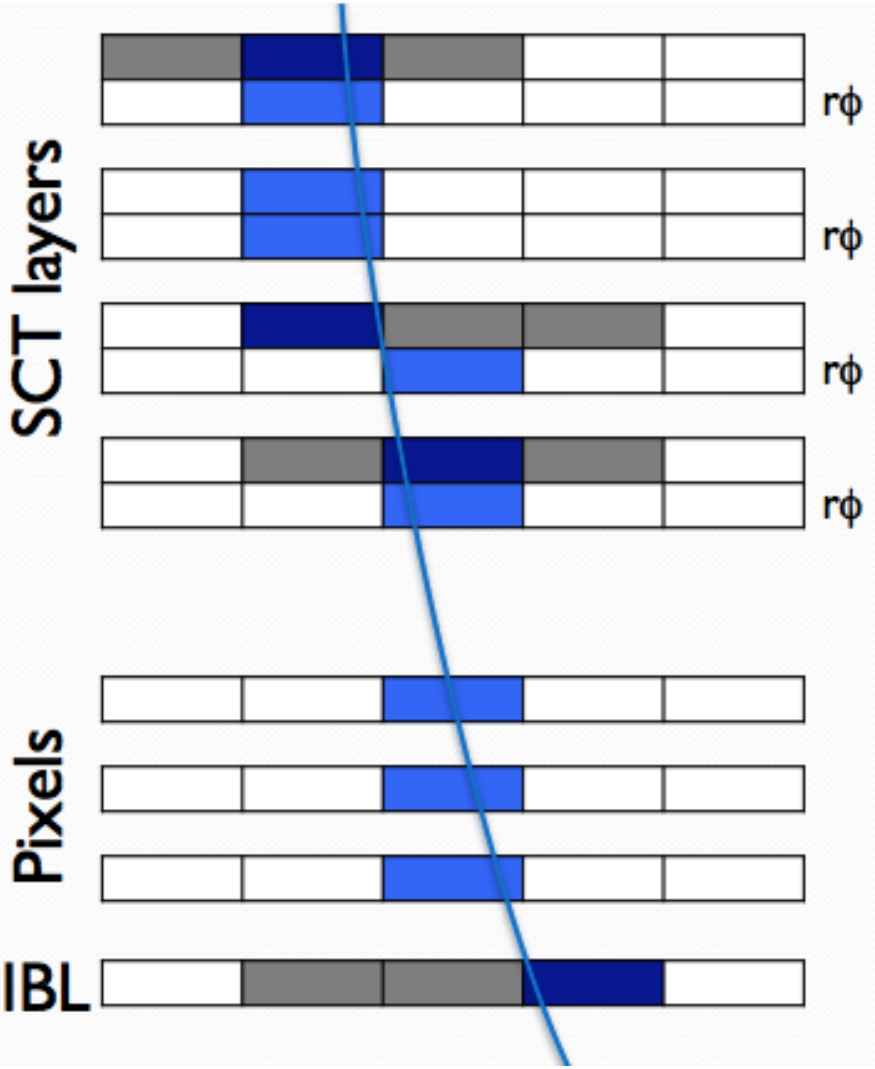
"each parameter depends linearly on the hits" → fast

multiplications with pre-computed constants ~**1 Gfits/s per FPGA**

Limiting factors: # constants in memory & speed of retrieval

Track fitting in FPGAs: 2nd stage (12 layers in FTK, 13 layers in HTT)

- Pattern recognition layers
- 8 layers track fit
  - full resolution hits
  - reject most fakes
- Extrapolate track to other layers
  - Look for hits in a narrow region
- Full 12 layer fit



Done on FPGAs, on a “2nd stage” board

Some documentation for details

FTK Technical Design Report (TDR): https://cds.cern.ch/record/1552953?ln=en
https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/UPGRADE/CERN-LHCC-2013-007/index.html

HTT described (some changes since then) in:
ATLAS Trigger and Data Acquisition Phase-II Upgrade Technical Design Report.
Tech. rep. ATL-COM-DAQ-2017-185. https://cds.cern.ch/record/2296879

FTK Public results: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/FTKPublicResults

A word on strategic decisions. FTK and HTT are not to be used after all:
the luminosity expected for the Run3 of LHC (2023-2025) will not be a factor of 3 higher than Run2, as thought at the time FTK was proposed and designed.
So, this did not make FTK a necessity; can increase the CPU farm to do the job.

For HTT to the HL-LHC era, given the cost of the system, the rapid growth of commercial solutions and (I think the most important after all) the number of experts needed to run it, the tracking will be done in a large CPU farm, with possibly FPGAs and GPUs as co-processors.

The final FTK paper:
The ATLAS collaboration (G. Aad et al.),
"The ATLAS Fast TracKer system", 2021 JINST 16 P07006 (DOI:10.1088/1748-0221/16/07/P07006).

Some Refs: Content Addressable Memory, the Associative Memory & FPGAs

- K. Pagiamtzis and A. Sheikholeslami, *“Content-addressable memory (CAM) circuits and architectures: A tutorial and survey,”* in IEEE Journal of Solid-State Circuits, vol.41, no.3, pp. 712-727, March 2006
- M. Dell'Orso and L. Ristori, "*VLSI Structures Track Finding*", Nucl. Instr. and Meth. A, vol. 278, pp. 436-440, 1989.
- W. Ashmanskas et al., "*The CDF online Silicon Vertex Tracker*", Nucl. Instr. and Meth. A, vol. 485, pp. 178-182, 2002.
- A. Annovi, et al., *“Associative memory design for the Fast TracK processor (FTK) at ATLAS,”* in IEEE NSS/MIC, 2009, Orlando, pp. 1866 – 1867.
- C.-L. Sotiropoulou, S. Gkaitatzis, A. Annovi, et al. *“A Multi-Core FPGA-based 2D-Clustering Implementation for Real-Time Image Processing”*, in IEEE Trans. on Nuclear Science, vol. 61, no. 6, pp. 3599 - 3606, December 2014.

The point to take home:

- Split the problem in a fast (coarse) one, and a refined one working with much reduced data (we do this all the time in the trigger)
- Use pre-calculated patterns & values wherever you can: if you get the desired precision, you gain a lot in time

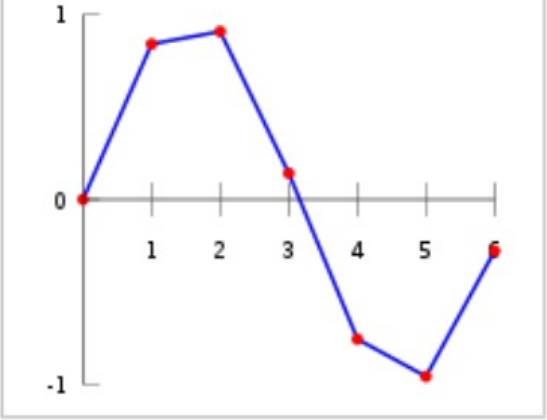
  ...And time is precious in the online world!

- We saw the example of the Fast TracKer (FTK) and the Hardware Tracking for the Trigger (HTT) upgrades in ATLAS
  - **AM-based pattern matching** with "AM chip" (ASIC),
  - **refined track-fitting** + **almost everything else** needed (from formatting to smart databases to I/O) in powerful modern FPGAs (recall Hannes Sakulin's & Mauricio Feo's talks)

Approximations and Tables with constants are very common calculation tools. Don't be afraid of them!

- Constants can be coefficients in Taylor expansions, Fourier series, etc. e.g.,
  - sin(x) = Taylor expansion gives a polynomial to calculate $\sin(x) \simeq x - x^3/6 + x^5/120$
- Or, use Look-Up Tables (LUTs = pre-calculated values stored in tables) → interpolate between stored values to get value of sin(x) you ask for

```
function lookup_sine(x)
    x1 := floor(x*1000/pi)
    y1 := sine_table[x1]
    y2 := sine_table[x1+1]
    return y1 + (y2-y1)*(x*1000/pi-x1)
```



Linear interpolation on a portion of the sine function

Some improvements on this method of tracking

- **Pattern matching:** Towards AM09: the 400k patterns per chip:
  - "The AM08 Associative Memory ASIC Design, Architecture and Evaluation methodology",
    A. Vgenopoulos et al., published as procceedings in MOCAST2022

- **Track Fitting:** We saw that we can make the Track Fitting very fast with linearized equations (track parameters expressed as linear functions of hit positions). But, need small regions for the linear-equations-approximation to be true → many sets of equation parameters (one set per region).
  - Can make a coordinate transformation to an "idealized geometry" and there the detector regions where the linear approximation holds are much larger: **much fewer regions → fewer equations → fewer sets of parameters → much smaller memory needs on the FPGAs:**
  - "*A High-performance Track Fitter for Use in Ultra-fast Electronics*", E. Clementa et al., 2018, arXiv:1809.01467

Linearized track fitting revisited (1/2)

linear fitter

- get track parameters p_i as a linear function of hit coordinates x_j

$$p_i = \Sigma_j A_{ij}(x_j - \overline{x_j}) + \overline{p_i}$$

where $\overline{p_i}$ and $\overline{x_j}$ are the track parameters and hit positions about which the linear expansion is performed

Linear approximation holds for small detector regions → specific constants for each region → many constants

- get $\chi_i^2 = \Sigma_i \chi_i^2$, where

$$\chi_i = \Sigma_j B_{ij}(x_j - \overline{x_j})$$

where the coefficients A_{ij} and B_{ij} are obtained from the PCA.

R,φ,z = hit positions of track with curvature ρ, starting at {R=0, φ_0, z_0}

$$\phi = \phi_0 - \arcsin\left(\frac{R}{2\rho}\right), \text{ and}$$

$$z = z_0 + 2\rho \arcsin\left(\frac{R}{2\rho}\right)\cot\theta$$

Transform the hit positions R,φ, z → R', φ', z' (ΔR=R-R', c= +/- 1), so that the detector looks linear to the linear fitter:

First order

$$\phi' = \phi + \Delta R\frac{c}{2\rho} + \frac{1}{6}\left(R\frac{c}{2\rho}\right)^3 + \Delta\phi_{strip}$$

$$z' = z - \Delta R\cot\theta + \frac{1}{6}\cot\theta\left(\frac{c}{2\rho}\right)^2 R^3$$

Strip corrections

Second order (important at low p_T)

R. Eusebi, "*A High-performance Track Fitter for use in Ultra-fast Electronics*", 2020 talk based on the original work of E. Clementa et al. https://arxiv.org/pdf/1809.01467.pdf

Linearized track fitting revisited (2/2)

- "*A High-performance Track Fitter for Use in Ultra-fast Electronics*", E. Clementa et al., 2018, arXiv:1809.01467

First order

$$\phi' = \phi + \Delta R \frac{c}{2\rho} + \frac{1}{6}\left(R\frac{c}{2\rho}\right)^3 + \Delta\phi_{strip}$$

$$z' = z - \Delta R \cot\theta - \frac{1}{6}\cot\theta\left(\frac{c}{2\rho}\right)^2 R^3$$

Strip corrections

Second order (important at low p_T)

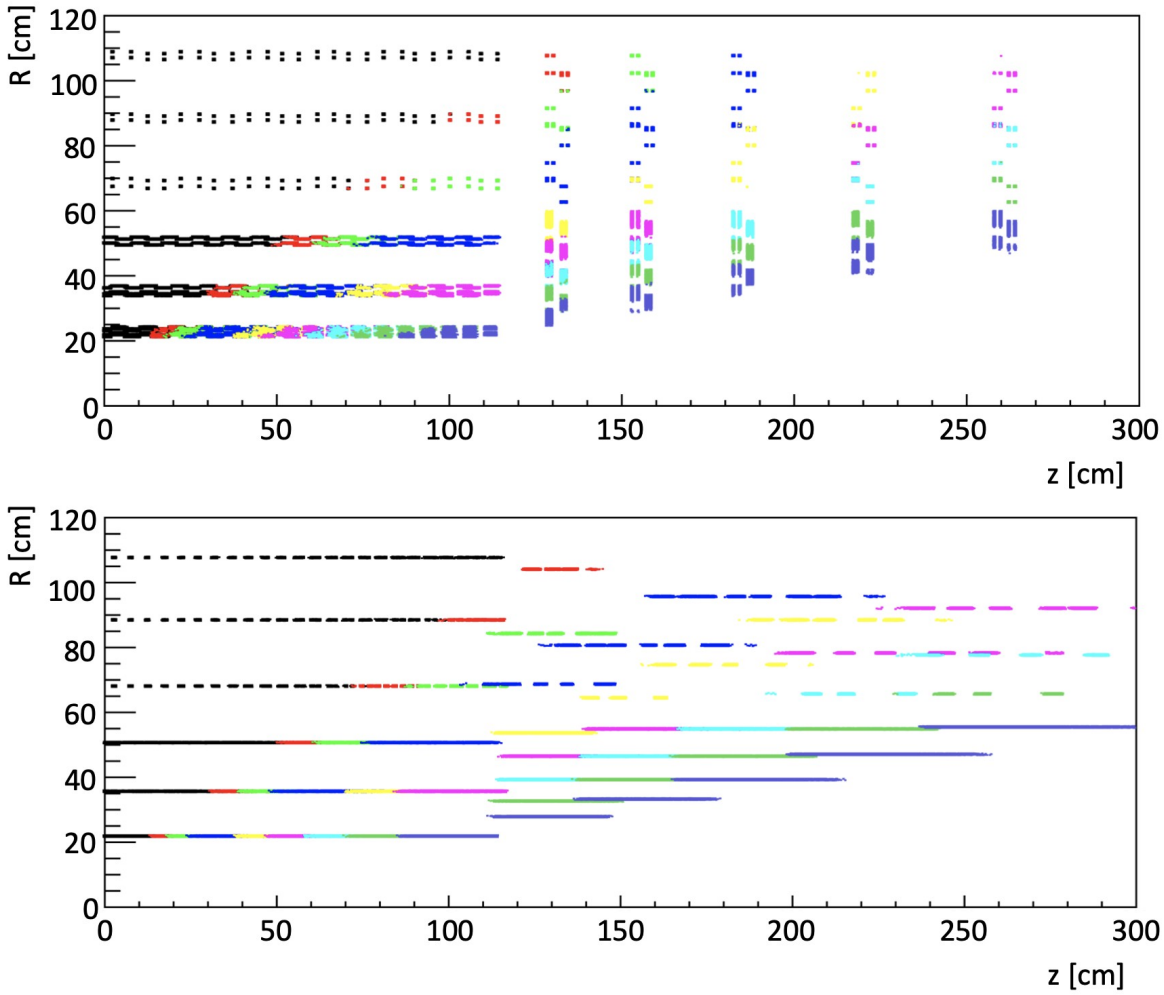


Figure 3: Hit positions in the (r, z) plane. The colors represent hits assigned to the different regions. Top: original positions. Bottom: transformed hit coordinates.

C. Other examples for track finding & track fitting

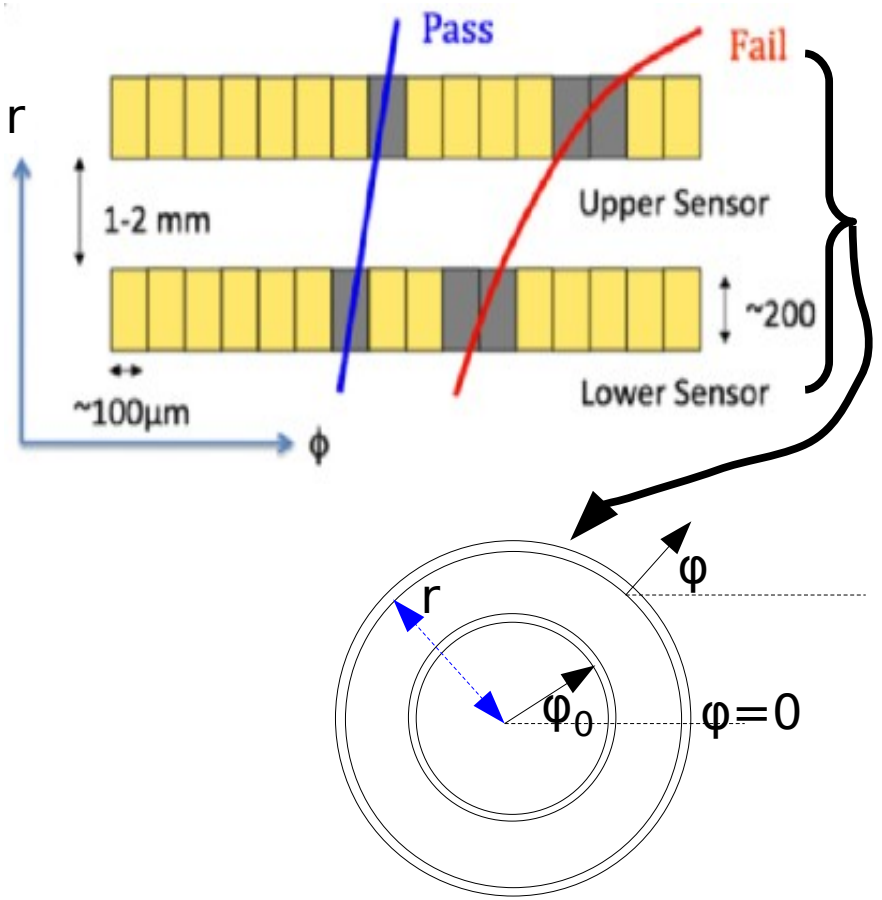
Other examples on ways to do the tracking

- What was presented here is not the only way to solve the tracking problem fast. Other solutions exist, e.g:
  - Hough transform in FPGAs,
  - Other algorithms in FPGAs (e.g., Retina algorithm: Luciano Ristori, NIM A 453 (2000) pp. 425-429)
  - Solutions implemented on GPUs (→ You heard from Gianluca Lamanna)
  - Machine Learning for TDAQ in GPUs and FPGAs (→ You heard from Satchit Chatterji & Thomas Owen James)
- Nothing can be as fast as doing the tracking while reading your data, as they pass through the system.
  - You an even build a specific ASIC to implement what you can do in an FPGA
  - Commercial solutions (e.g., CPUs, FPGAs, GPUs, etc.) can overcome slower speed with high parallelism

  → it's all a matter of cost at the end...

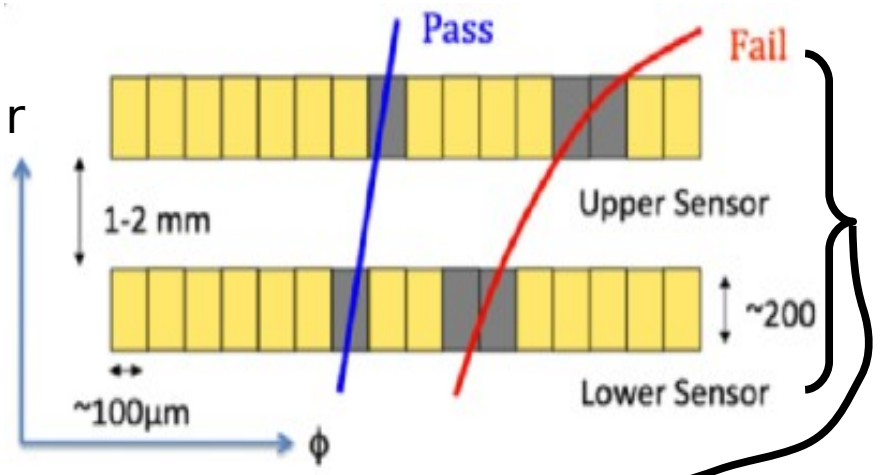
L1 Track Trigger at CMS : starts with a much cleaner picture than ATLAS (CMS: stubs, ATLAS:hits)

- ~99% of tracks have $p_T < 2$ GeV/c ; interesting things have higher p_T tracks.
- CMS makes the detector itself selective on such tracks by finding track “stubs” on closely spaced layers (~15,000 stubs in total, every 25 ns)

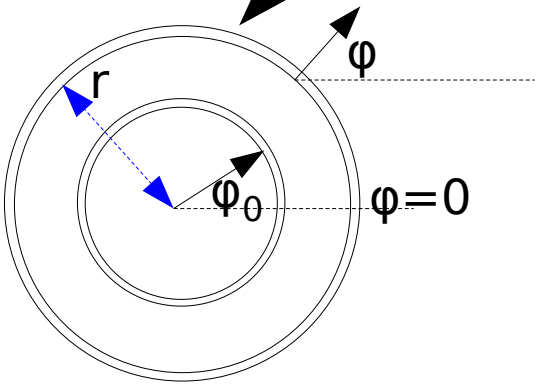


L1 Track Trigger at CMS : starts with a much cleaner picture than ATLAS (CMS: stubs, ATLAS:hits)

- ~99% of tracks have $p_T < 2$ GeV/c ; interesting things have higher p_T tracks.
- CMS makes the detector itself selective on such tracks by finding track "stubs" on closely spaced layers (~15,000 stubs in total, every 25 ns)



- From a track generated with φ_0 and p_T, in B=4 Tesla, you have stubs (straight lines) of angle φ at double layers at radius r.



$$Radius\ of\ curvature = R_{curv} = \frac{p_T}{0.3\,B\,q}, q = \pm 1$$

$$\sin(\varphi - \varphi_0) = \frac{r}{2R_{curv}} \rightarrow \varphi - \varphi_0 \simeq \frac{r}{2R_{curv}}$$

$$\varphi = \frac{\pm 0.006}{p_T} r + \varphi_0$$

φ = u r + v

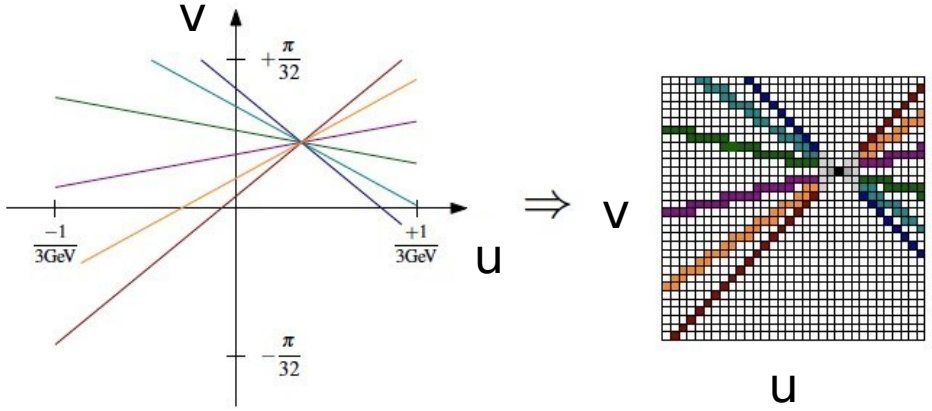
Can do "Hough Transform" for track finding (1/3)

- This φ(r) behaviour is a straight line: **φ = u r + v**

  → Could do a **"Hough transform"**: map individual {r, φ} measurement points to a whole-line in 2D space; 2D = **the slope (u) & the intercept (v)**

  → Given an {**r,φ**} pair; for each **u**, get **v**: **v = -u r + φ**

  and put {u,v} in a 2-dimensional histogram



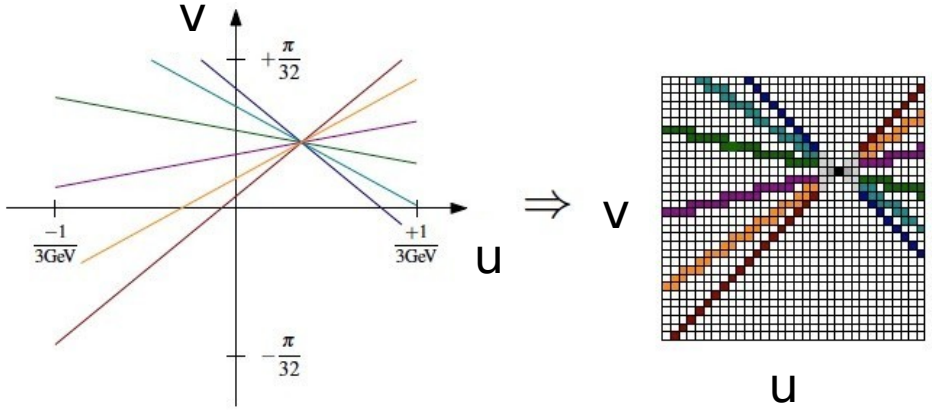
Can do "Hough Transform" for track finding (1/3)

- This φ(r) behaviour is a straight line: **φ = u r + v**

  → Could do a **"Hough transform"**: map individual {r, φ} measurement points to a whole-line in 2D space; 2D = **the slope (u) & the intercept (v)**

  → Given an {**r,φ**} pair; for each **u**, get **v**: **v = -u r + φ**

  and put {u,v} in a 2-dimensional histogram



- **{r,φ} measurements from same track will populate same {u,v} bin**
- **Most populated bin = characterises whole track**
- Note: Small |u| values:
  $|u| = 0.006/P_T \rightarrow |u| < 0.003$

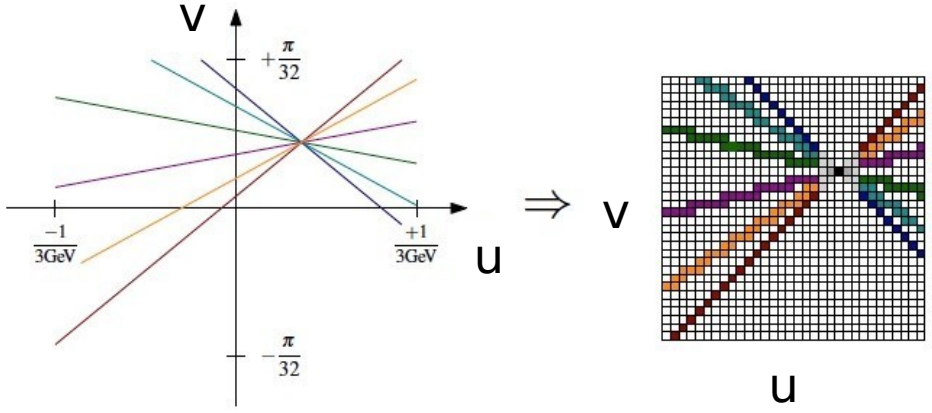
Can do “Hough Transform” for track finding (2/3)

- This φ(r) behaviour is a straight line: **φ = u r + v**

  → Could do a **“Hough transform”**: map individual {r, φ} measurement points to a whole-line characteristic in 2D space; 2D = **the slope (u) & the intercept (v)**

  → Given an {r,φ} pair; for each m, get c: **v = -u r + φ**

  and put {u,v} in a 2-dimensional histogram



- **{r,φ} measurements from same track will populate same {u,v} bin**
- **Most populated bin = characterises whole track**
- Note: Small |u| values:
  $|u| = 0.006/P_T \rightarrow |u| < 0.003$

Pileup events: {u,v} array heavily populated and such peaks are not initially prominent.

But, by requiring e.g., all stubs in the (u,v) histogram bin to be from different radial layers, significantly reduces the background

Hough transform (3/3)

- P. V. C. Hough, i) *"Machine Analysis of Bubble Chamber Pictures"*, 2nd International Conference on High-Energy Accelerators and Instrumentation, HEACC 1959 : CERN, Geneva, Switzerland, September 14-19, 1959, 554-558. Published in: Conf.Proc.C 590914 (1959) 554-558, ii) *"Method and means for recognizing complex patterns,"* U.S. Patent 3,069,654, 1962.
- R. O. Duda and P. E. Hart, *"Use of the Hough transformation to detect lines and curves in pictures"* Communications of the ACM, vol. 15, no. 1, pp. 11–15, 1972.
- J. Illingworth and J. Kittler, *"The Adaptive Hough Transform"*, IEEE Trans. On Pattern Analysis and Machine Inteligence, Vol PAMI-9, No. 5, Sept. 1987, pp. 690-698.
- L. Voudouris, S. Nikolaidis, A. Rjoub, *"High Speed FPGA implementation of hough transform for real-time applications"* , 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, Tallinn, Estonia, 2012, pp. 213-218, doi: 10.1109/DDECS.2012.6219060

  ....etc...

- On FPGAs: **important to adapt the algorithms to the constraints of FPGA operation**. Algorithms can overflow the capacity of even a very large FPGA because of timing constraints or routing congestion, as you learned in the lectures

Artificial Retina Algorithm (1/5)

a) Luciano Ristori, NIM A 453 (2000) pp. 425-429. b) R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016)

Inspired by the quick detection of edges in the visual cortex of mammals :

Specific neurons, called receptive fields, receive signals only from specific regions of the retina, in order to reduce the connectivity and save bandwidth.

The neurons are tuned to recognize a specific shape and the response is proportional to how close are the stimulus shape and the shape for which the neuron is tuned to. Generated in parallel, the responses of neurons are then interpolated to create a preview of image edges

Artificial Retina Algorithm (2/5)

a) Luciano Ristori, NIM A 453 (2000) pp. 425-429. b) R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016)

Inspired by the quick detection of edges in the visual cortex of mammals : Specific neurons, called receptive fields, receive signals only from specific regions of the retina, *in order to reduce the connectivity and save bandwidth. The neurons are tuned to recognize a specific shape and the response is proportional to how close are the stimulus shape and the shape for which the neuron is tuned to. Generated in parallel, the responses of neurons are then interpolated to create a preview of image edges*

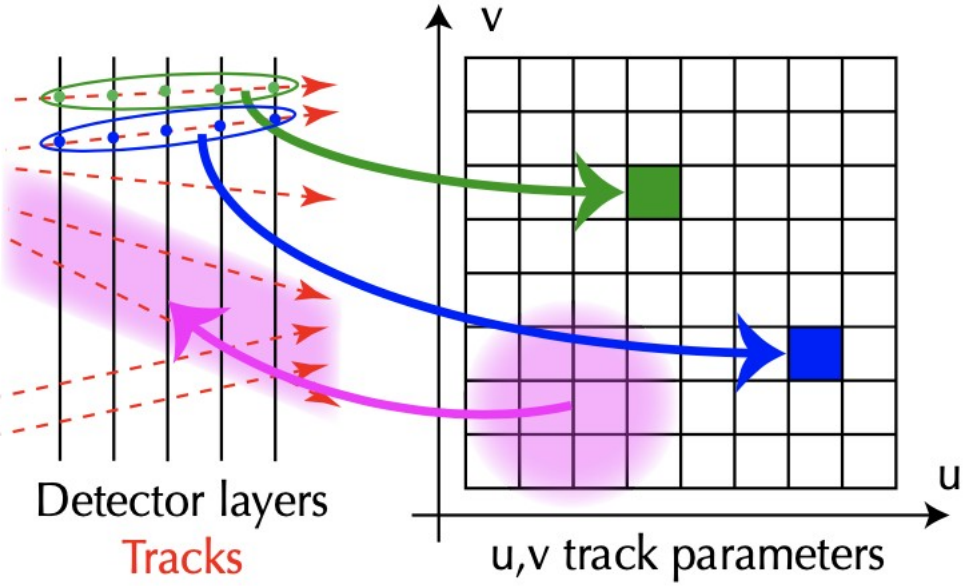
- Algorithm has mathematical similarities with"Hough transform"

First, map: *template tracks ↔ cells in the track-parameter space*

"template track hits

= receptors"

"cells = neurons"



Artificial Retina Algorithm (3/5)

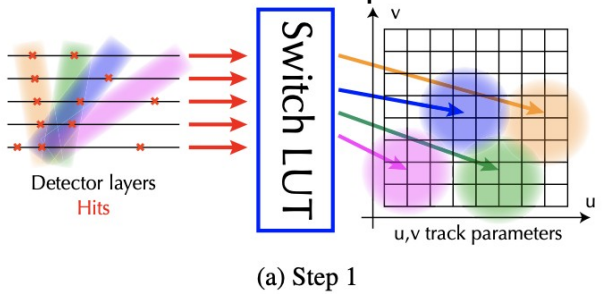
a) Luciano Ristori, NIM A 453 (2000) pp. 425-429. b) R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016)

Two levels of parallelization:

Each cell processes in parallel hits from a limited detector region: small bandwidth per cell

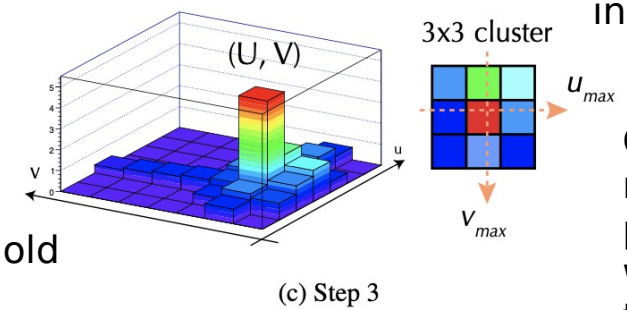
If events have time stamps, events can be processed simultaneously.

For each cell, each hit adds a weight, calculated according to distance of the actual hit from the "receptor hit" for this cell at the same layer



Track → 1 cell
Road → cell cluster

Each (u,v) cell shows up in all n detector layers
sumW = Σw_n



sumW > threshold

Centroid measures track parameters (u,v) with better precision than bin size

Figure 2: Processing steps of track reconstruction with the artificial retina algorithm.

Artificial Retina Algorithm (3/5)

a) Luciano Ristori, NIM A 453 (2000) pp. 425-429. b) R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016)

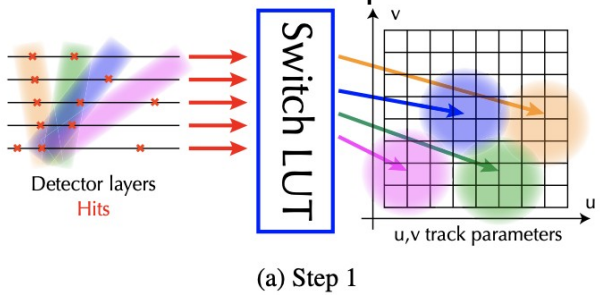
Inspired by the quick detection of edges in the visual cortex of mammals : Specific neurons, called receptive fields, receive signals only from specific regions of the retina, in order to reduce the connectivity and save bandwidth. The neurons are tuned to recognize a specific shape and the response is proportional to how close are the stimulus shape and the shape for which the neuron is tuned to. Generated in parallel, the responses of neurons are then interpolated to create a preview of image edges

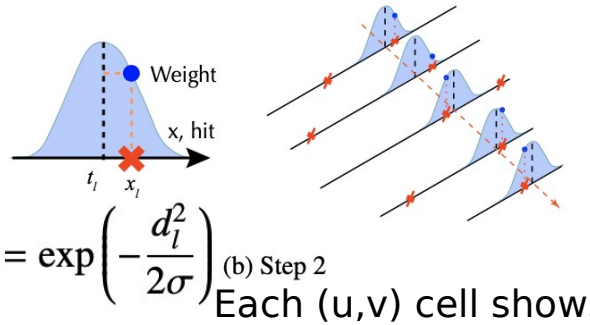
Thanks to the weight,

the contribution from hits which are far from the mapped track is negligible

and Retina is able to find the track even in the presence of noise hits or hits from additional tracks.

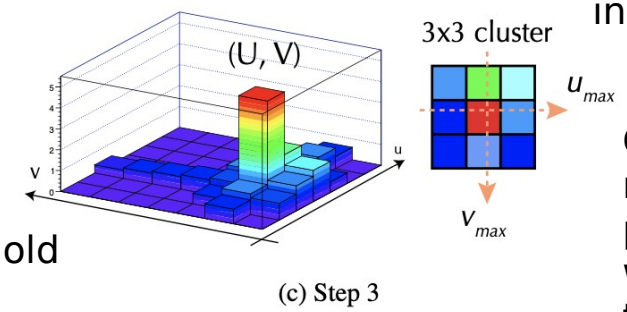
For each cell, each hit adds a weight, calculated according to distance of the actual hit from the "receptor hit" for this cell at the same layer





Track → 1 cell
Road → cell cluster

Each (u,v) cell shows up in all n detector layers
sumW = Σw_n



sumW > threshold

Centroid measures track parameters (u,v) with better precision than bin size

Figure 2: Processing steps of track reconstruction with the artificial retina algorithm.

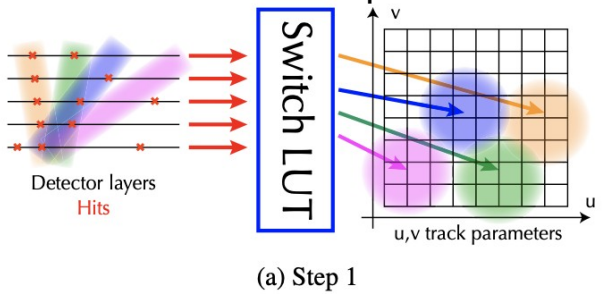
Artificial Retina Algorithm (4/5)

a) Luciano Ristori, NIM A 453 (2000) pp. 425-429. b) R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016)

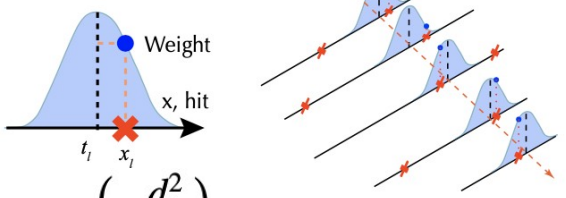
Inspired by the quick detection of edges in the visual cortex of mammals : Specific neurons, called receptive fields, receive signals only from specific regions of the retina, in order to reduce the connectivity and save bandwidth. The neurons are tuned to recognize a specific shape and the response is proportional to how close are the stimulus shape and the shape for which the neuron is tuned to. Generated in parallel, the responses of neurons are then interpolated to create a preview of image edges

Big bandwidth reduction at the output: send out centroids centered at cells with sumW > threshold
(a compromise between
track finding efficiency and # fake tracks , like always)

For each cell, each hit adds a weight, calculated according to distance of the actual hit from the "receptor hit" for this cell at the same layer



(a) Step 1

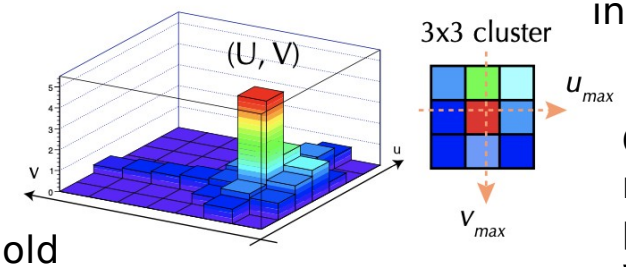


$$w = \exp\left(-\frac{d_l^2}{2\sigma}\right)$$

(b) Step 2

Track → 1 cell
Road → cell cluster

Each (u,v) cell shows up in all n detector layers
$sumW = \Sigma w_n$



sumW > threshold

(c) Step 3

Centroid measures track parameters (u,v) with better precision than bin size

Figure 2: Processing steps of track reconstruction with the artificial retina algorithm.

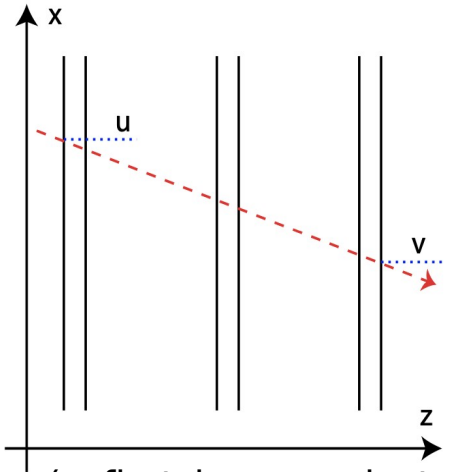
Artificial Retina Algorithm (5/5)

Luciano Ristori, An artificial retina for fast track finding, Nucl. Instrum. Meth. A 453 (2000) 425-429

R. Censi et al., First Results of an "Artificial Retina" Processor Prototype, EPJ Web of Conferences 127, 00005 (2016), Connecting the Dots 2016.

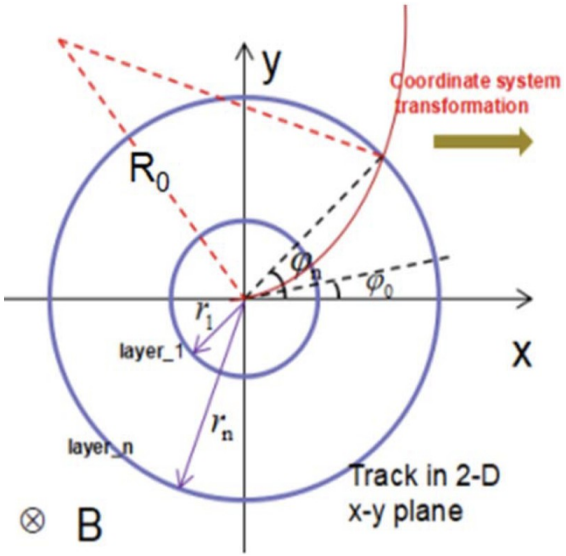
W. Deng et al., Iterative Retina for High Track Multiplicity in a Barrel-Shaped Tracker and High Magnetic Field, IEEE TransNuclScience Vol. 68, No 8, Aug 2021

Retina (like Hough) can be applied
to cylindrical geometries with magnetic field,
to planar geometries without magnetic field, etc.
Each time define (u,v) accordingly

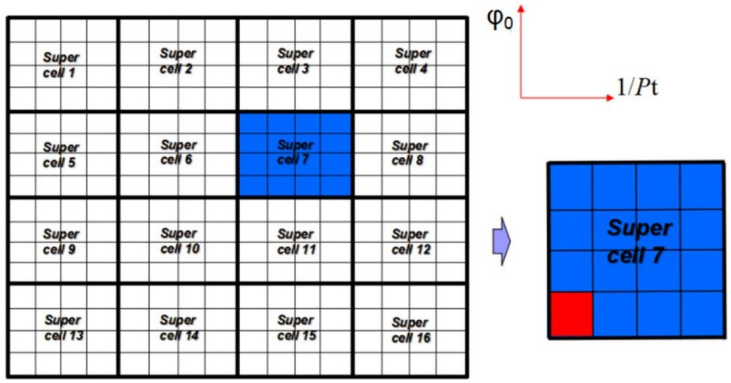


(u,v) = (x_first_layer , x_last_layer) ⊗ B

(e.g., VELO at LHCb)



Can also make an "iterative scan".
E.g., have a 4x4 grid of super-cells, find the super-cell with the candidate track and scan finer 4x4 inside. Thus, do
4x4 and then 4x4 = 2 * 4x4 = 32 scans,
instead of 16x16 = 256 for same resolution



“Global” and “local” tracking

- What we’ve seen so far is “global tracking”: all hits available simultaneously (pattern matching and linear approximation wanted all hits present to work with the patterns and constants needed).
- “Local tracking” (~progressive tracking): add hits on the way
- **Track finding at CMS for L1 tracking**:
  - stubs in adjacent layers form *“tracklet seeds”* → **a)** growth of tracks by projection to next layers and χ^2 test for adding a new stub, **b)** update track parameters, **c)** extrapolate further to include more stubs and so on.
- E.g, see (and references therein): T. James, “*Level-1 Track Finding with an all-FPGA system at CMS for the HL-LHC*”, arXiv:1910.12668

  A. Hart, “*Level 1 Track Finder at CMS*” arXiv:1910.06614

  E.Bart, *FPGA-based tracking for the CMS Level-1 trigger using the tracklet algorithm*, arxiv:1919.09970 JINST 15 P06024 (2020)

“Local” tracking

- “Local tracking” (~progressive tracking): add hits on the way
- **Track finding at CMS for L1 tracking**:
  - stubs in adjacent layers form *“tracklet seeds”* → **a)** growth of tracks by projection to next layers and χ^2 test for adding a new stub, **b)** update track parameters, **c)** extrapolate further to include more stubs and so on.
- **Track fitting at CMS for L1 tracking:**
  - “*Kalmam filter*” → project the track parameters of the tracklet to each next layer, recalculate hit positions based on extrapolation and observed hits, recalculate and extrapolate track parameters and so on.
- E.g,
- T. James, “A hardware track-trigger for CMS at the high lumi- nosity LHC,” Ph.D. dissertation, Dept. Phys., Imperial College London, London, U.K., Feb. 2018.
- From: W. Deng et al., Iterative Retina for High Track Multiplicity in a Barrel-Shaped Tracker and High Magnetic Field, IEEE TransNuclScience Vol. 68, No 8, Aug 2021

Kalman filter (1/4)

- Typically, *need refinement after track finding*. (e.g., in both Hough and Retina, the finite cell size in 1/Pt means bad Pt resolution for high Pt).We saw the linearized track-fitting approach of FTK/HTT in FPGAs. Here, see *Kalman fitler* (R. Frühwirth, "Application of Kalman filtering to track fitting in the DELPHI detector", Tech. Rep. CERN-DELPHI-87-23-PROG-70, 1987).
- Kalman returns track parameters (1/Pt, φ_0, cotθ, and z_0), at the end of an iterative process. Track parameters from track finding (e.g., 1/Pt, φ_0) serve as initial "state vector" (e.g., x_0 = {1/Pt, φ_0, cotθ=1, and z_0=0}

Kalman filter (1/4)

- Typically, *need refinement after track finding*. (e.g., in both Hough and Retina, the finite cell size in 1/Pt means bad Pt resolution for high Pt).We saw the linearized track-fitting approach of FTK/HTT in FPGAs. Here, see *Kalman fitler* (R. Frühwirth, "Application of Kalman filtering to track fitting in the DELPHI detector", Tech. Rep. CERN-DELPHI-87-23-PROG-70, 1987).
- Kalman returns track parameters (1/Pt, φ_0, cotθ, and z_0), at the end of an iterative process. Track parameters from track finding (e.g., 1/Pt, φ_0) serve as initial "state vector" (e.g., x_0 = {1/Pt, φ_0, cotθ=1, and z_0=0}
  - Propagation of the state vector from one detection layer "t-1" to the next detection layer "t": $x_{t \mid t-1}$
    - Propagation is subject to error due to multiple scattering and energy losses. State vector x_t comes with its covariance matrix P_t
  - Update the state vector and covariance matrix according to the measurements (the detector hits) on this layer and their uncertainties.
  - Continue with next layer, and so on.
    - Best track at end of iteration.

Kalman filter (2/4) – Predictions

- Propagation of the state vector from detection layer "t-1" to the next detection layer "t": $x_{t\,|\,t\text{-}1}$
  - Propagation is subject to error due to multiple scattering and energy losses

1

Predict state vector at t from info till t-1

State vector calculated with info till t-1

$$\hat{x}_{t|t-1} = F_{t-1}\,\hat{x}_{t-1|t-1}$$

Propagation matrix

Predict covariance at t with info till t-1

$$P_{t|t-1} = Q_{t-1} + F_{t-1} P_{t-1|t-1} F_{t-1}^{T}$$

Noise propagation

Covariance matrix :

$$P_{t|t} = \begin{bmatrix} \sigma_a^2 & \sigma_{ab}0 & 0 & \\ \sigma_{ab} & \sigma_b^2 & 0 & 0 \\ 0 & 0 & \sigma_c^2 & \sigma_{cd} \\ 0 & 0 & \sigma_{cd} & \sigma_d^2 \end{bmatrix}$$

{a,b,c,d} = {1/Pt, φ_0, cotθ, z_0}

Initial covariance:

$$\sigma_{ab} = \sigma_{cd} = 0$$

$$\sigma_a, \sigma_b = 1/\sqrt{12}\ of\ Hough/Retina\ cell\ width$$

$$\sigma_c = 1/\sqrt{12}\ of\ \Delta\eta\ region$$

$$\sigma_d = RMS(z_0) \approx 4.5\,cm\ for\ ATLAS/CMS$$

From: W. Deng et al., Iterative Retina for High Track Multiplicity in a Barrel-Shaped Tracker and High Magnetic Field, IEEE TransNuclScience Vol. 68, No 8, Aug 2021

Kalman filter (3/4) - Updates

- Update of the state vector and its covariance with the inclusion of measurements at detection layer "t"

2

Estimate measurements at t based on predicted state vector at t

$$\hat{m}_t = H_t \hat{x}_{t|t-1}$$

Projector matrix

Actual measurement vector m_t, with associated covariance R_t

Measurement error matrices : R_t

$$= \begin{bmatrix} \sigma_\varphi^2 & 2 \\ 2 & \sigma_z^2 \end{bmatrix},$$

Strip pitch p

strip length

$$\sigma_\varphi = \frac{p/\sqrt{12}}{r} \qquad \sigma_z = \frac{l}{\sqrt{12}}$$

Pre-fit residuals of measurements at t: $m_t - \hat{m}_t = m_t - H_t \hat{x}_{t|t-1}$

Kalman gain matrix → describes improvement in precision of state vector from the extrapolation

$$W_t = P_{t+1|t} H_t^T \left(H_t P_{t+1|t} H_t^T + R_t \right)^{-1}$$

3

Update state vector, using the measurements m_t $\quad x_{t|t} = \hat{x}_{t|t-1} + W_t \left(m_t - \hat{m}_t \right)$

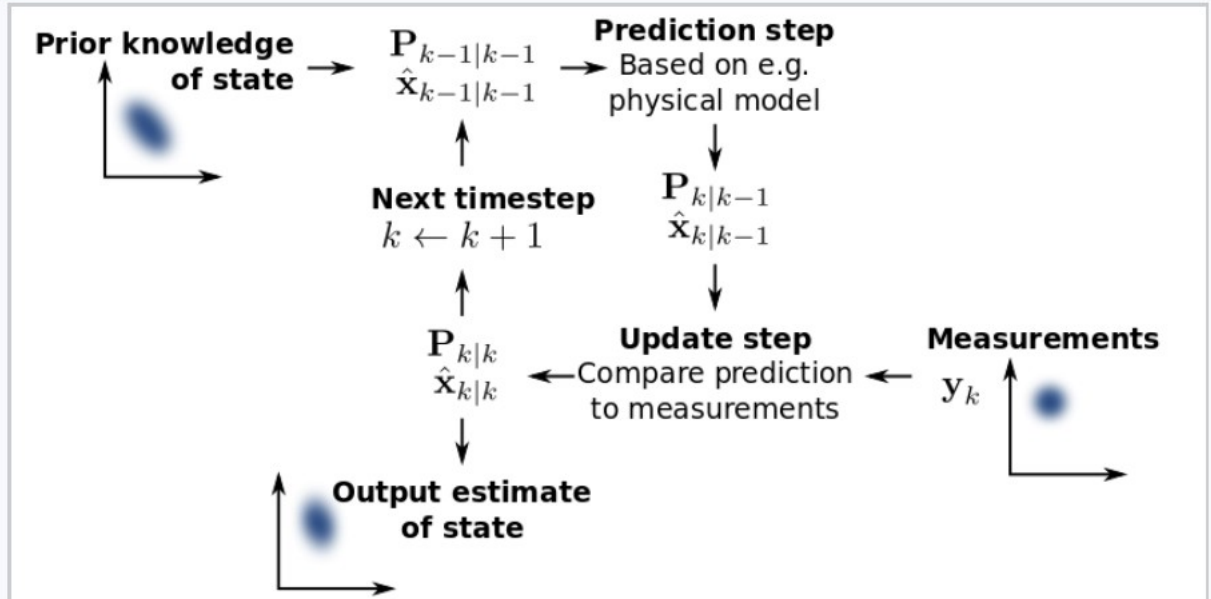
Update covariance, using the measurements m_t $\quad P_{t|t} = \left(I - W_t H_t \right) \hat{P}_{t|t-1}$

Post-fit residuals of measurements at t: $m_t - H_t x_{t|t}$

Kalman gain should minimize the residual

Kalman Filter (4/4)

- From wikipedia https://en.wikipedia.org/wiki/Kalman_filter



The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements. $\hat{x}_{k|k-1}$ denotes the estimate of the system's state at time step *k* before the *k*-th measurement y_k has been taken into account; $P_{k|k-1}$ is the corresponding uncertainty.

Kalman filtering is used in many applications. The “next level” could be the “next time step”, for example.

D. Beyond High Energy Physics applications: E.g., image processing with pattern matching

Beyond High Energy Physics applications: E.g., image processing with pattern matching

- Hough transform is one of the classic techniques used in generic image processing to do first an “edge-detection”
- Let's see another interesting example on image perception:

  M. Del Viva, G. Punzi, and D. Benedetti. *“Information and perception of meaningful patterns.”* PloS one 8.7 (2013): e69154.

  “… models describe the initial processing of visual information as the extraction of a simplified “sketch” based on a limited number of “salient features” [11], [12], that therefore contains a much reduced amount of information.”

Beyond High Energy Physics applications: E.g., image processing with pattern matching

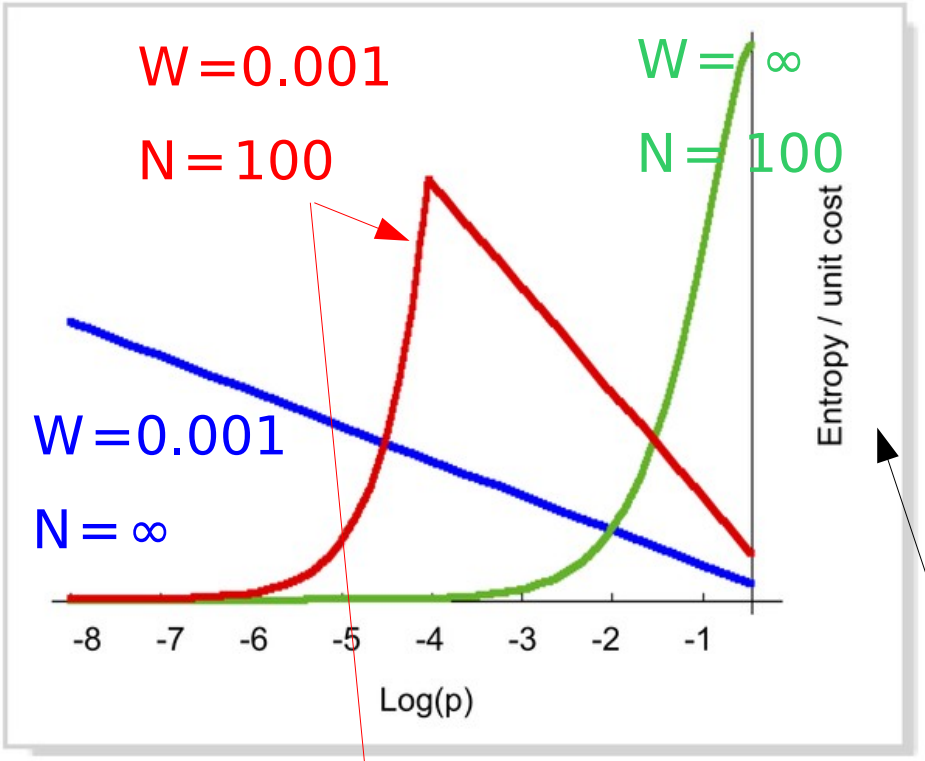
- Hough transform is one of the classic techniques used in generic image processing to do first an “edge-detection”
- Let's see another interesting example on image perception:

  M. Del Viva, G. Punzi, and D. Benedetti. *“Information and perception of meaningful patterns.”* PloS one 8.7 (2013): e69154.

  “… models describe the initial processing of visual information as the extraction of a simplified “sketch” based on a limited number of “salient features” [11], [12], that therefore contains a much reduced amount of information.”

  “We adopt the **principle of maximum entropy** as a measure of optimization: we ask what choice of the **pattern set** is producing the largest amount of “entropy” allowed by the given limitations of the system. We will see that this simple requirement, together with the imposed strict limitations to the computing resources of the system, allows to completely determine the choice of the pattern set from the knowledge of the statistical properties of the input data.”

Image processing, an interesting example with patterns: M. Del Viva, G. Punzi, and D. Benedetti. Fig. 1



Constrain #1: storage

N = number of patterns

Constraint #2: output bandwidth

W = reduction factor (e.g., W=0.001 → 1/1000 can be selected with this pattern set)

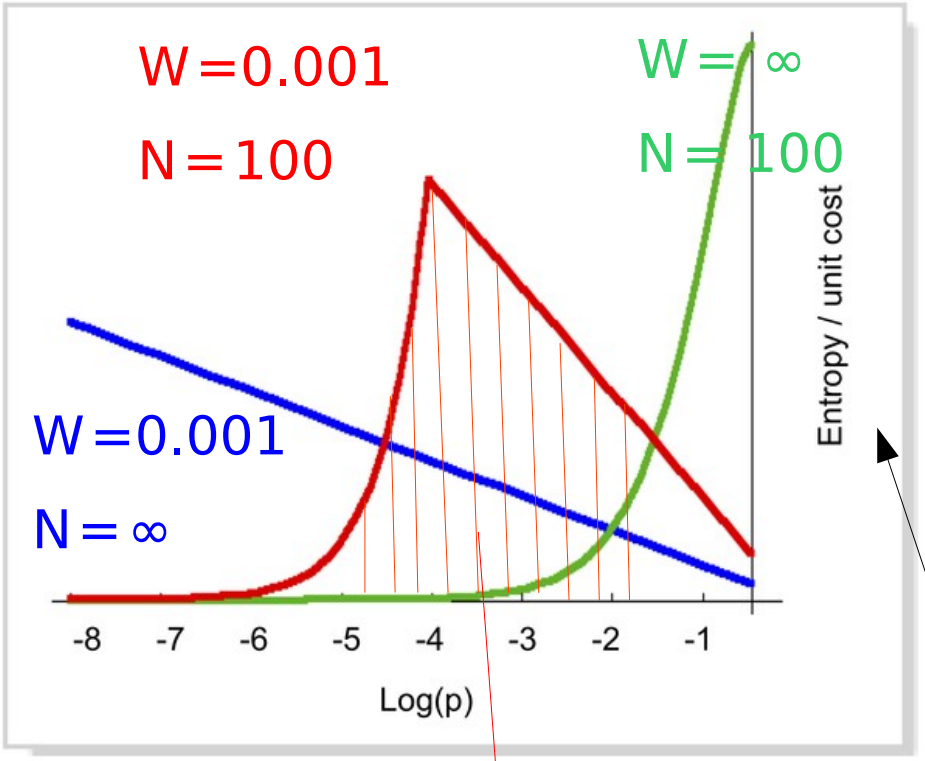
p = probability that the given pattern matches the (sub)image we check

$$f(p) = \frac{-p \log(p)}{\max(1/N, p/W)}$$

unit cost for each pattern

Systems with limitations always have a maximum of the “entropy per unit cost”, f(p), somewhere

Image processing, an interesting example with patterns: M. Del Viva, G. Punzi, and D. Benedetti. Fig. 1



Constrain #1: storage

N = number of patterns

Constraint #2: output bandwidth

W = reduction factor (e.g., W=0.001 → 1/1000 can be selected with this pattern set)

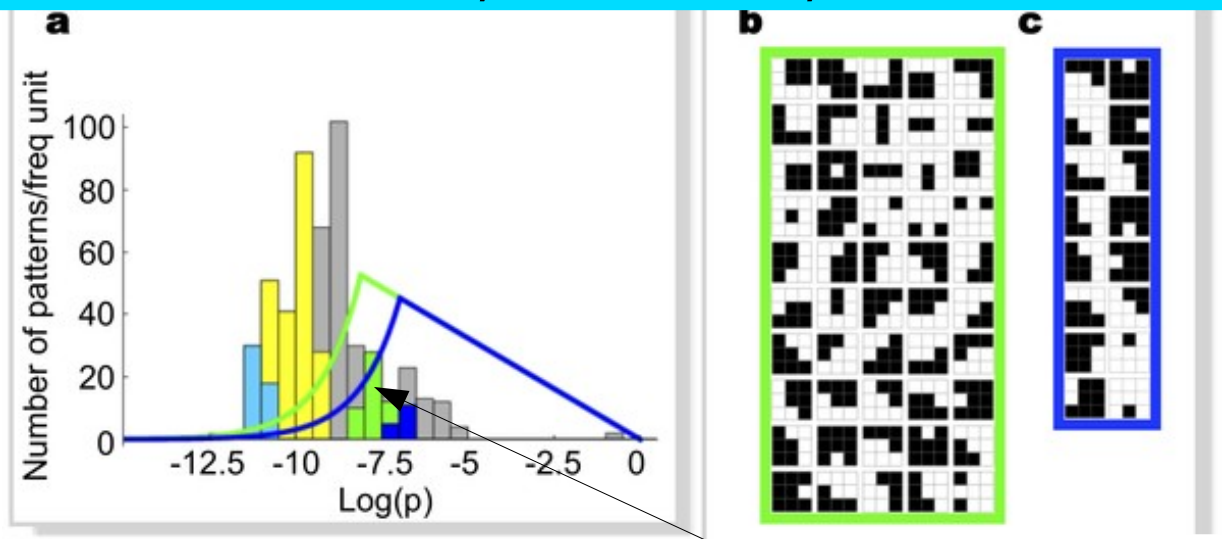
p = probability that the given pattern matches the (sub)image we check

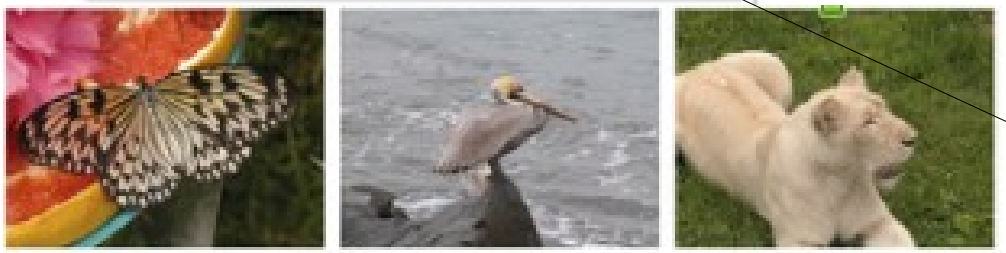
$$f(p) = \frac{-p \log(p)}{\max(1/N, p/W)}$$

unit cost for each pattern

Most relevant patterns fall around this maximum

Image processing, an interesting example with patterns: M. Del Viva, G. Punzi, and D. Benedetti, Fig. 3 and 4

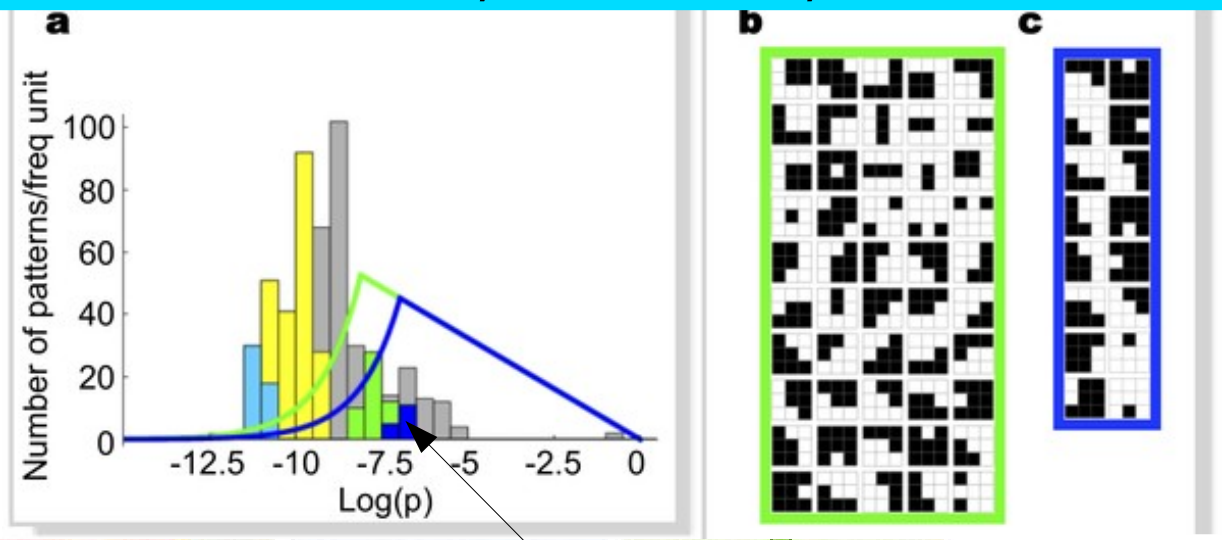


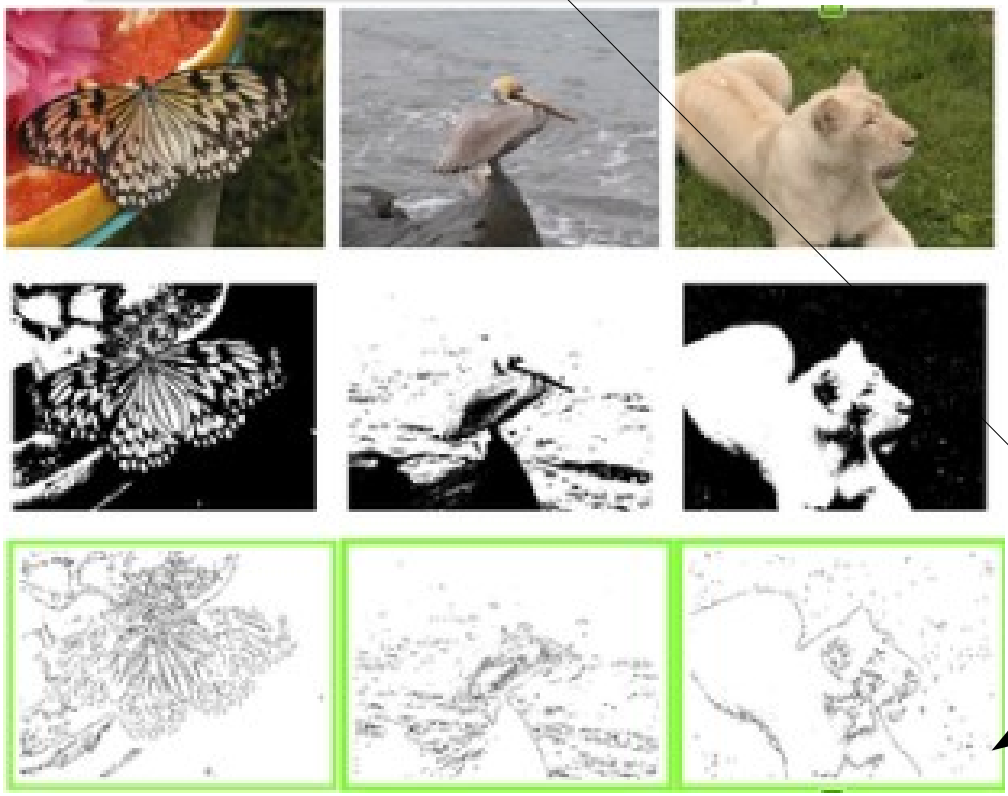


- In a 3x3 grid of possible B&W cells: 512 possible patterns
- Using the Green patterns:
  - the “best” 50 of them

    (use them in the images below)
- Using the Blue patterns:
  - the “best” 15 of them

Image processing, an interesting example with patterns: M. Del Viva, G. Punzi, and D. Benedetti, Fig. 3 and 4



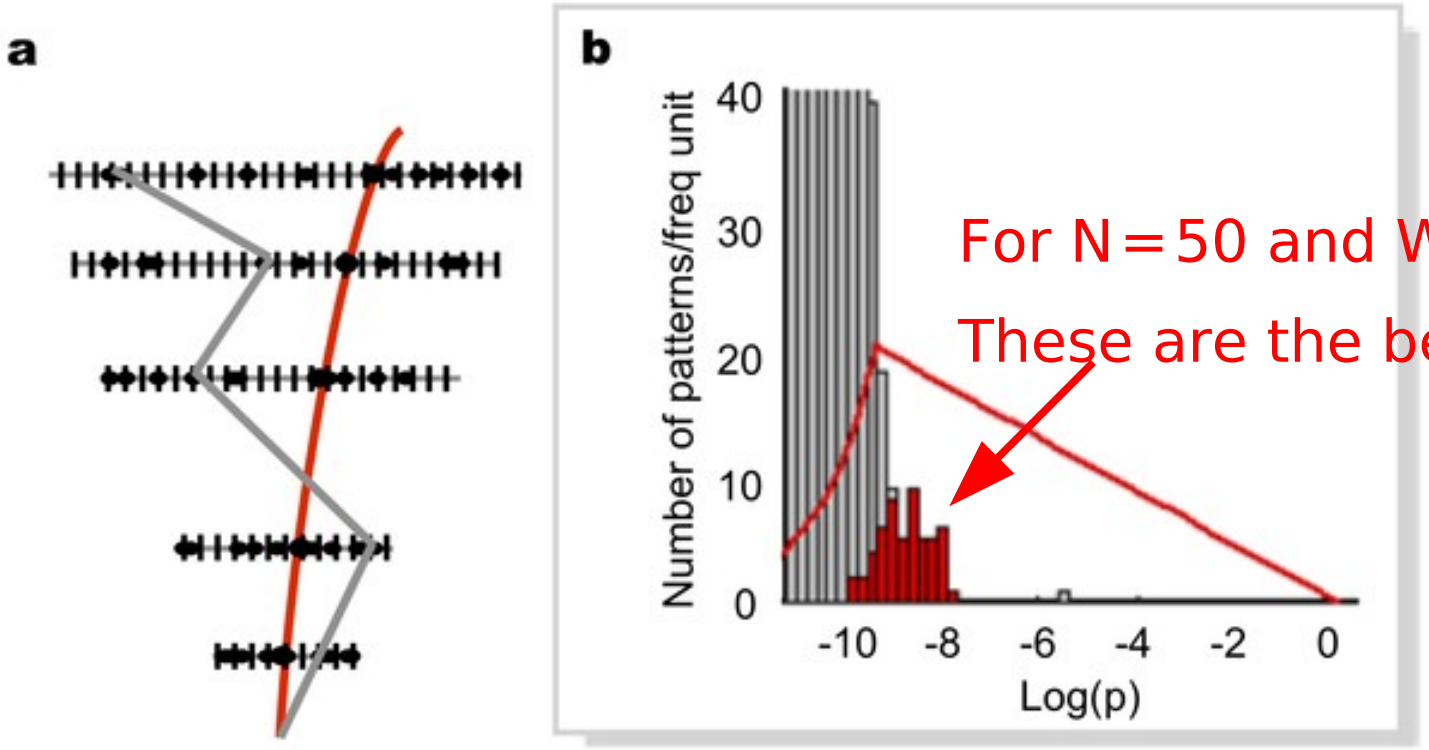


- In a 3x3 grid of possible B&W cells: 512 possible patterns
- Using the Green patterns:
  - the “best” 50 of them

    (use them in the images below)
- Using the Blue patterns:
  - the “best” 15 of them

Image processing, an interesting example with patterns: M. Del Viva, G. Punzi, and D. Benedetti. Fig. 2

Our tracking detectors also produce “images” (= the set of hits), and we select events based on them



Of course, we know that all these zig-zag lines are meaningless
Training on simulated events, to get the patterns with max. entropy, picks up the patterns we also select when we do simulations to define the pattern bank.

Done.
Announcements before closing

There are specialized conferences on tracking :
Connecting The Dots 2023
10-13 Oct 2023, Toulouse, France https://indico.cern.ch/event/1252748/)

An announcement: MOCAST 2023

http://mocast.physics.auth.gr/

28-30 June 2023, Athens, Greece.

Summary

- Saw that need fast **tracking information at the Trigger** of High Energy Physics experiments
- We split the problem into **“track finding”** (define fast a “road” where a track can be) and **“track fitting”** (determine the track parameters)
- Saw in some detail the ATLAS FTK and HTT case, using
  - **Track finding** with **Pattern matching** in Associative Memories , and **Track fitting** in FPGAs
  - Basically we saw that: if we want to avoid or cannot afford calculating something time consuming, we can split the problem and use pre-calculated patterns and quantities.
- We saw also examples of other approaches, with both steps done in FPGAs
  - Track finding with Hough transform, Artifical Retina Algorithm, Tracklet seeding.
  - Track fitting with linearised track fitting, Kalman filter.
- We saw an example of patterns used in image processing

Thank you!

Acknowedgments especially to:
Alberto Annovi, Francesco Crescioli, Mauro Dell' Orso, Paola Giannetti, Andrea Negri , FTK and HTT members

Extras...

Tracking at HEP: AMs , CPUs, FPGAS and GPUs

- V. Halyo, et. al., *"GPU Enhancement of the Trigger to Extend Physics Reach at the LHC,"* Journal of Instrumentation 8 P10005, 2013.
- C. Gentsos, F. Crescioli, P. Giannetti, D. Magalotti, S. Nikolaidis, *"Future evolution of the Fast TracKer (FTK) processing unit"*, PoS (TIPP2014) 209
- A. Annovi, et al., *"Associative Memory for L1 Track Triggering in LHC Environment,"* in IEEE Trans. on Nuclear Science, Vol. 60, No. 5, pp. 3627 – 3632, 2013.
- G. Hall, et al., *"A time-multiplexed track-trigger for the CMS HL-LHC upgrade"*, in NIM A, Vol.824, 11 July 2016, pp. 292–295
- A. Abba et al., *"Simulation and performance of an artificial retina for 40 MHz track reconstruction"*, in JINST 10 C03008 (2015)
- …etc, etc….

High**L**uminosity-**LHC** (HL-LHC): pile-up of ~140-200 events/crossing will be typical; up to 200 events per crossing are considered likely.

At L1: need tracking in <10 μs (<2-3μs giving time to rest)

Extras

- D. Emeliyanov, et al., “GPU-based tracking algorithms for the ATLAS high-level trigger” in Journal of Phys. Conf., Ser. 396, 012018, 2012.
- J. Mattmann, et al., “Track finding in ATLAS using GPUs,” in Journal of Phys. Conf., Ser. 396, 022035, 2012.
- Y. Ago, Y. Ito, and K. Nakano, “An FPGA implementation for neural networks with the FDFM processor core approach,” International Journal of Parallel, Emergent and Distributed Systems, vol. 28, no. 4, pp. 308–320, 2012.

FTK - Fast Tracker for Hadron Colliders

An FP7 IAPP project (February 1, 2103 - January 31, 2017)

Home

Jobs

Partners

- Pisa (coord)
- AUTH
- CAEN
- CERN
- CNRS/Paris
- Prisma Electronics

The Project

- The FTK Project
- FTK application in HEP
- Other FTK applications
- Transfer of Knowledge

Dissemination

- Talks and Proceedings
- Papers
- FTK events

Collaboration

http://ftk-iapp.physics.auth.gr/

This project aims to develop an extremely fast but compact processor, with supercomputer performances, for pattern recognition, data reduction, and information extraction in high quality image processing.

The proposed hardware prototype features flexibility for potential applications in a wide range of fields, from triggering in high energy physics to simulating human brain functions in experimental psychology or to automating diagnosis by imaging in medical physics. In general, any artificial intelligence process based on massive pattern recognition could largely profit from our device, provided data are suitably prepared and formatted.

The project has received funding from the
European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement n.324318
Participants (2 SMEs and 4 Academic Institutions)

FP7 Project 324318

University of Pisa, Italy

Aristotle University of Thessaloniki (AUTH), Greece

CAEN SpA, Italy

CERN, Switzerland

CNRS, France

Prisma Electronics ABEE, Greece

Contact: Mauro Dell'Orso

Input Mezzanine card(IM)
+ Data Formatter(DF)

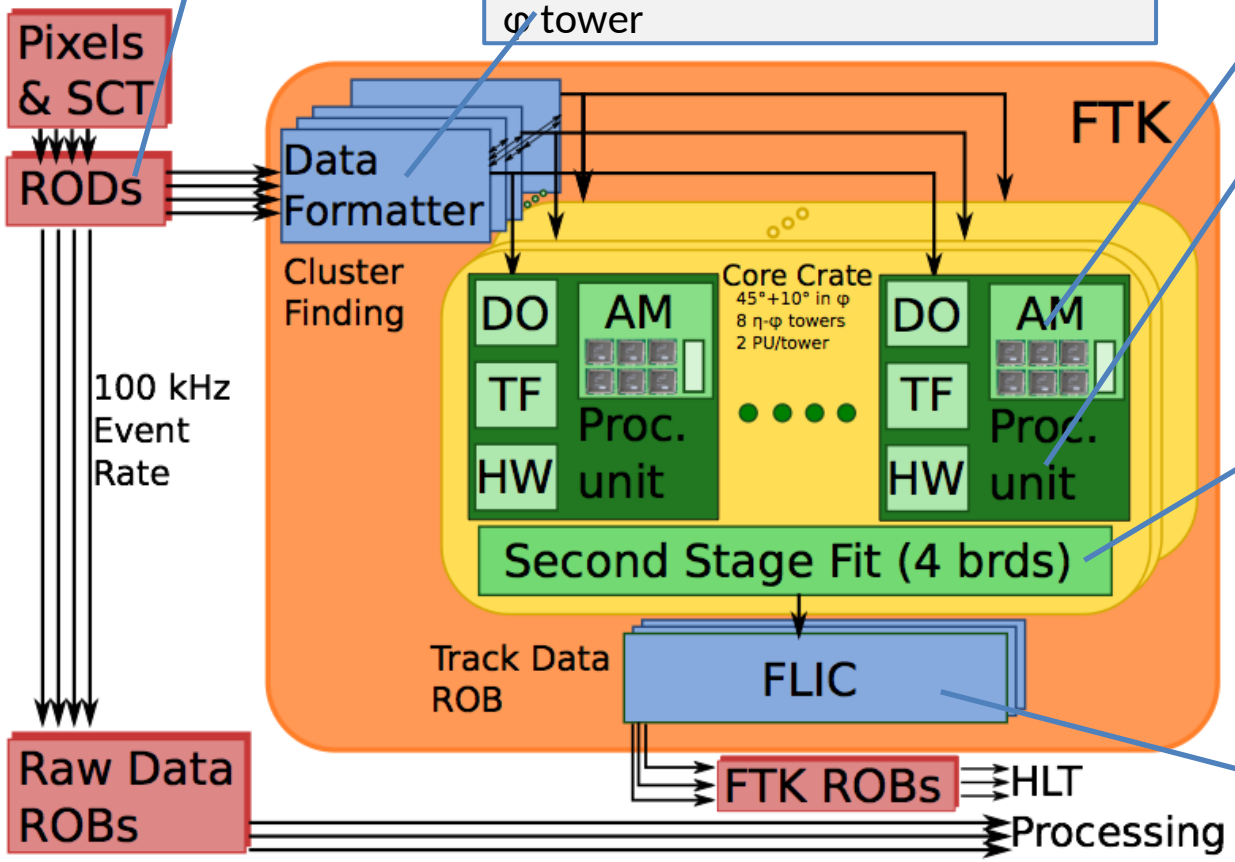
Processor Units: Auxiliary card(AUX) +
Associative Memory Board(AM)

IM: Receive the hits and perform clustering

DF: hit sharing and provide pipeline (the "custom switch" to fan-out hits to the relevant Processor for this η-φ tower

Dual HOLA card

Copy the hit from ID and send to FTK



AM: pattern recognition in SuperBin ("SuperStrip") resolution

AUX: a) mapping between hits and SuperStrips",
b) track fitting: pt, η, φ, d0, z0

Second Stage Board(SSB)

Reduce the fake track using remaining silicon layers.

FTK to Level2 Interface Crate(FLIC)

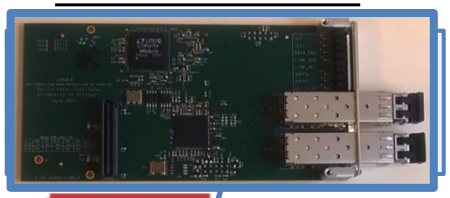
Send track info to HLT

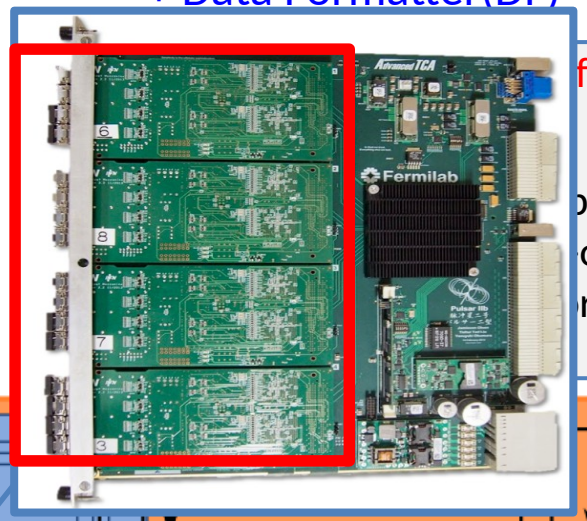
* Red: involvement of the group

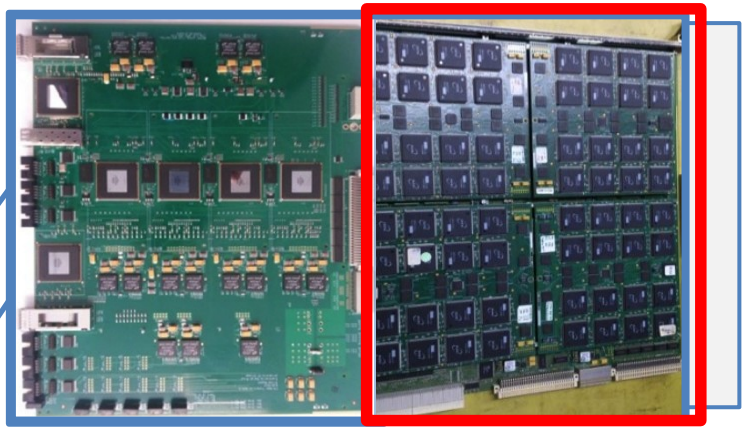
Input Mezzanine card(IM)
+ Data Formatter(DF)

Processor Units: Auxiliary card(AUX) +
Associative Memory Board(AM)

Dual HOLA card







Pixels & SCT

RODs

Data Formatter

Cluster Finding

FTK

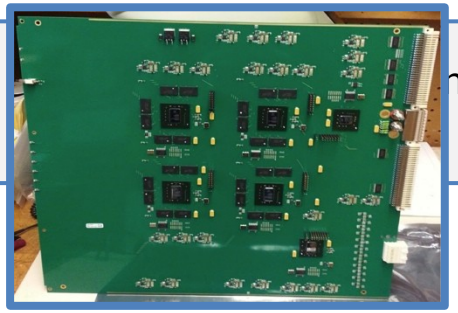
Core Crate
45°+10° in φ
8 η-φ towers
2 PU/tower

DO
TF
HW
AM
Proc. unit

DO
TF
HW
AM
Proc. unit

Second Stage Fit (4 brds)

Second Stage Board(SSB)

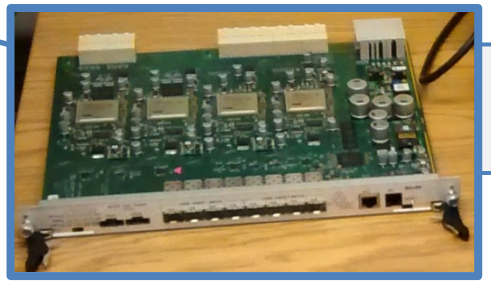


100 kHz Event Rate

Track Data ROB

FLIC

FTK to Level2 Interface Crate(FLIC)



Raw Data ROBs

FTK ROBs

HLT Processing

* Red: involvement of the group

FTK : working configuration

- High resolution patterns: $(15x36)_{pix}x16_{sct}$
  - Pixels: 15 channels along ϕ, 36 ch. along η
  - Strips: 16 strips
- Background events with 69 superimposed pp collisions
  - Instantaneous luminosity $3*10^{34}$ Hz/cm^2
- Hardware constraints (for each of 64 η-ϕ towers)
  - # AM patterns < $16.8 * 10^6$
  - # roads/event < $16 * 10^3$
  - # fits/event < $80 * 10^3$

DC bits group detector channels together and increase the pattern resolution

Work load for track fitter

	Coarse resolution roads	Max # DC bits / layer	# AM pattern * 10^6	Efficiency %	roads / evt * 10^3	fits / evt * 10^3
Barrel	$(30x72)_{pix}x32_{sct}$	$2_{pix}x1_{sct}$	16.8	93.3%	3.2	26
Endcap	$(30x72)_{pix}x32_{sct}$	$2_{pix}x1_{sct}$	16.8	91.2%	6.9	55

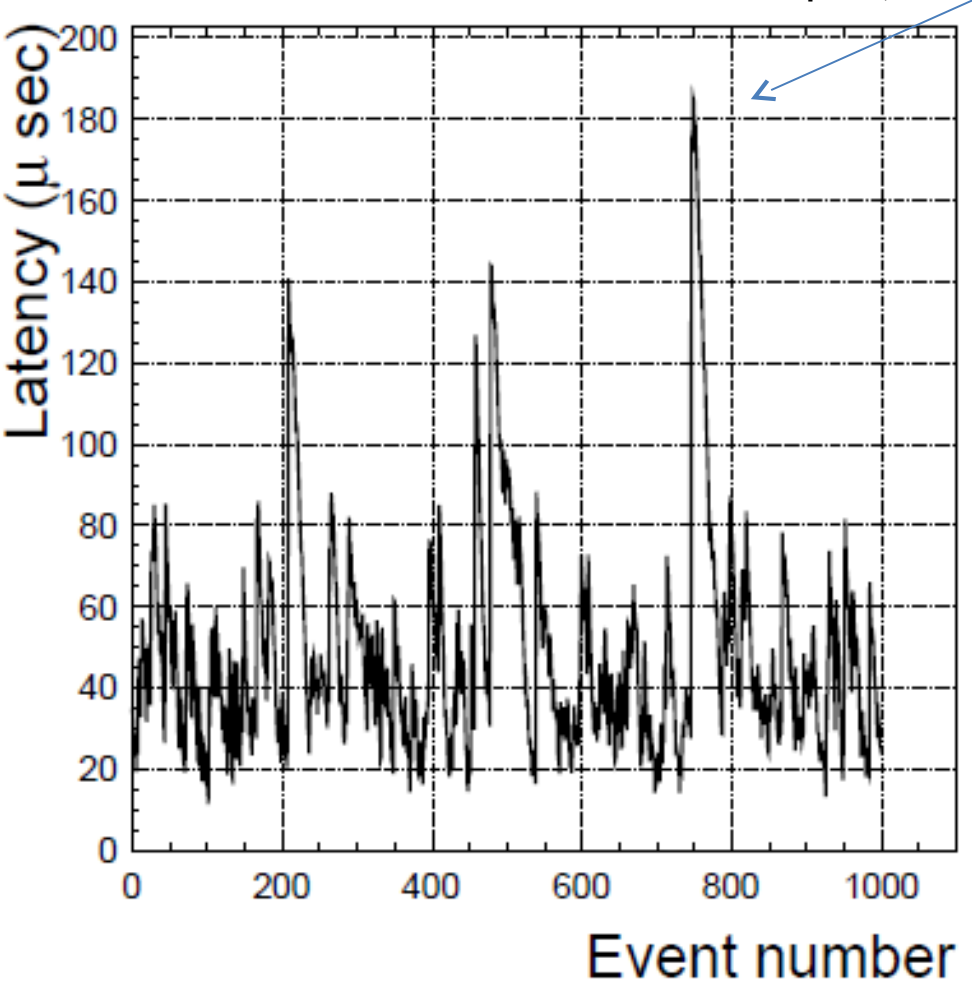
Alberto Annovi
* *ANIMMA - A new "Variable Resolution Associative Memory" for High Energy Physics ATL-UPGRADE-PROC-2011-004, doi:10.1109/ANIMMA.2011.6172856*
* *"Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013*

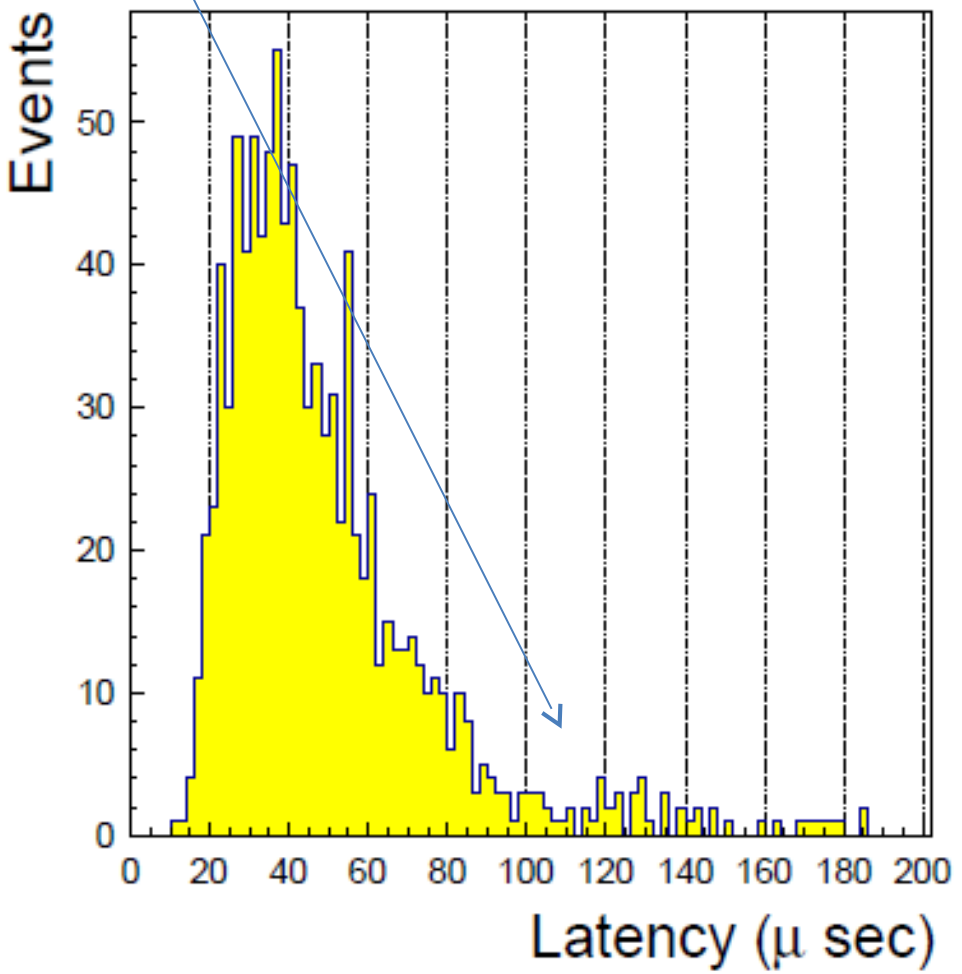
FTK Latency

FTK has enough processing power at L=3x1034cm-2s-1 (operating rate ~60%)

Latency was rise-up by heavy event, but after such an event the latency quickly return to the typical range.

L=3x1034 MC sample (Z->mm) @ 100 kHz LVL1 rate.

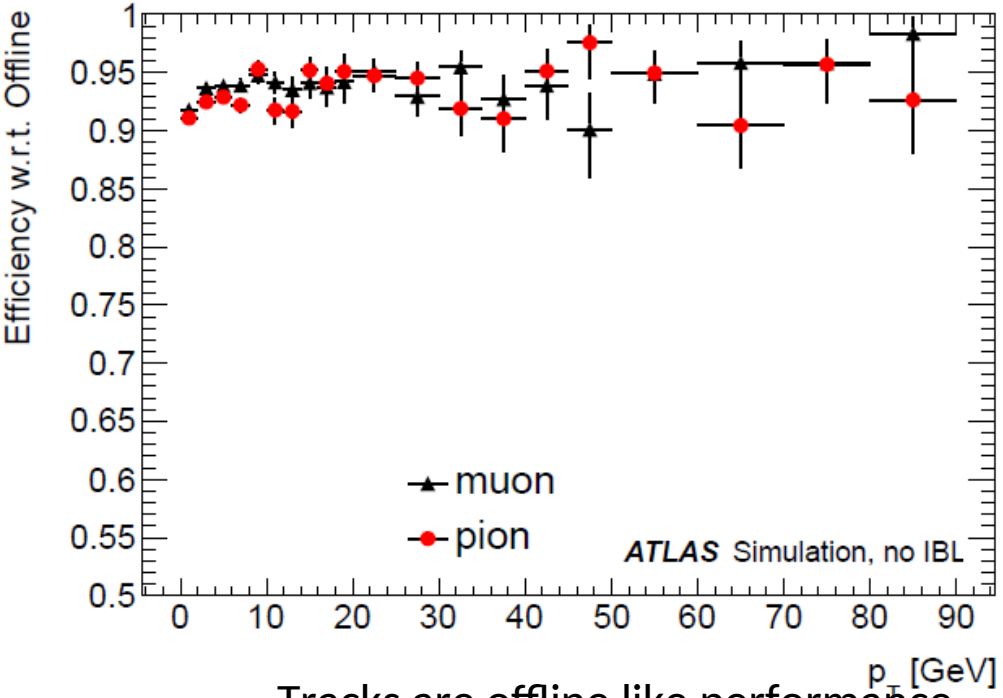


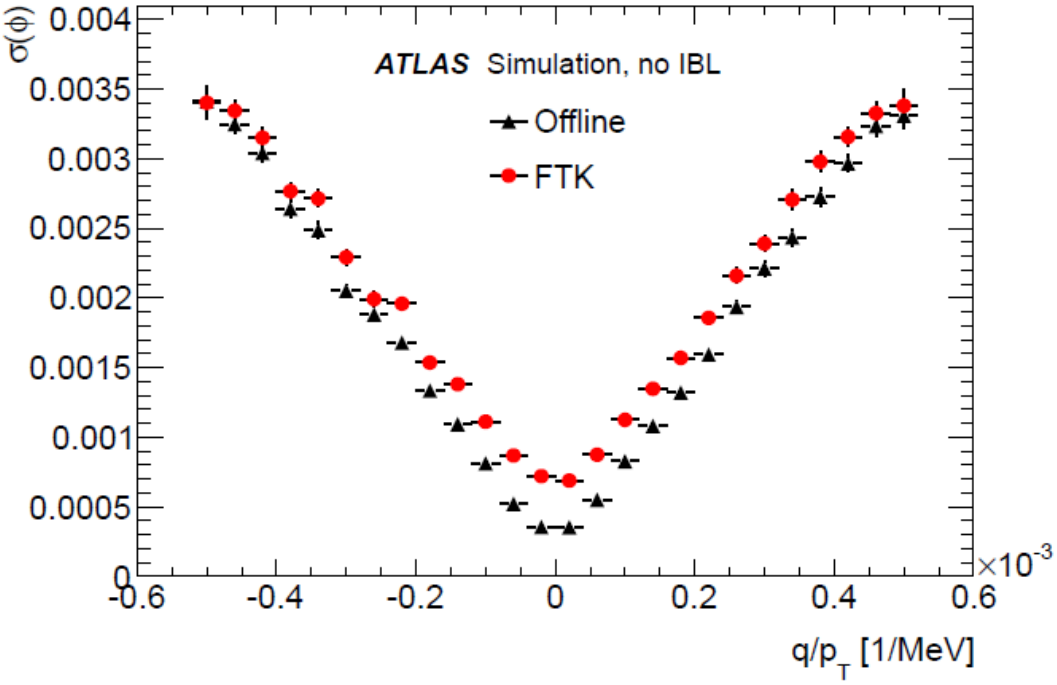


Averagely latency is ~50 μsec and maximum on tail is ~ few handed μsec. It is enough speed for HLT requirement.

FTK Track performance

All results are base line of FTK performance!





Tracks are offline like performance

Difference is ：

- Algorism of hit clustering
- Lack of Low Pt patterns
- Broken of linear approximation.
- No TRT,　　not δray correction, etc

More than 90 % efficiency with respect to offline.

Beyond FTK

- FTK was using the AMchip06, an Associative Memory with
  - 128k patterns of 8 words × 18 bits each word
  - high speed serial links
  - variable resolution (up to 6 ternary bits)
  - low power
  - 8 × 16 bit comparisons at 100 MHz

- The ATLAS Hardware Track Trigger (HTT) for the HL-LHC era, was to use an AM chip (AMchip09) with many more patterns (~400k patterns/chip).
  - Applications outside HEP (medical imaging, smart cameras, genomics, ...)

Kalman Filter

- From wikipedia https://en.wikipedia.org/wiki/Kalman_filter

Innovation or measurement pre-fit residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k\vert k-1}$
Innovation (or pre-fit residual) covariance	$\mathbf{S}_k = \mathbf{H}_k\hat{\mathbf{P}}_{k\vert k-1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \hat{\mathbf{P}}_{k\vert k-1}\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\mathbf{x}_{k\vert k} = \hat{\mathbf{x}}_{k\vert k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\vert k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\hat{\mathbf{P}}_{k\vert k-1}$
Measurement post-fit residual	$\tilde{\mathbf{y}}_{k\vert k} = \mathbf{z}_k - \mathbf{H}_k\mathbf{x}_{k\vert k}$