Intelligent triggering: pattern recognition with Associative Memories and other tools (& the Fast TracKer example in ATLAS)

Kostas Kordas

Aristotle University of Thessaloniki

FP7 Grant No: 324318

SEVENTH FRAMEWORK PROGRAMME

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ISOTDAQ2014 , Budapest, January 28 - February 5, 2014

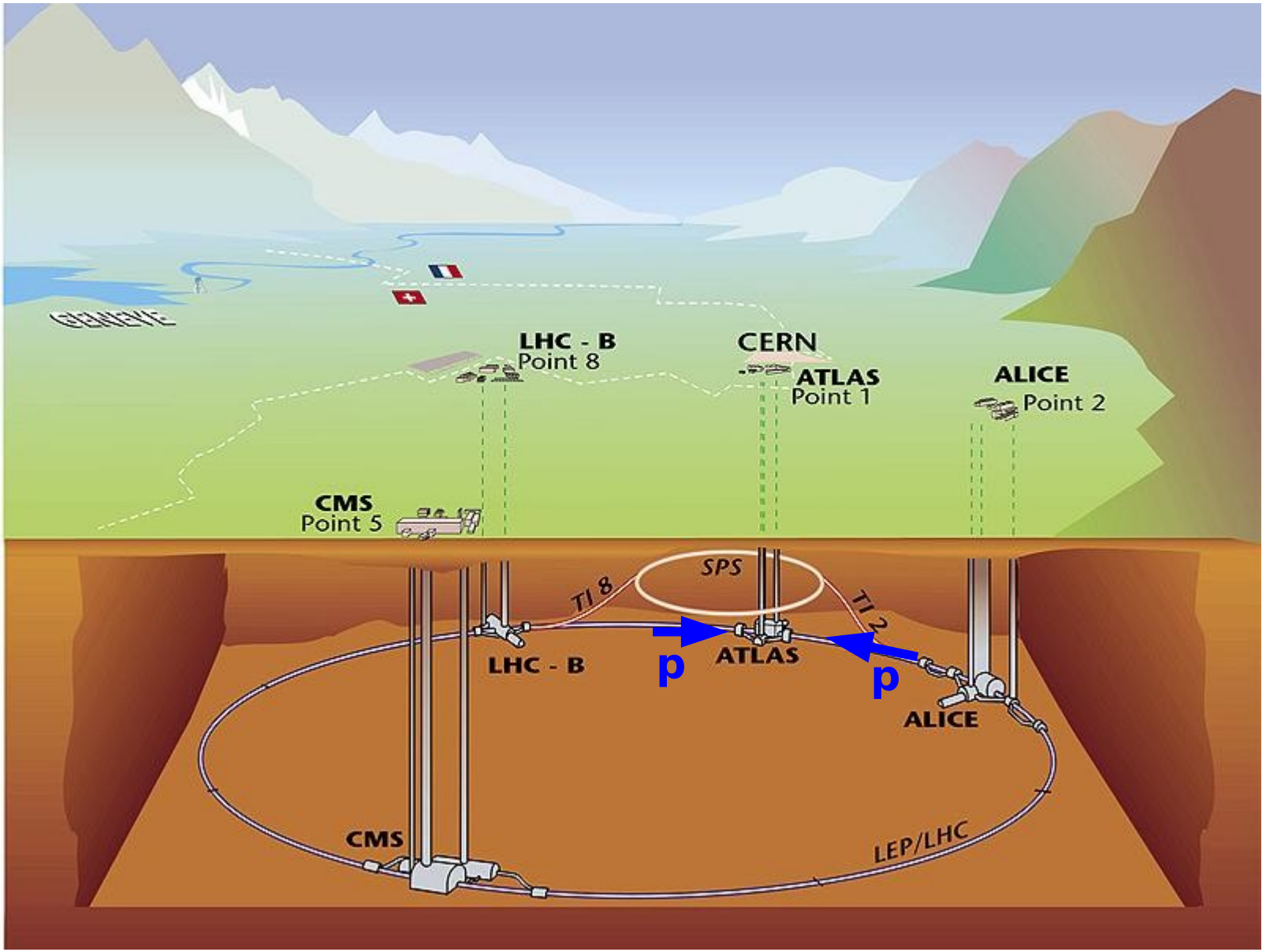
Overview

- Trigger and DAQ in HEP experiments
- The need for tracking info at the Trigger and how to do it fast
- FTK : 2-stage track fitting
- Pattern matching with Associative Memories
- Refined fitting with FPGAs

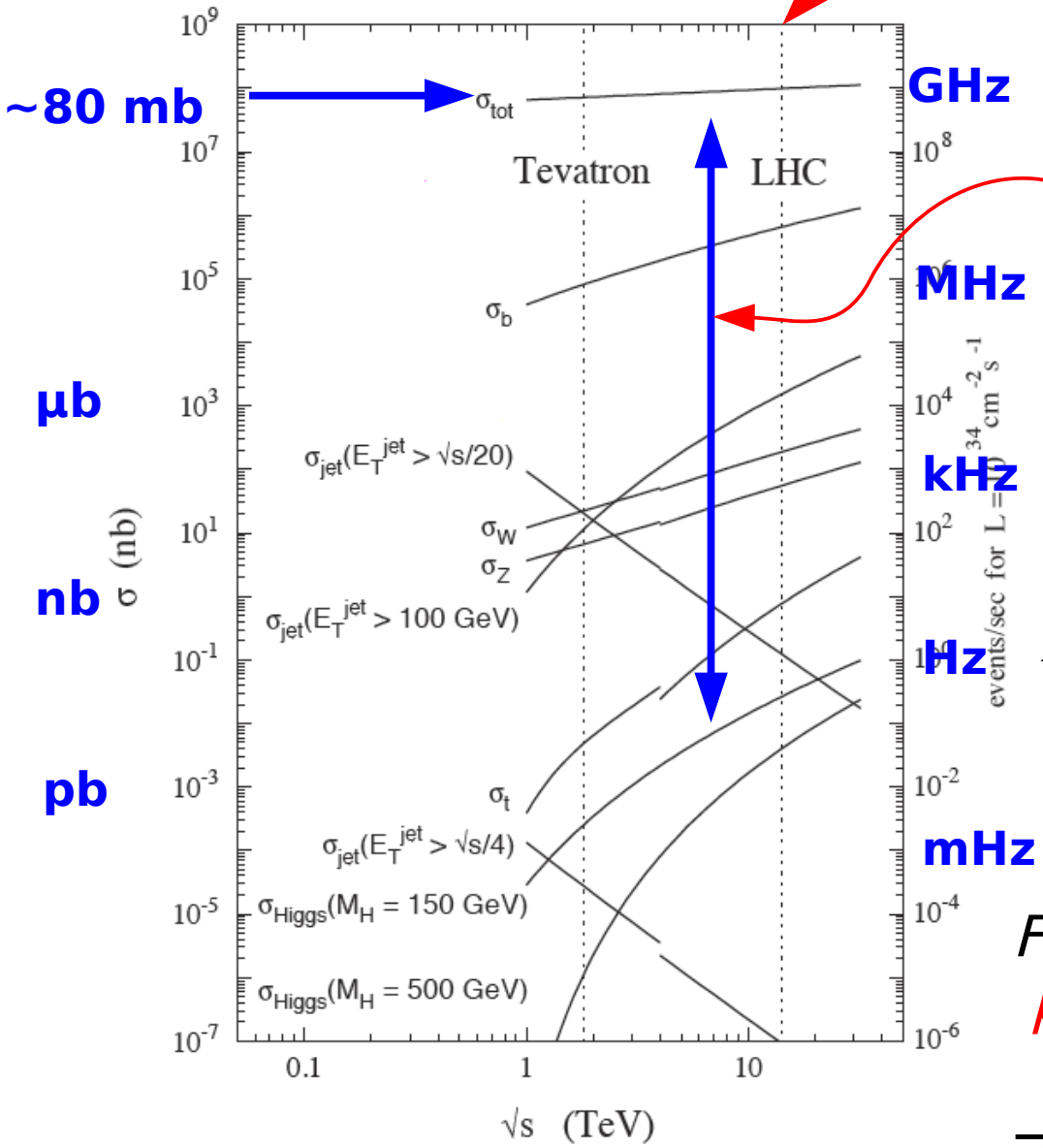
Basically you'll see that:

if you want to avoid or cannot afford calculating something time consuming, split the problem and just use pre-computed patterns and quantities.

Experiments at the LHC - pp collisions



The energy frontier -
interested in rare processes



$Design: \sqrt{s} = 14\,TeV$

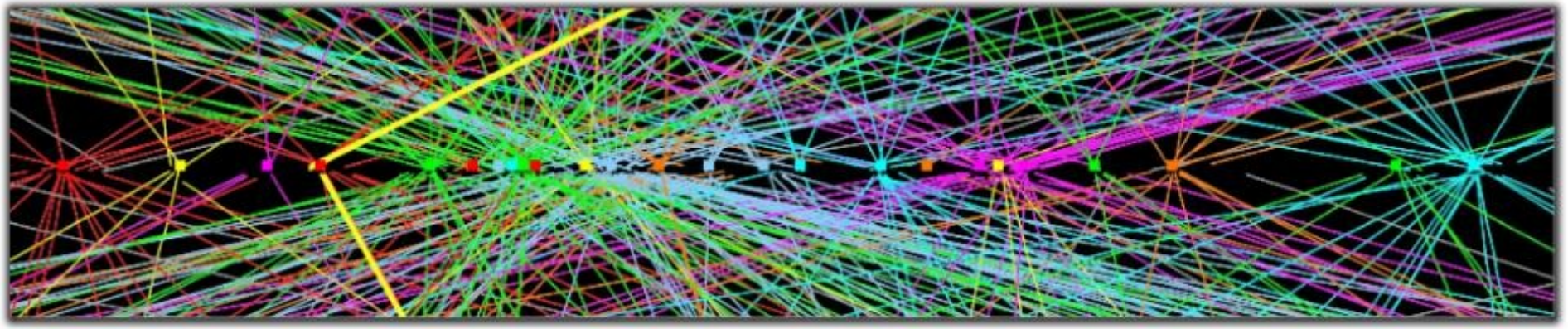
>10 orders of magnitude!
(>12 orders when including branching ratios to leptons,
e.g. H → Z Z → μ⁺ μ⁻ μ⁺ μ⁻) !!!

In order to have a reasonable number of observed events, we need high luminosity colliders

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

For 25ns bunch spacing:
pp interactions/bunch crossing:
<μ> ~ 25
→ L = 3-5 times more coming soon

Looking at many & complex events
every 25ns look at a superposition of ~25 pp collisions



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

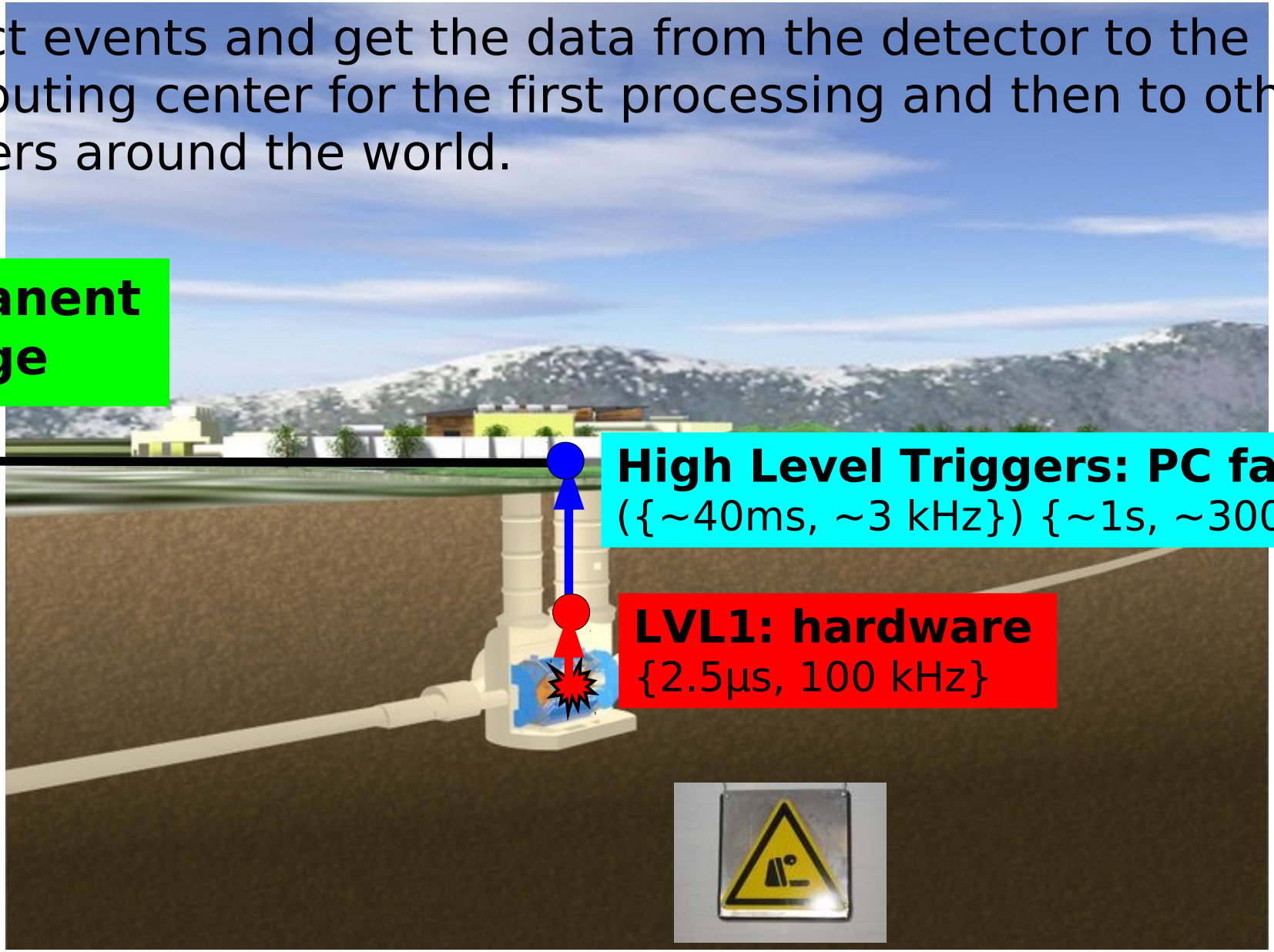
~90 M channels in modern HEP experiments, recording signals from these, produces:

→ **1.5 MB of info per "event", i.e., every time I read this info**

* The Trigger and Data Acquisition system, has to watch ~1 billion pp collisions per second (40M proton bunch crossings / sec),
* **select online "the most interesting" O(300) events/sec,**
* **and log them for offline use with a resolution of a ~90 Mpixel camera.**

The job of a Trigger&DAQ system

- Select events and get the data from the detector to the computing center for the first processing and then to other centers around the world.



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

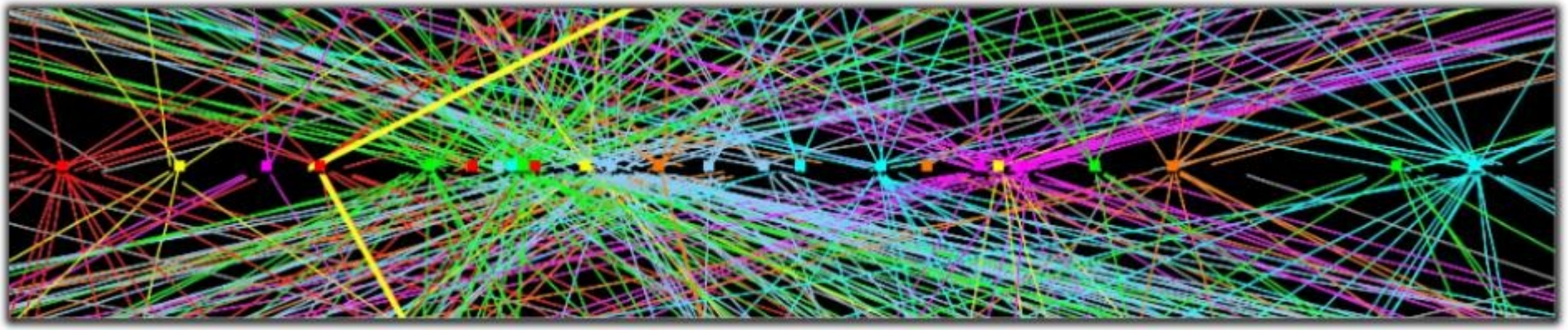
When we have a limited time budget, we decide based on the muon and the calorimeter systems.

There are channels, where you absolutely need the tracks as early as possible in order to make a decision and keep as much signal as possible

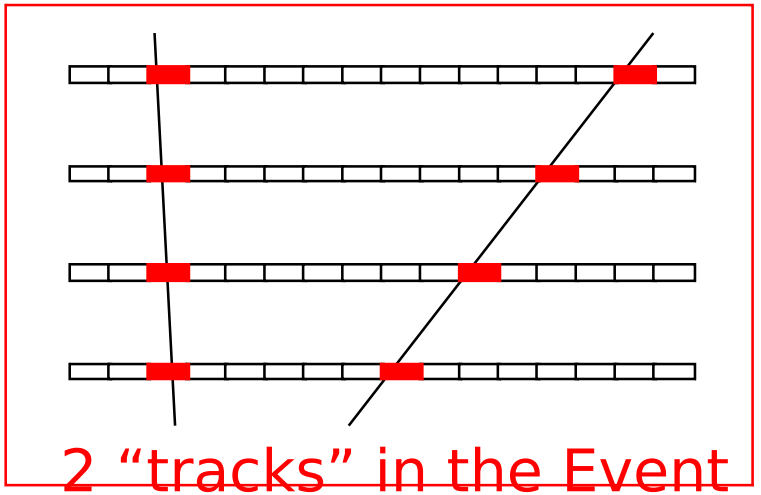
e.g., 2 “jets” of tracks, which are usually boring, they could actually be

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

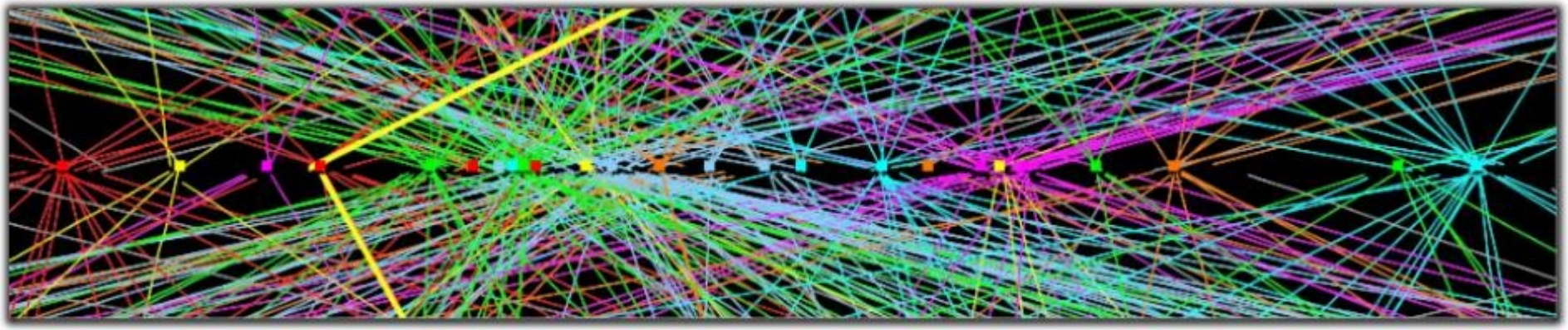
Tracking is a combinatorics problem: which combinations of hits fit track hypothesis?



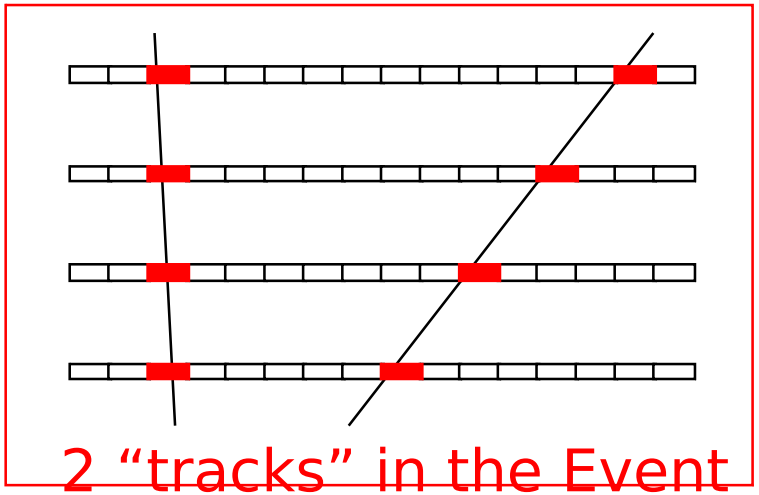
Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.



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

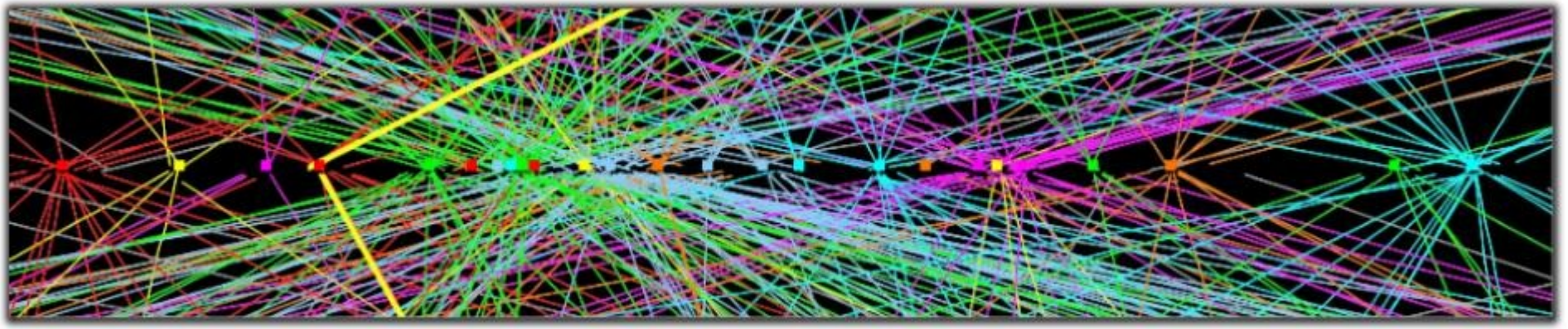


Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.

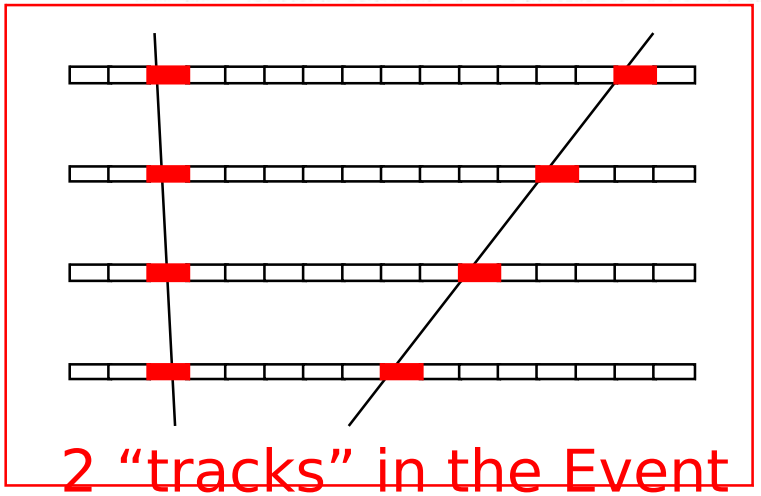


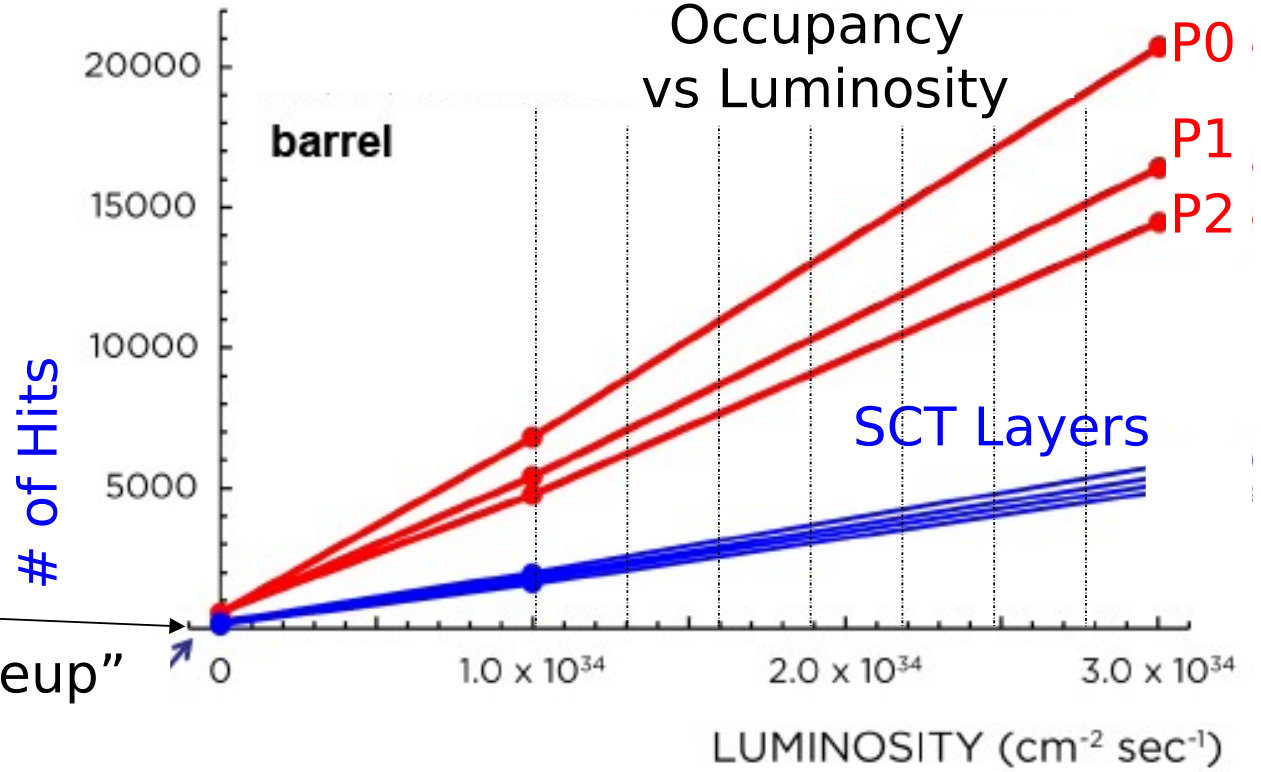
Connecting the “hits”
from one detection layer to the other,
to trace the charged particle's path
as it moves radially outward and its'
position is measured in
each detector layer, divided in
a number of “bins”.

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.

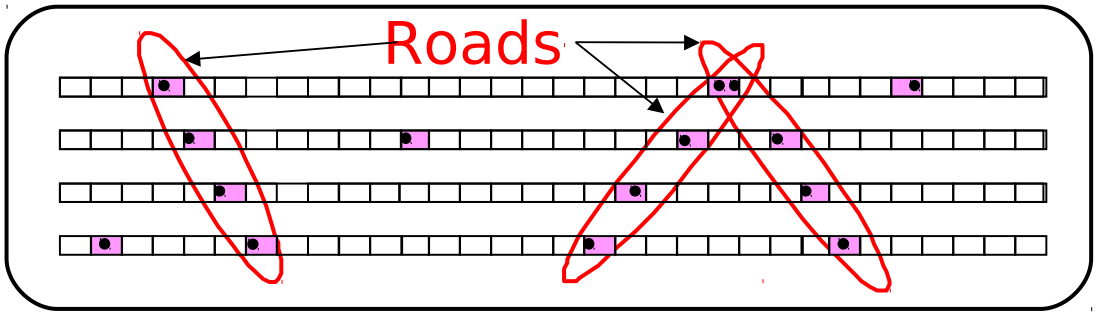




Y intercept:
“Hard scattering”
negligible compared to “pileup”
pp interactions

Very fast tracking – tracking in 2 steps

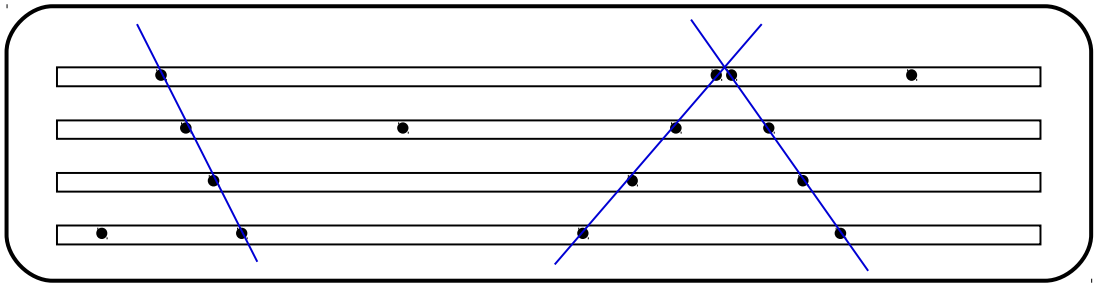
1. Find low resolution track candidates called “roads”. Solve most of the combinatorial problem.



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.

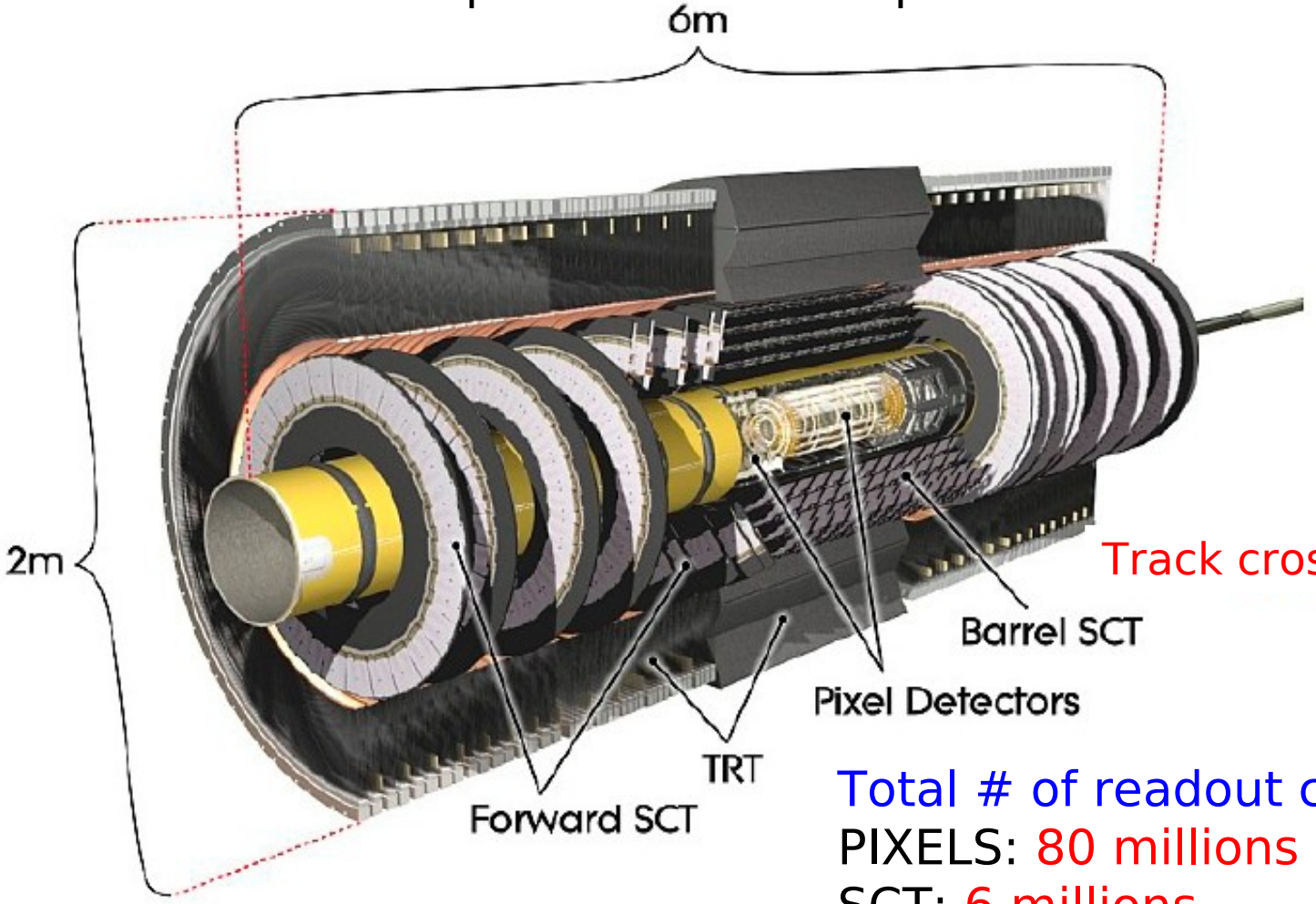


http://www.pi.infn.it/~orso/ftk/IEEECNF2007_2115.pdf

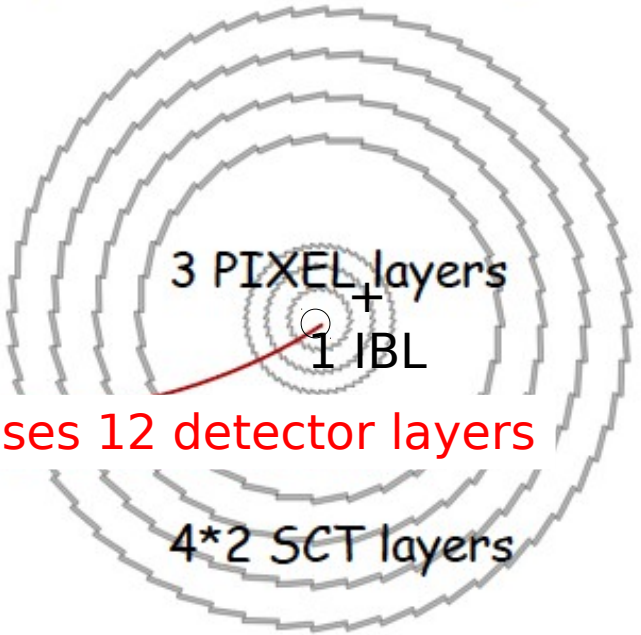
Excellent results with linear approximation!

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 L2. Typical FTK latency ~100μs, compared to O(50ms) L2
Advantages: high-bandwidth connection with detector &
HW optimized for the specific tasks



Example:
R-phi view of Barrel region:

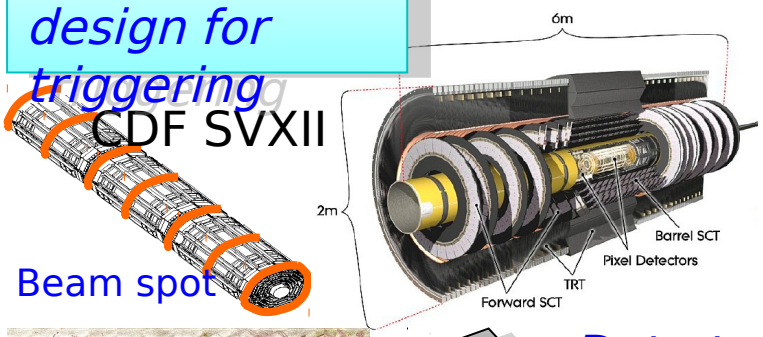


Track crosses 12 detector layers

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

From hits to tracks ready for the HLT in < 100 μs

Detector design for triggering



For L2: SVT trigger at CDF
For L2: FTK in ATLAS
For L1? : CMS & ATLAS

Data transfer

Data formatting

Here FPGAs cluster hits with sliding window and get their centroid as the hit position. Forward the data to proper Processing Units

Pattern Recognition

Associative Memory approach discussed here

Track Fitting

FPGA (here)

HLT

Processing Units (PUs) made of these two steps

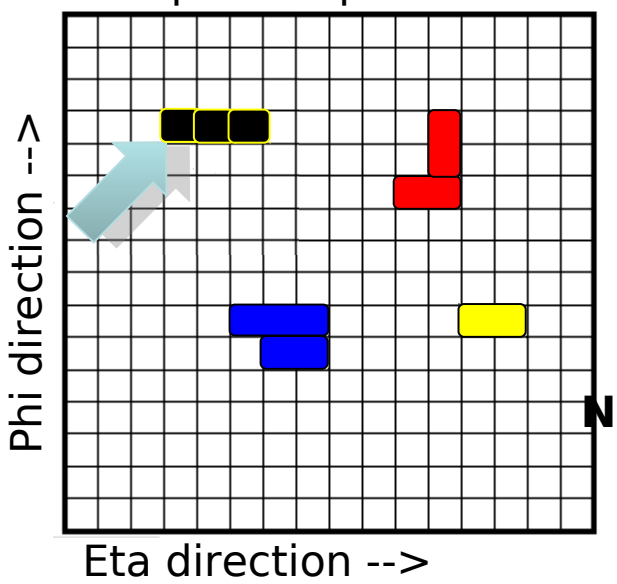
Each PU, takes care of a given detector η-φ tower
In FTK: 64 towers

1. Data Formatting:

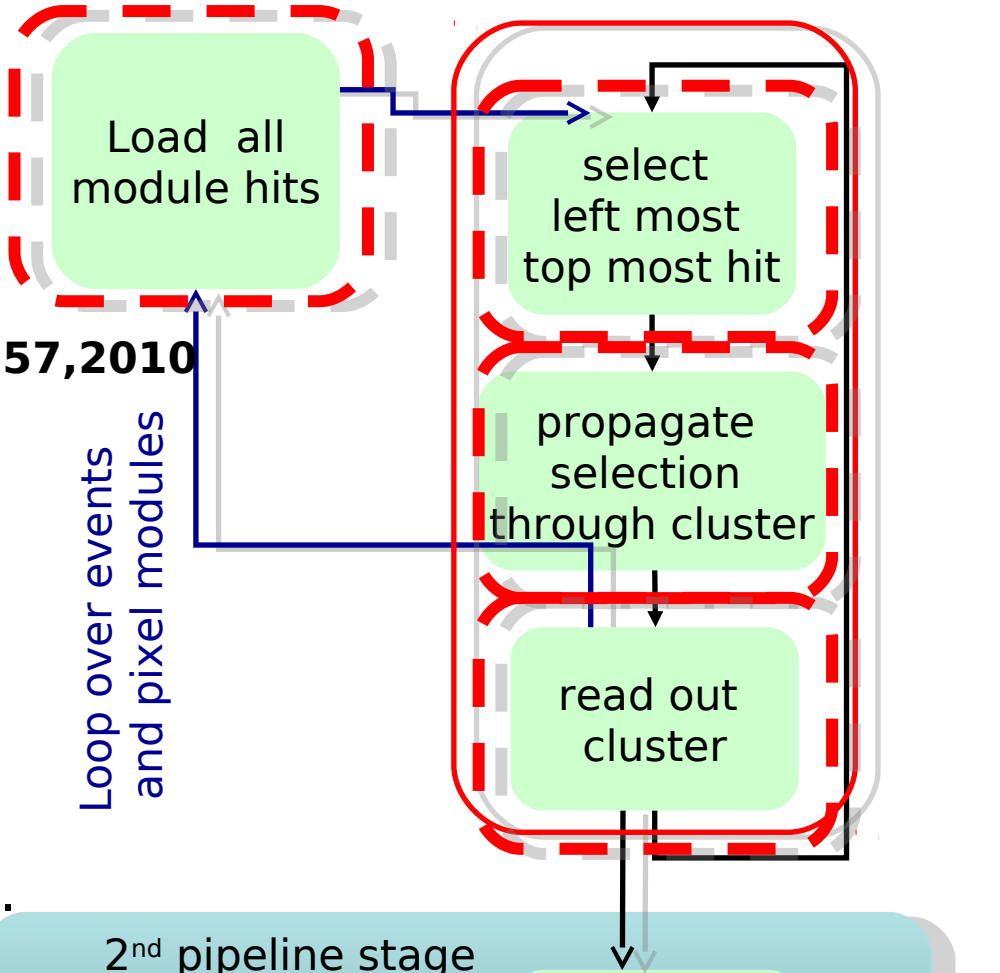
cluster adjacent hits,
find the position of each cluster,
forward them to the Processing Unit responsible for this geometrical region

Clustering algorithm: how-to

FPGA replica of pixel matrix



NIM A617:254-257,2010



1st phase:

- Pixel module: a 328x144 matrix.
- Replicate a part of it (8x164) in hw matrix.
- The matrix identifies hits in the same cluster (local connections).

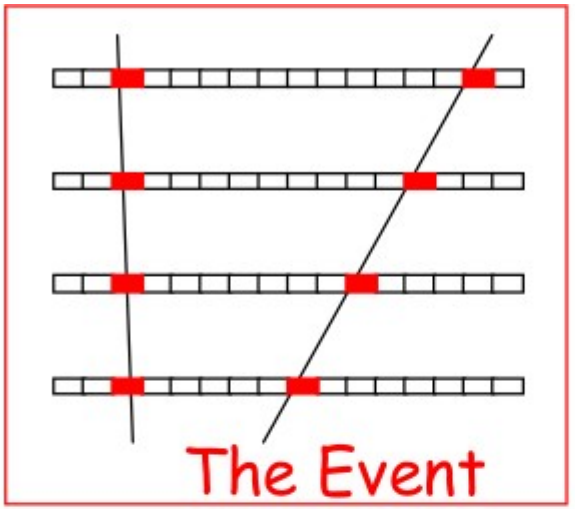
2nd phase:

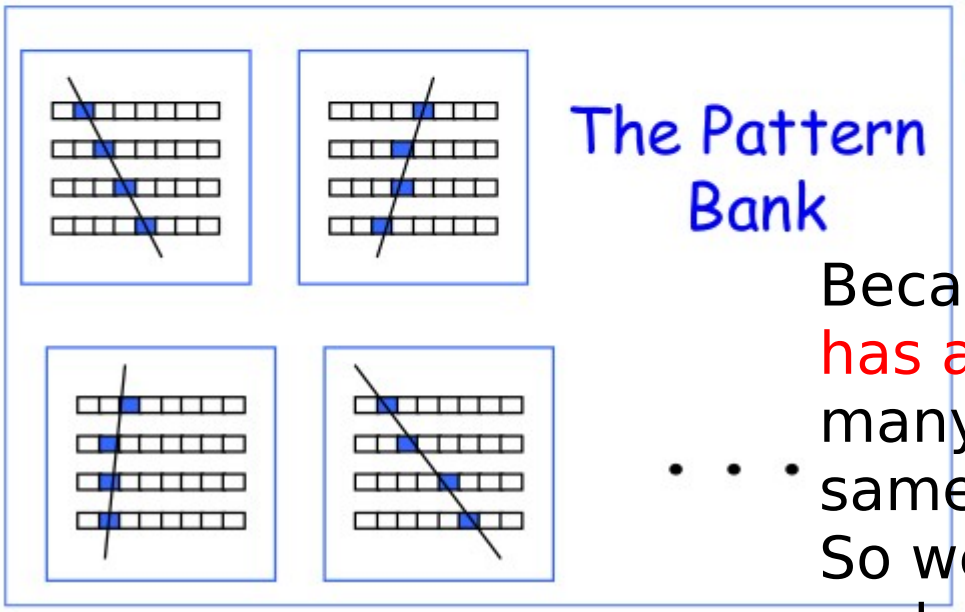
- Hits in cluster are analyzed (averaged).
- Flexibility to choose algorithm!

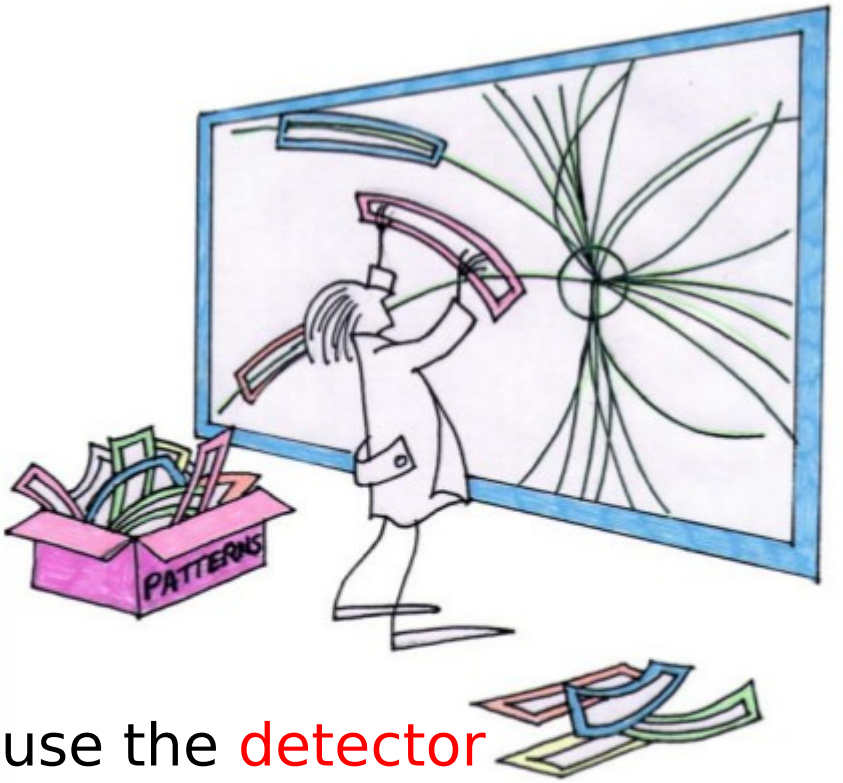
2.
The coarse pattern matching first

Using 8 out of the 12 silicon layers

Matching hit combinations to the known patterns, you find the tracks





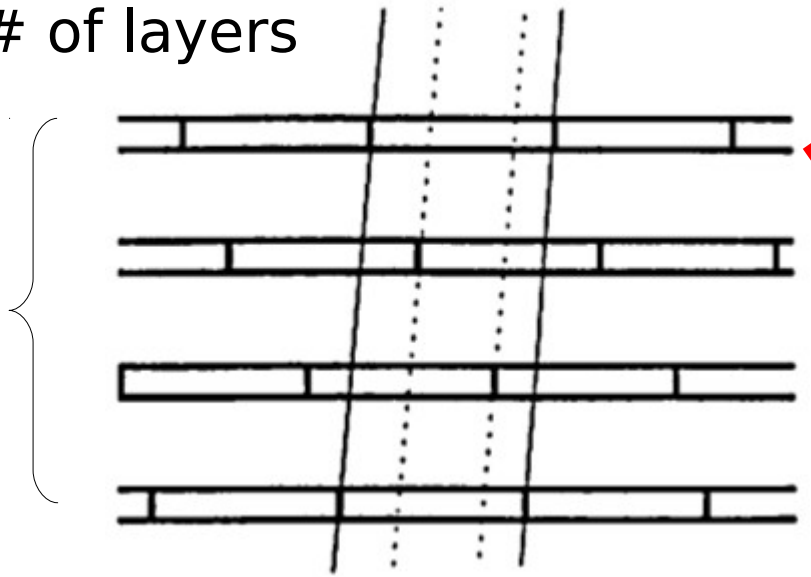


Because the detector has a finite resolution (“bin size”), many different tracks generate the same hit pattern,
So we have a finite number of patterns and a finitie-size pattern-bank.

How many patterns you need in your bank?

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, $N_p = 7\ 10^{12}$

!!!

$N_{patterns}$ 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).
- And still have to match the patterns
  - 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.
  - etc
  - **speed is extremely important at triggering, so let's pay attention to this problem first**

2a.

Find track candidates very very fast

Ultimate speed: Associtive memories

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.

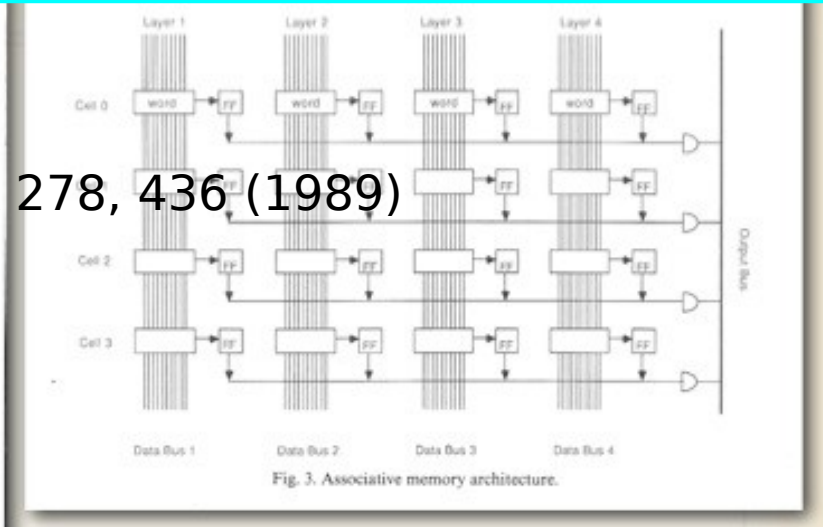


Fig. 3. Associative memory architecture.

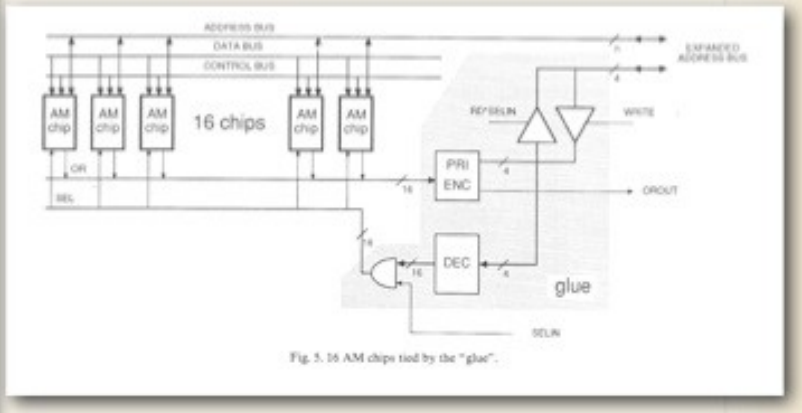


Fig. 5. 16 AM chips tied by the "glue".

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

What is an Associative Memory (AM) a.k.a Content Addressable Memory (CAM)?

- A memory that is accessed by its **contents,** not its **location.**
- E.g., while in a RAM we ask:
  - what you have in location ***xyz***?
- In an Associative Memory we ask:
  - Are there any locations holding the value ***abc***?

Binary CAMs

- **Binary CAM** (simplest):
  - uses data search words consisting entirely of 1s and 0s.

Ternary CAMs

- **Ternary CAM**: added flexibity to the search
  - allows a third matching state of "X" or "Don't Care" for one or more bits in the stored dataword.
- **Example #1, crossword:**

							Match	
D	**W**	**?**	**?**	**?**	**?**	**?**		**Search term**
D	R	U	N	K	E	N	-	
D	W	A	R	F			*	
D	W	E	L	L			*	
D	W	I	N	D	L	E	*	

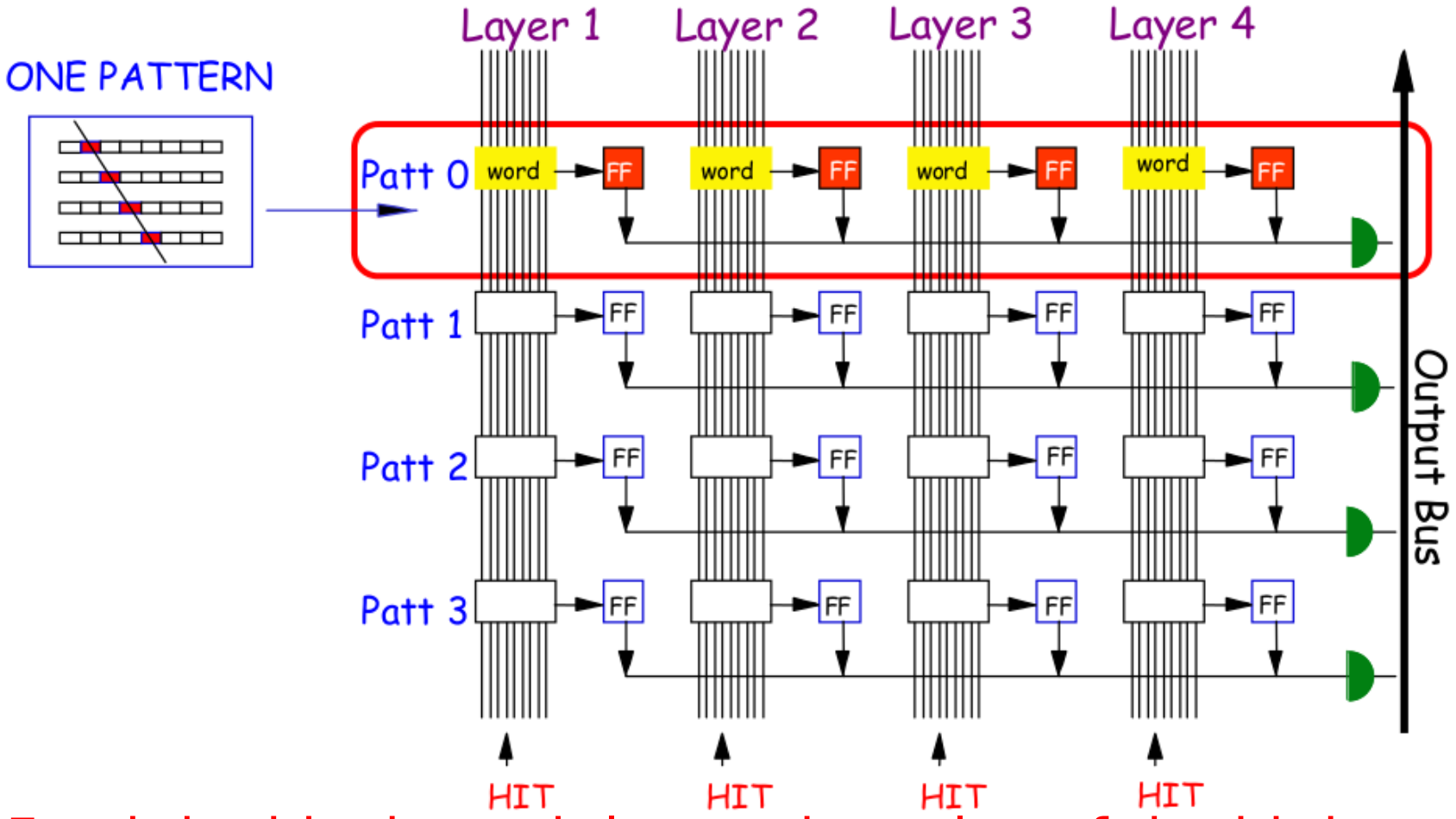
- **Example #2**, a ternary CAM might have a stored word of "**10XX0**" ("the pattern")

  This will match any of the 4 search words ("the data"):
  - "10000", "10010", "10100", or "10110".

  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*.

AM chip working principle

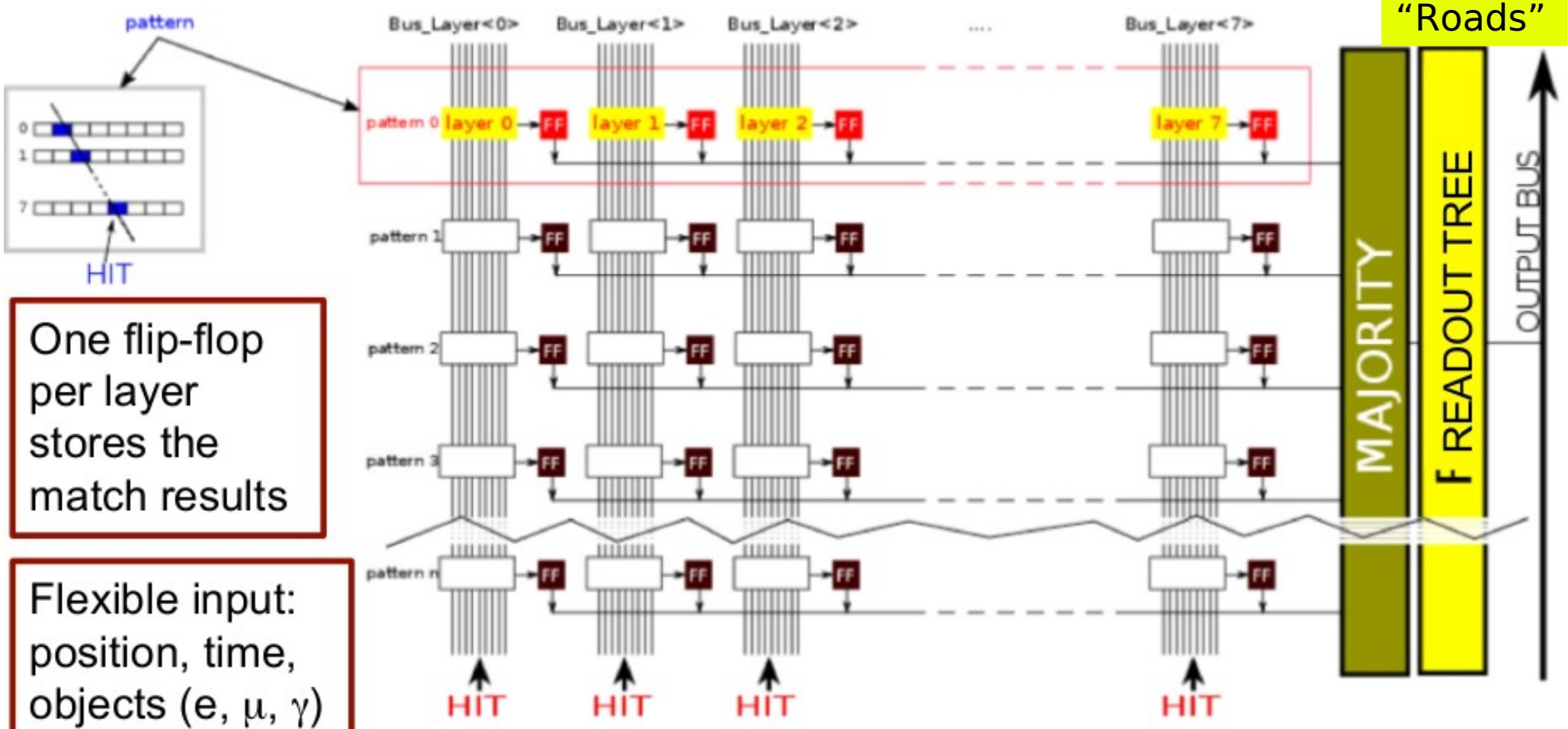
1. Load the patterns: each active bin in each layer is a number



2. Feed the hits in each layer: the value of the hit is seen all along the data-line by all the patterns in the layer:
   → Flags raised if matching found.
   → AND all flags to get a complete pattern matching.

Track trigger w/ pattern matching AM

Result: Matched Patterns "Roads"

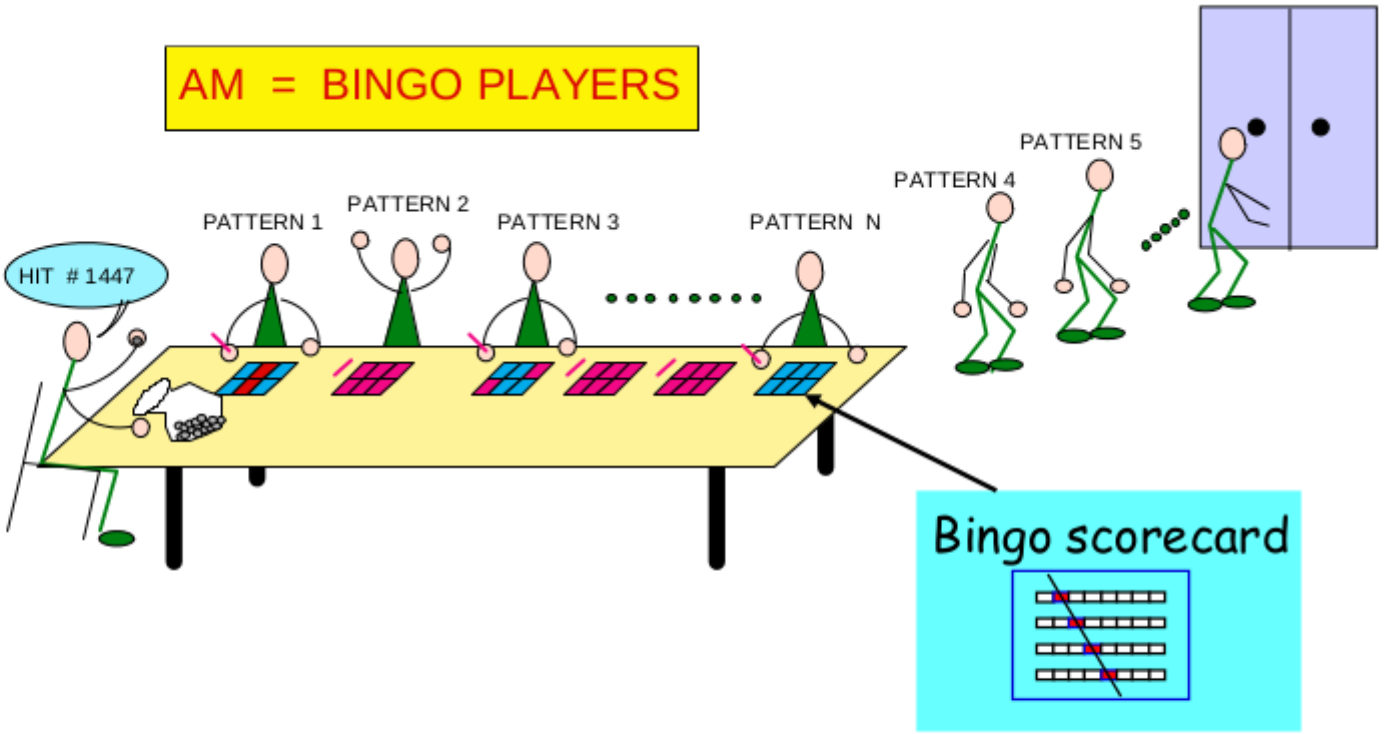


One flip-flop per layer stores the match results

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.

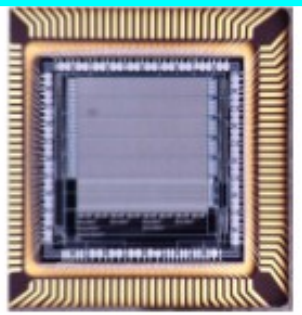
AM = hits compared to pre-computed patterns



- **Dedicated device: maximum parallelism**
- **Each pattern with private comparator**
- **Track search during detector readout**

AM evolution: mainly ASICs

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

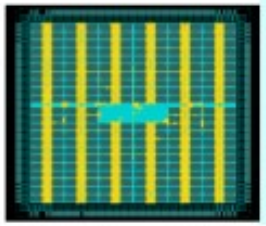
P. Giannetti et al., Nucl. Intsr. and Meth., vol. A413/2-3, **(1998)**

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**

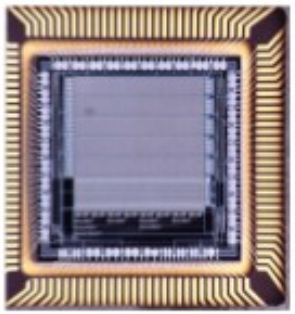
FTK R&D



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

AM evolution: mainly ASICs

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

FTK R&D

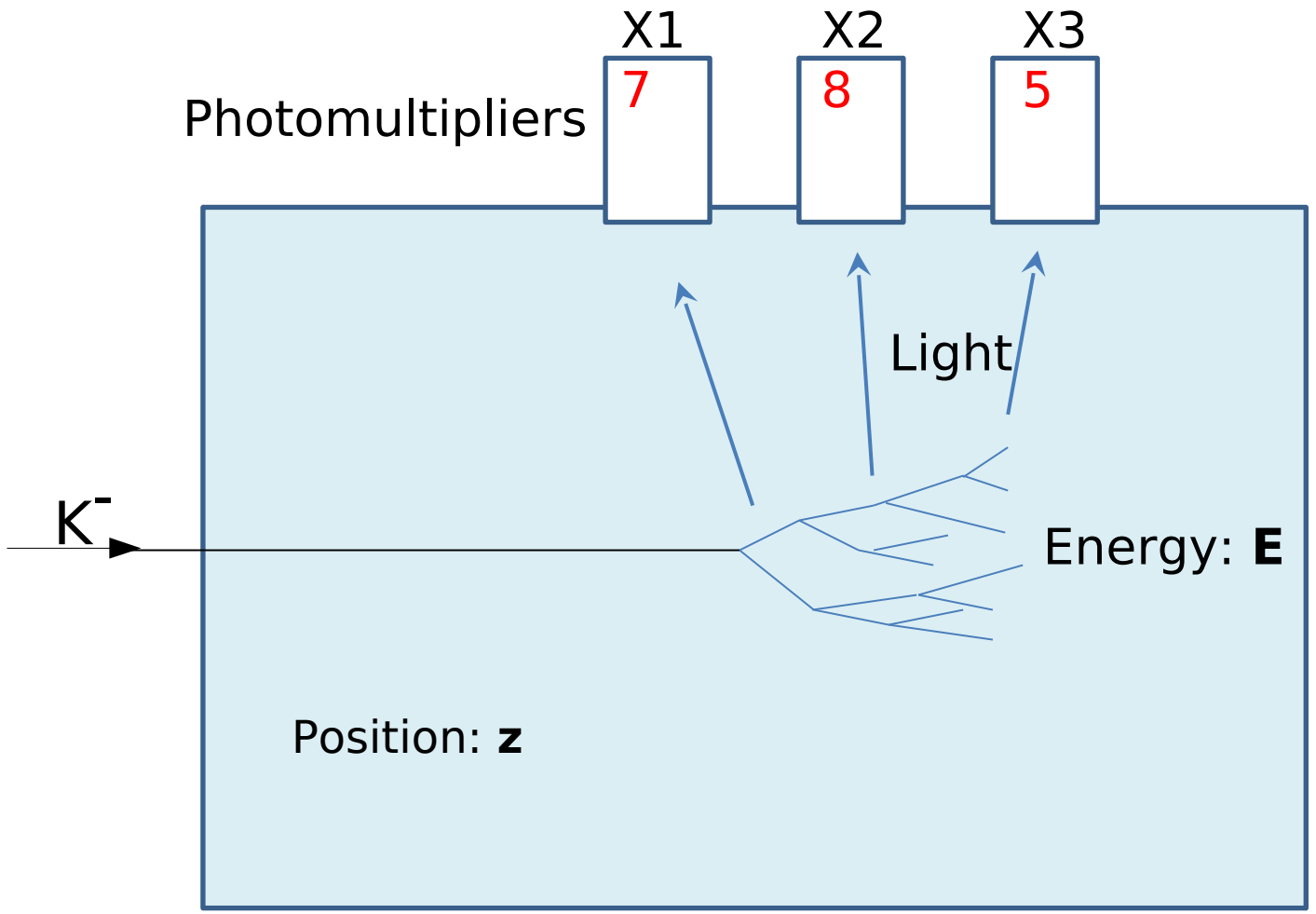
AMchip04

x12.

JINST **8 C01040,** doi:10.1088/1748-0221/8/01/C01040

FTK R&D in progress:
AMchip05 prototype: switched to serialized IO (11*2Gbs)
AMchip06: the FTK AM chip with 128k patterns/chip

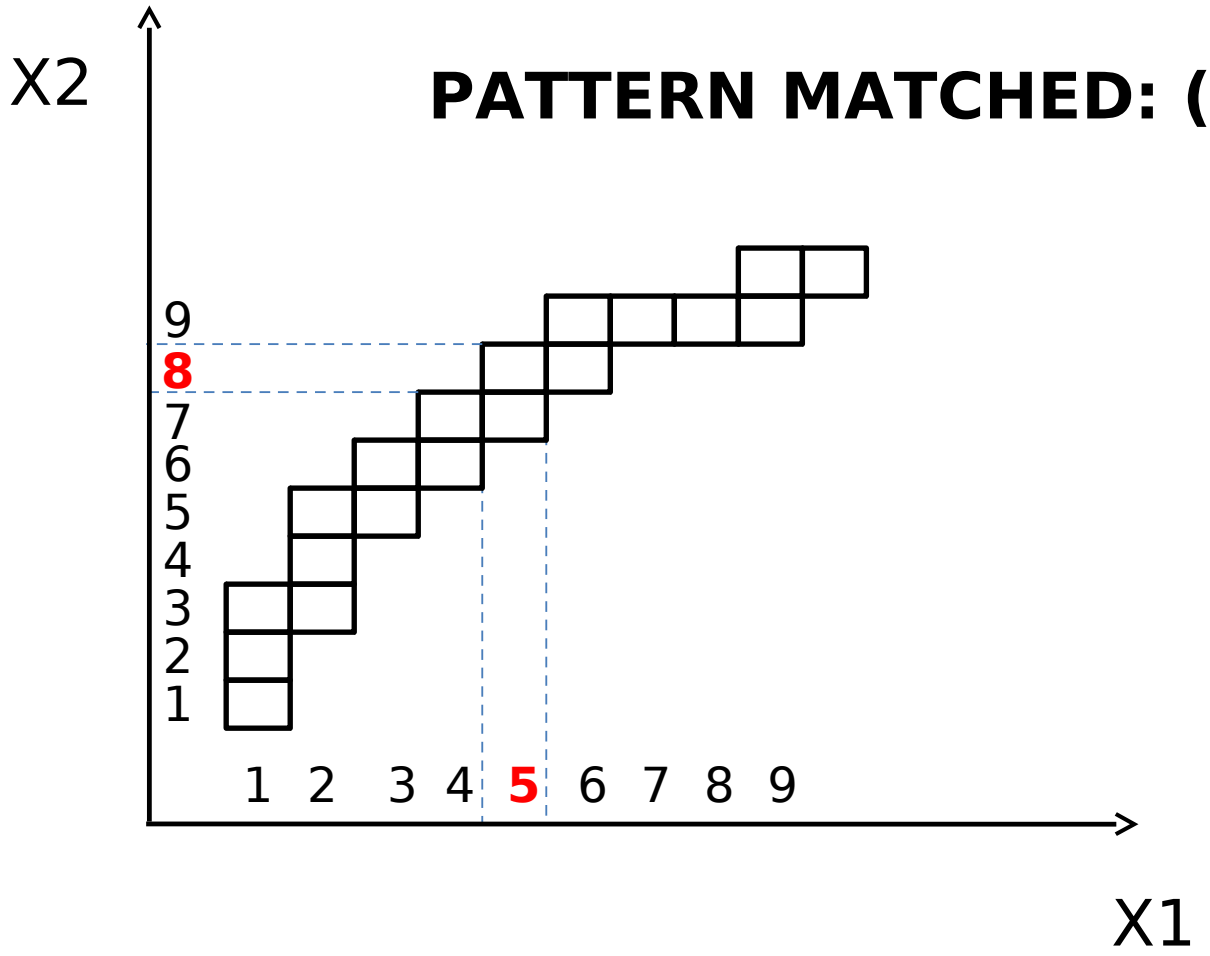
Pattern matching not restricted to trackers. Applies everywhere there is correlated behaviour. e.g:



These is a pattern for the correlated values in X1 and X2: a trace

Associate: X1 = 5 with X2 = 8

PATTERN MATCHED: (5,8)

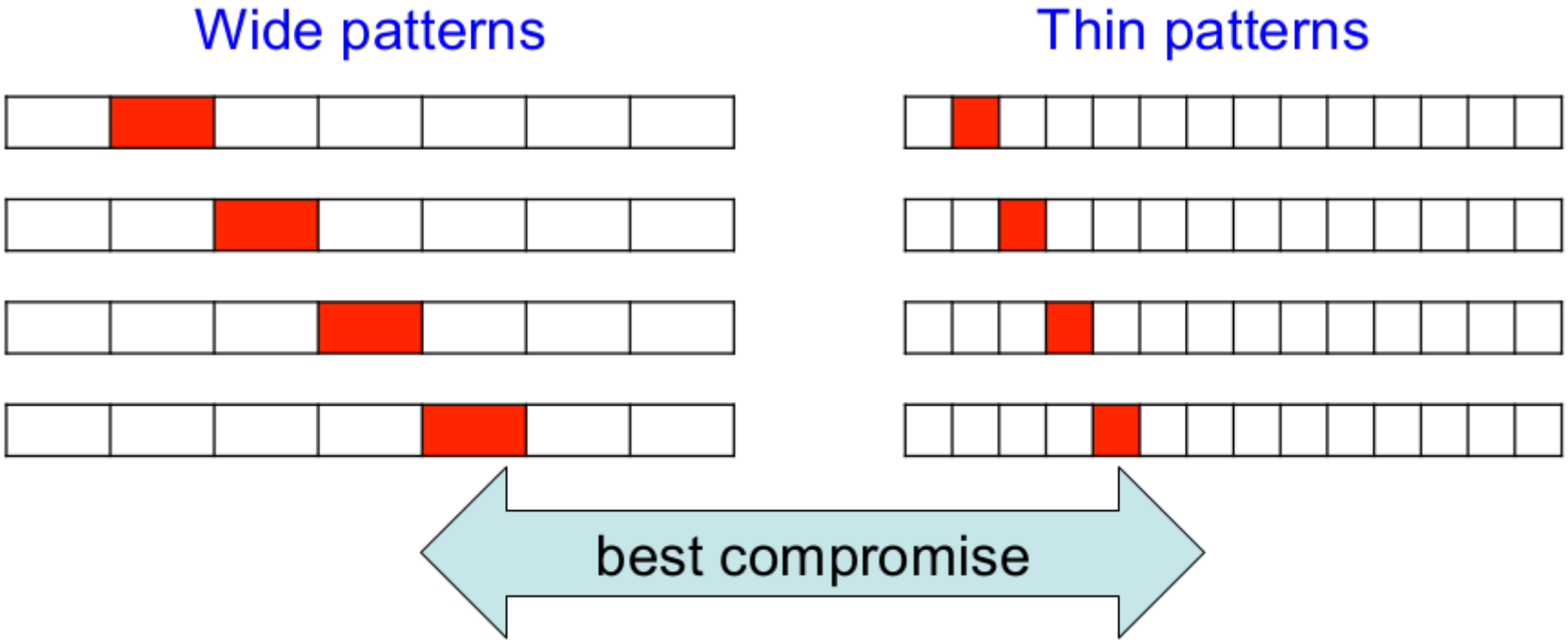


2b.

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 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

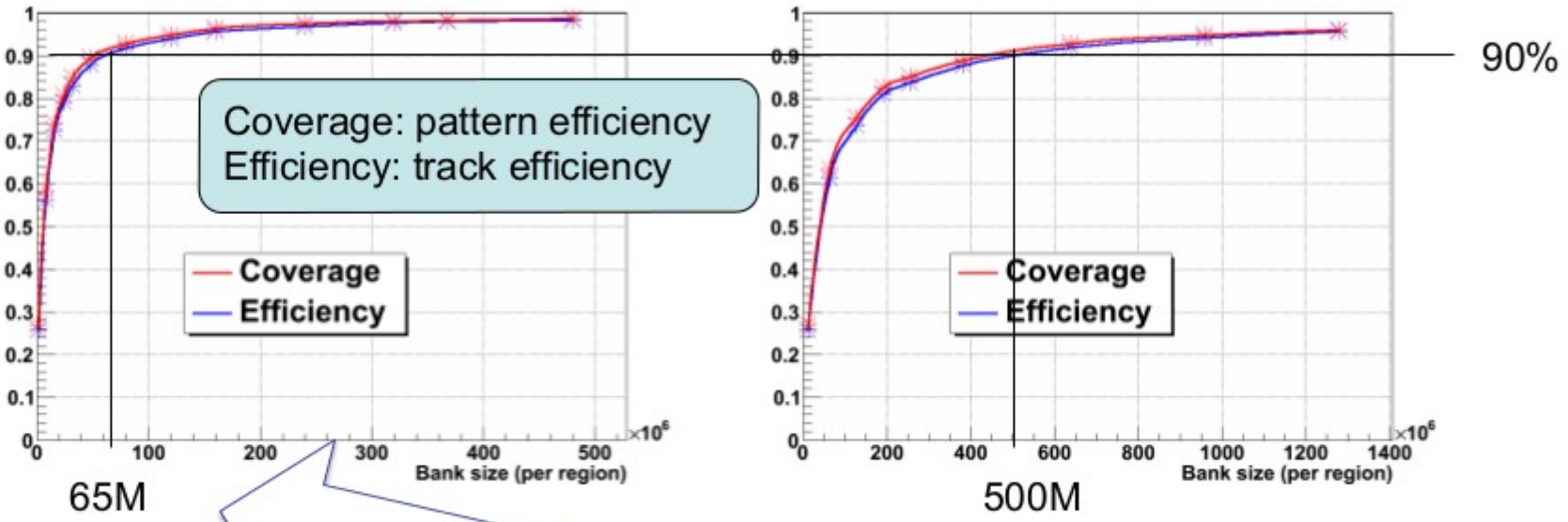
Recall: the number of patterns **N_p**, with **m** layers, of **n** bins each, is $N_p \simeq (m-1)n^2$

High efficiency with large bins (small number of patterns) → but, lots of fake tracks found → lots of work for detailed track fitting!
We prefer “few pattterns”-”few fakes” scenario

ATL-UPGRADE-PROC-2011-004
doi:10.1109/ANIMMA.2011.6172856

Pattern size
r-ϕ: 24 pixels, 20 SCT strips
z: 36 pixels

Pattern size (half size)
r-ϕ: 12 pixels, 10 SCT strips
z: 36 pixels



# of patterns in AM chips (barrel only, 45 ϕ degress)

<# matched patterns/event @ 3E34> = 342k

<# matched patterns/event @ 3E34> = 40k

roads (large fake fraction) represents the workload for the track fitter

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

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "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

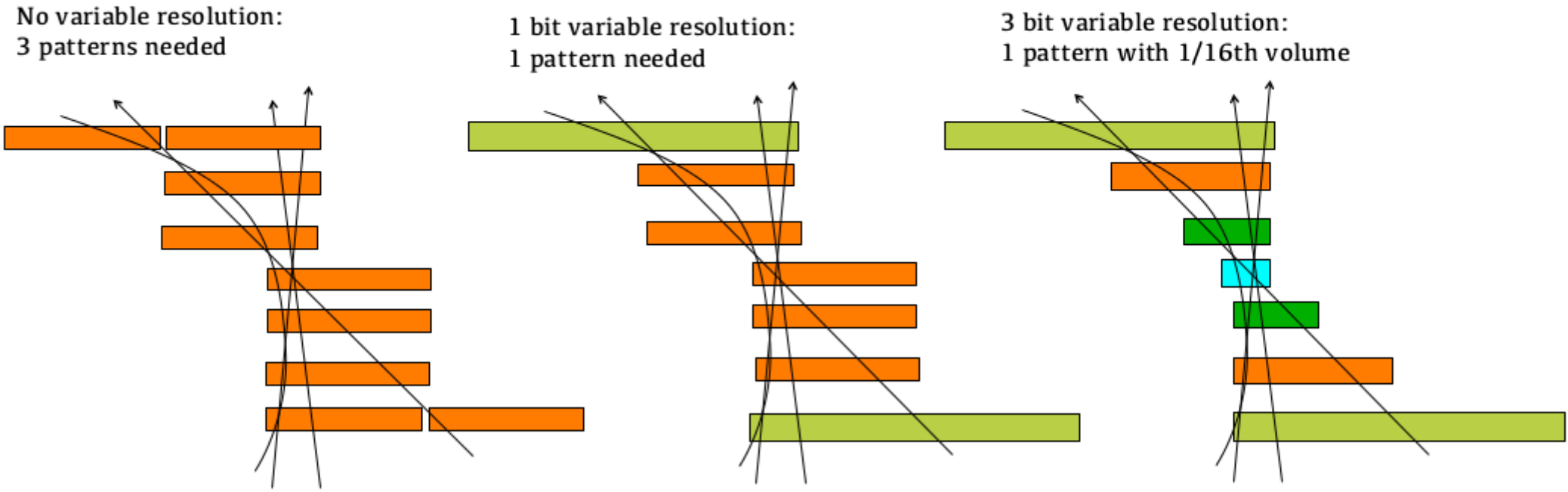
Pixels:

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)
  - Reduces the number of patterns and fake matches

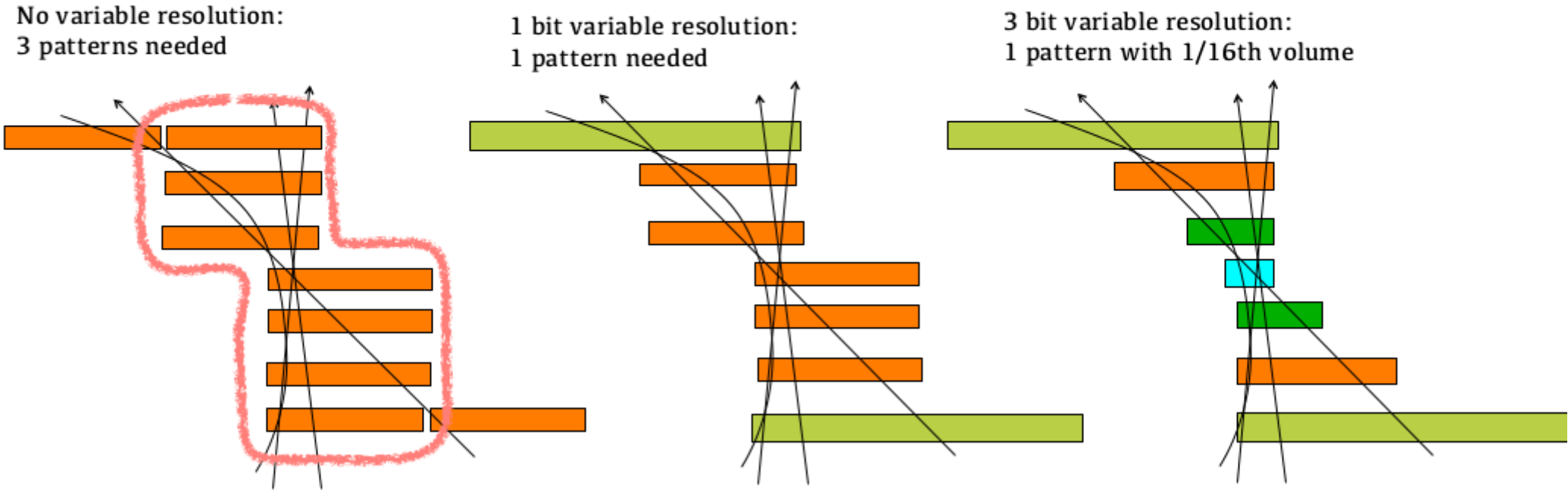


- Number of don't care bits set on a layer by layer, pattern by pattern basis

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

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)
  - Reduces the number of patterns and fake matches

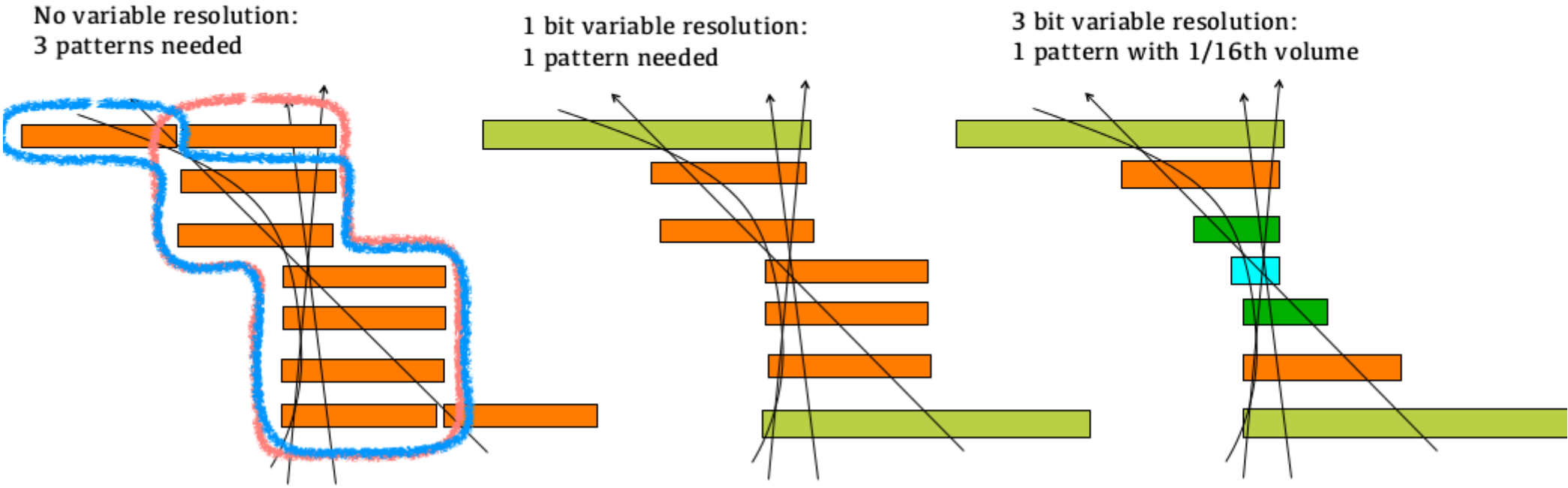


- Number of don't care bits set on a layer by layer, pattern by pattern basis

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

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)
  - Reduces the number of patterns and fake matches



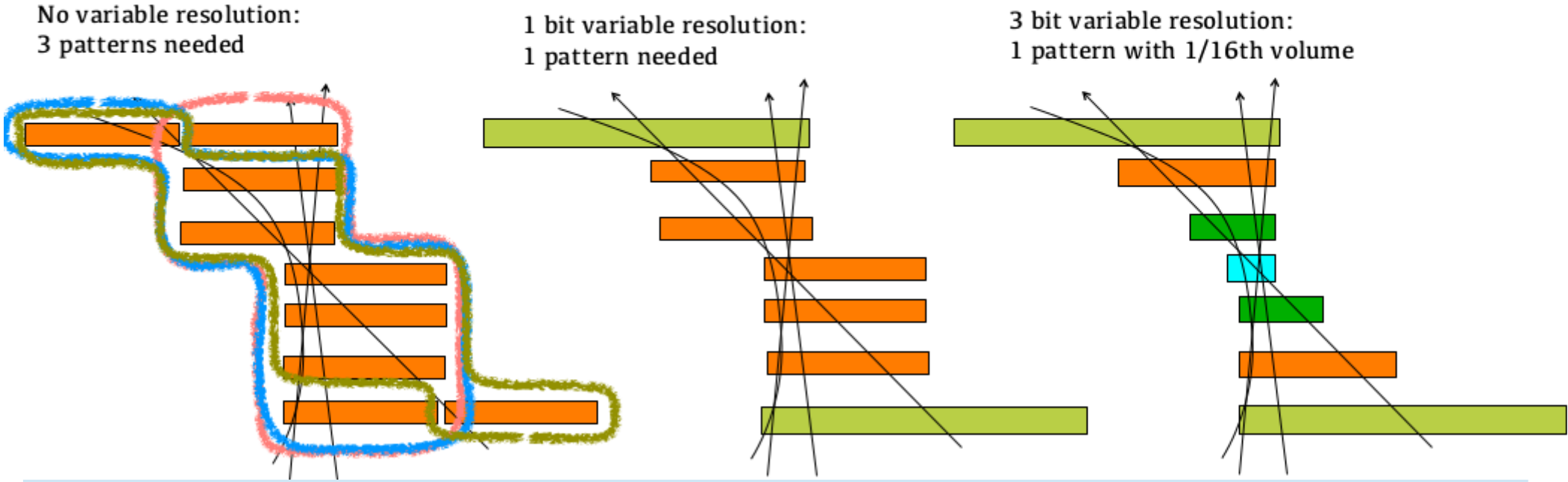
- Number of don't care bits set on a layer by layer, pattern by pattern basis

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

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!



This technique
can be exploited by any coincidence based trigger!

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

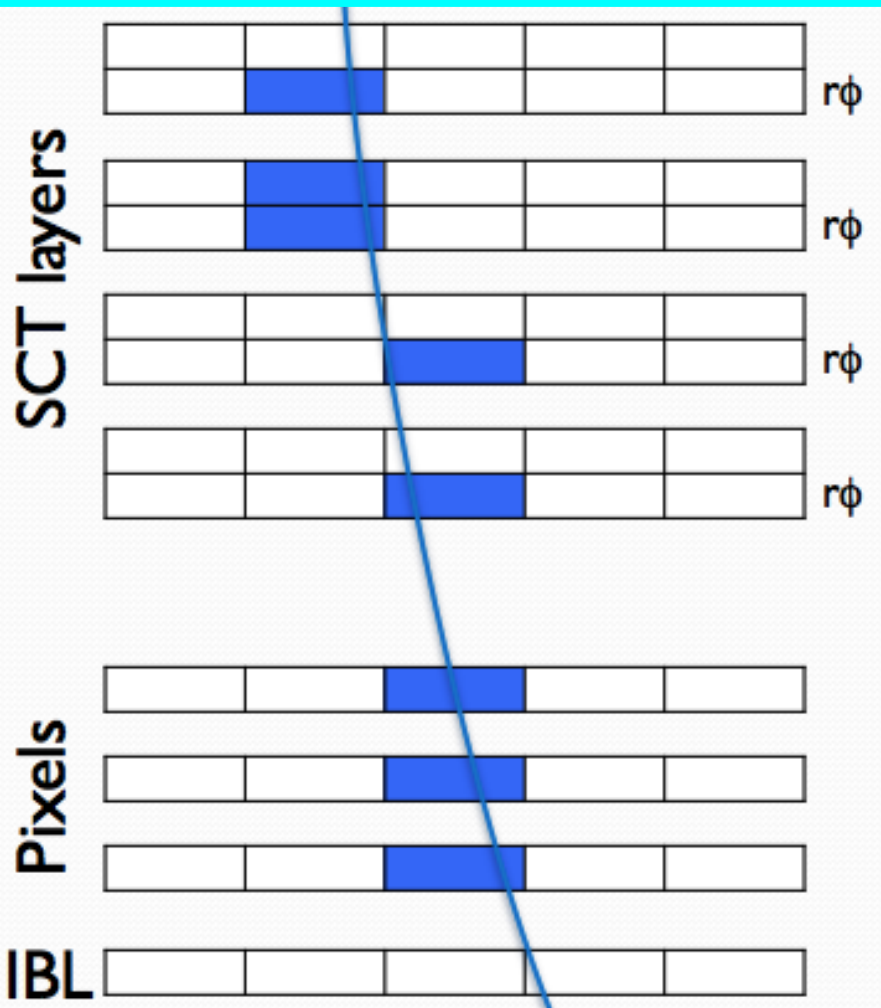
3.

Now that we have found some matching patterns

each matching pattern defines a "road" where we go and fetch all the (few now) hits it contains, and we should fit them to a helical track to measure the track parameters precisely

Track fitting in FPGAs: 1st stage, 8 layers

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

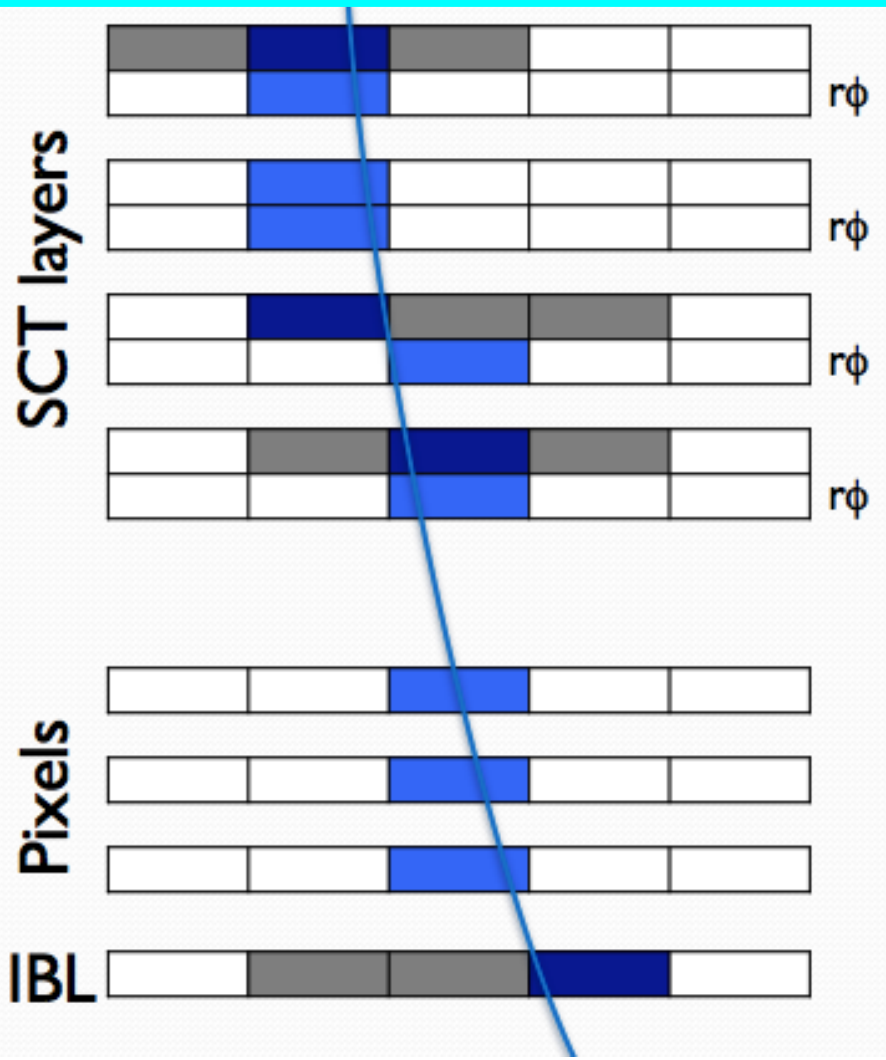


Track fitting in FPGAs w/ many Digital Signal Processors (DSPs)

BUT: Linear approximation: get a set of linear equations (instead of solving helix) → fast multiplications with pre-computed constants → ~**1 Gfits/s per FPGA**

Track fitting in FPGAs: 2nd stage, 12 layers

- 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

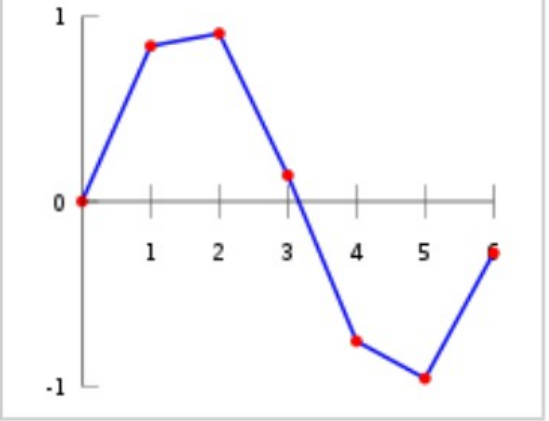


Done on FPGAs, on a “2nd stage” board

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

- E.g., in a computer if normal math calls are time-expensive:
- sin(x) = Taylor expansion gives a polynomial to calculate
- Or, use Look-Up Tables (LUTs = precalculated values stored in tables) and interpolate between the neighbouring sin(x) to get the value 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

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

Idea and implementation of Don't Care bits for variable resolution, by Alberto Annovi
See "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

The point to take home:

- Split the problem in a fast (coarse) one, and a refined one working with much reduced data.

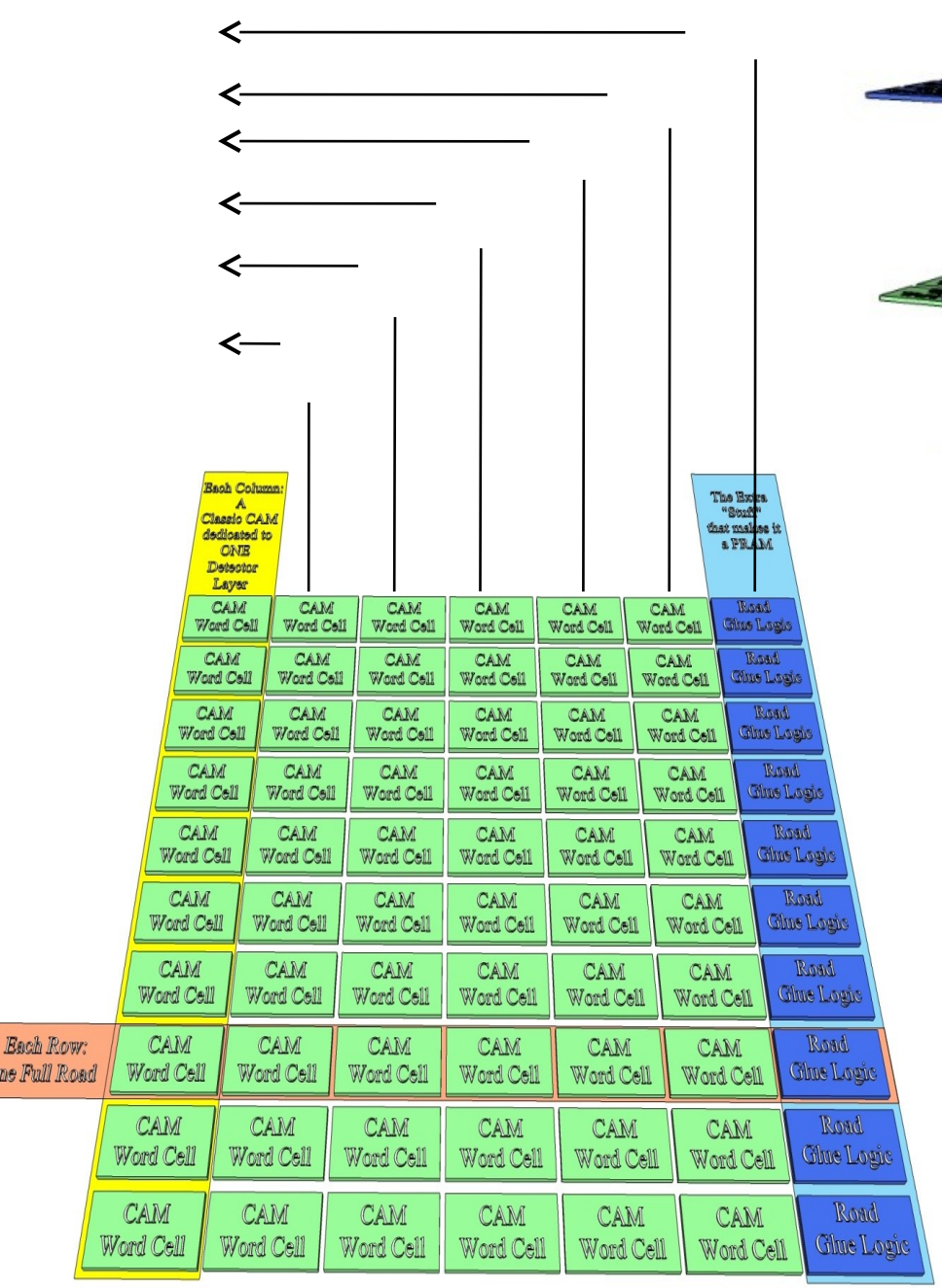
  (you know now that we do this all the time in the trigger)

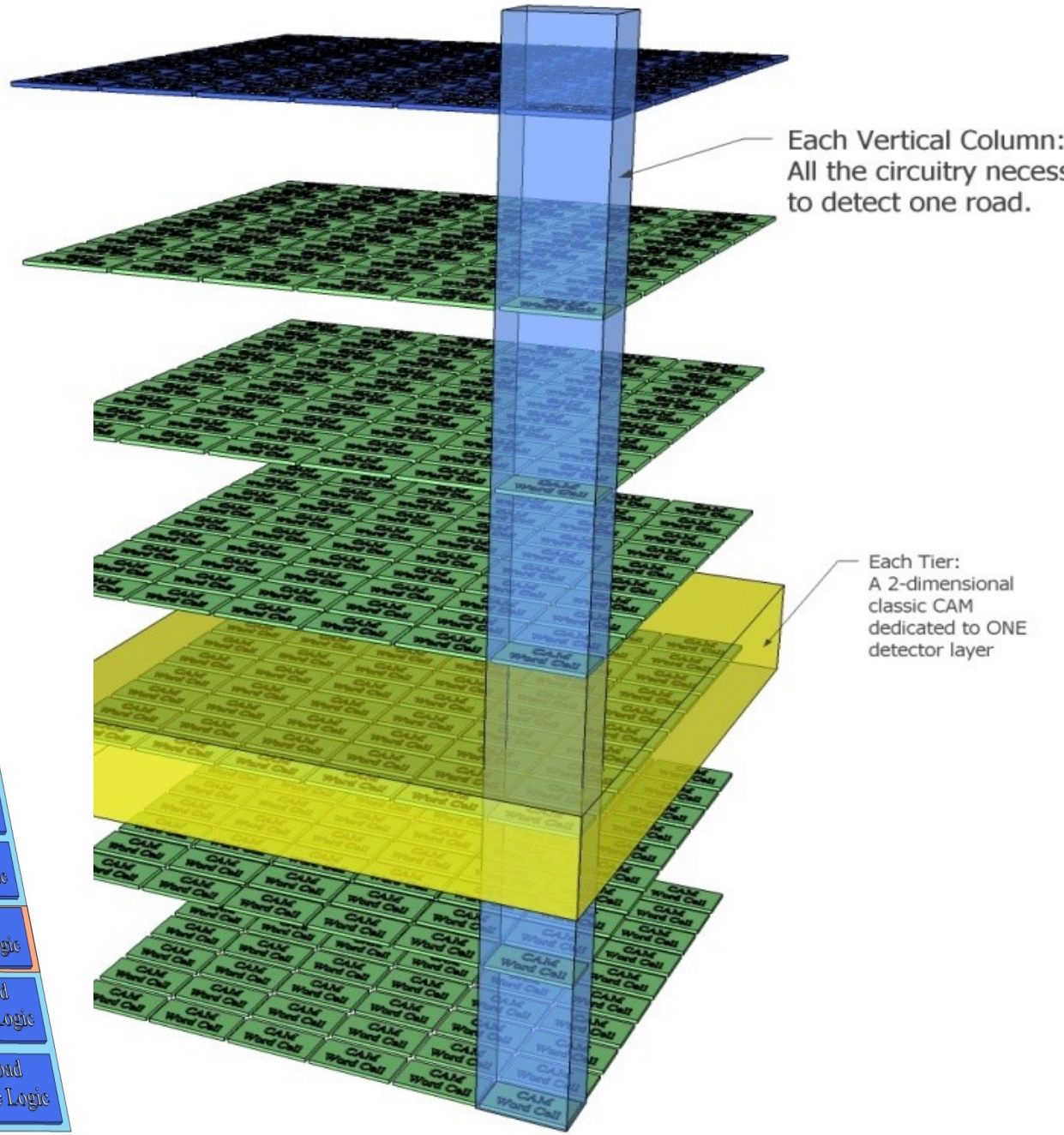
- Use pre-calculated patterns & values wherever you can: if you get the desired precision, you gain a lot in time

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

- We saw the example of the Fast TracKer upgrade in ATLAS, using
  - AM-based pattern recognition with ASICs,
  - refined track-fitting (and almost everything else needed, from smart databases, to I/O) in powerfull modern FPGAs

...and to keep an eye in what new technology can do for you: From 2D to 3D

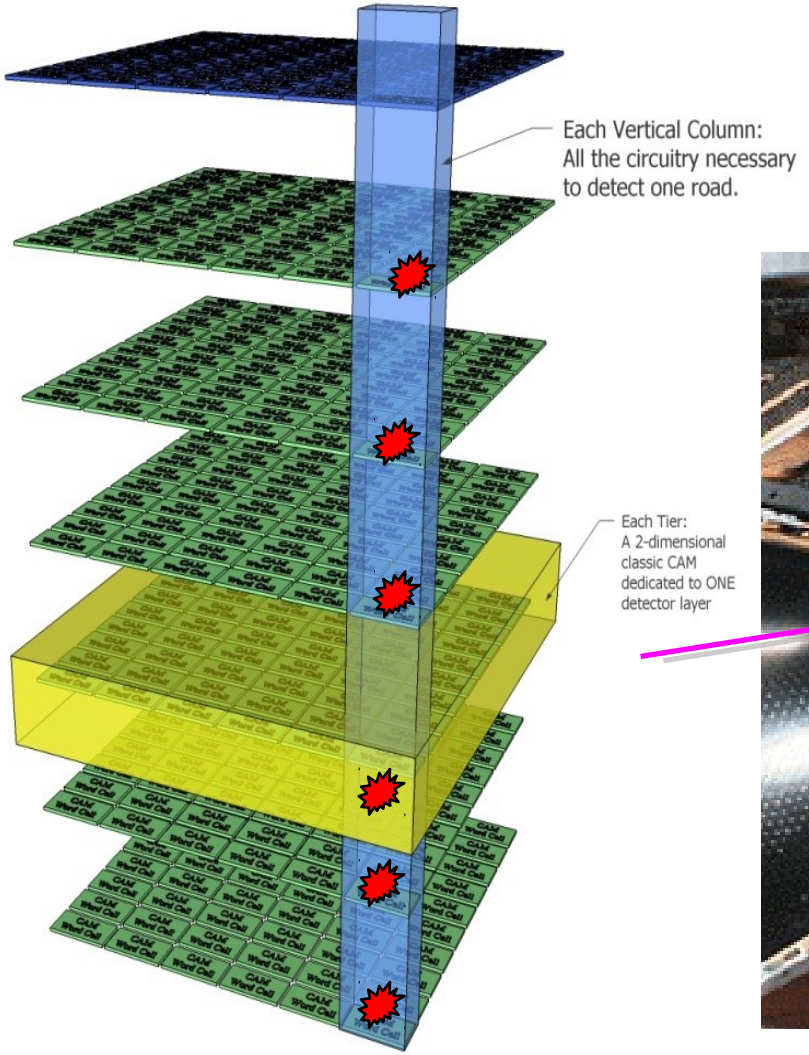




VIPRAM

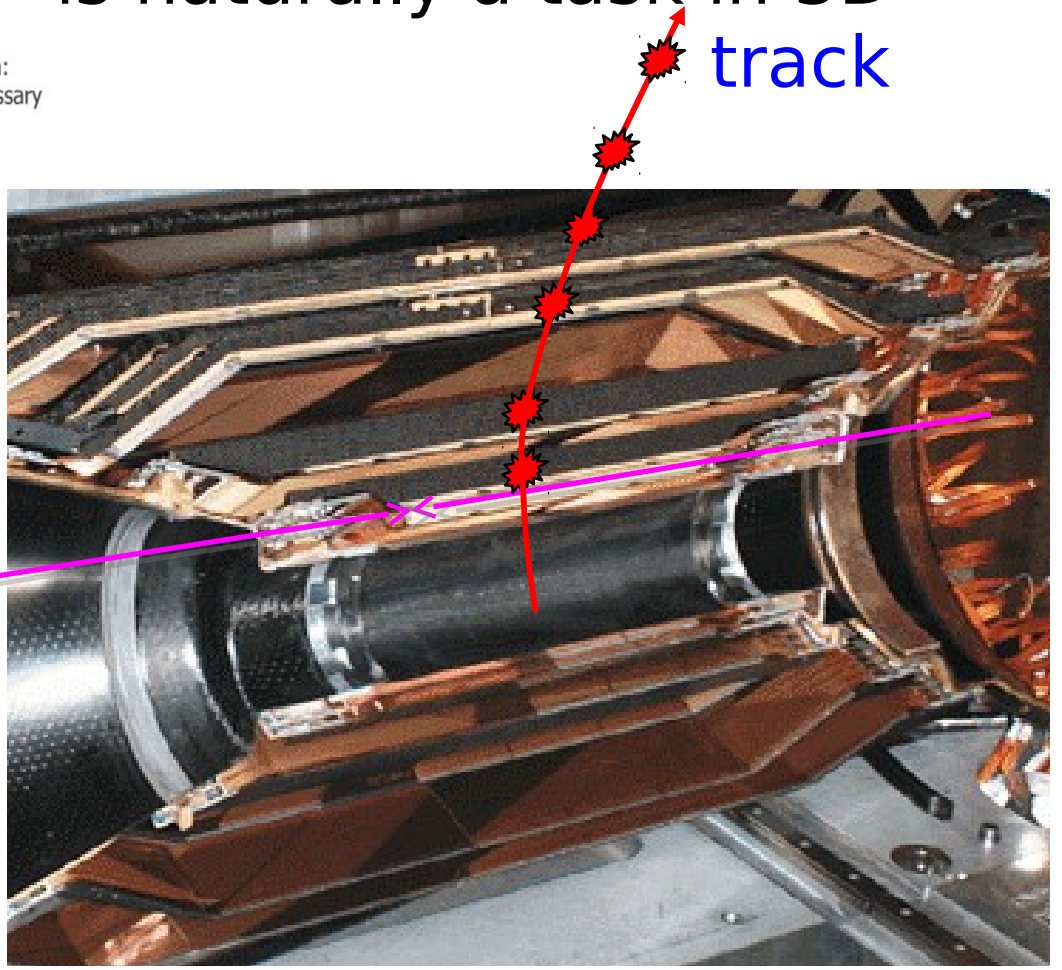
(Vertically Integrated Pattern Recognition Associative Memory)

road



Pattern recognition for tracking is naturally a task in 3D

track



VIPRAM concept (developed at Fermilab):

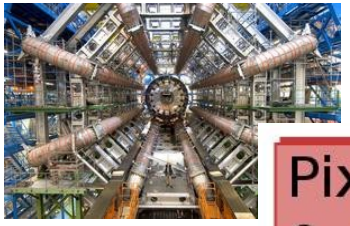
http://hep.uchicago.edu/~thliu/projects/VIPRAM/TIPP2011_VIPRAM_Paper.V11.preprint.pdf

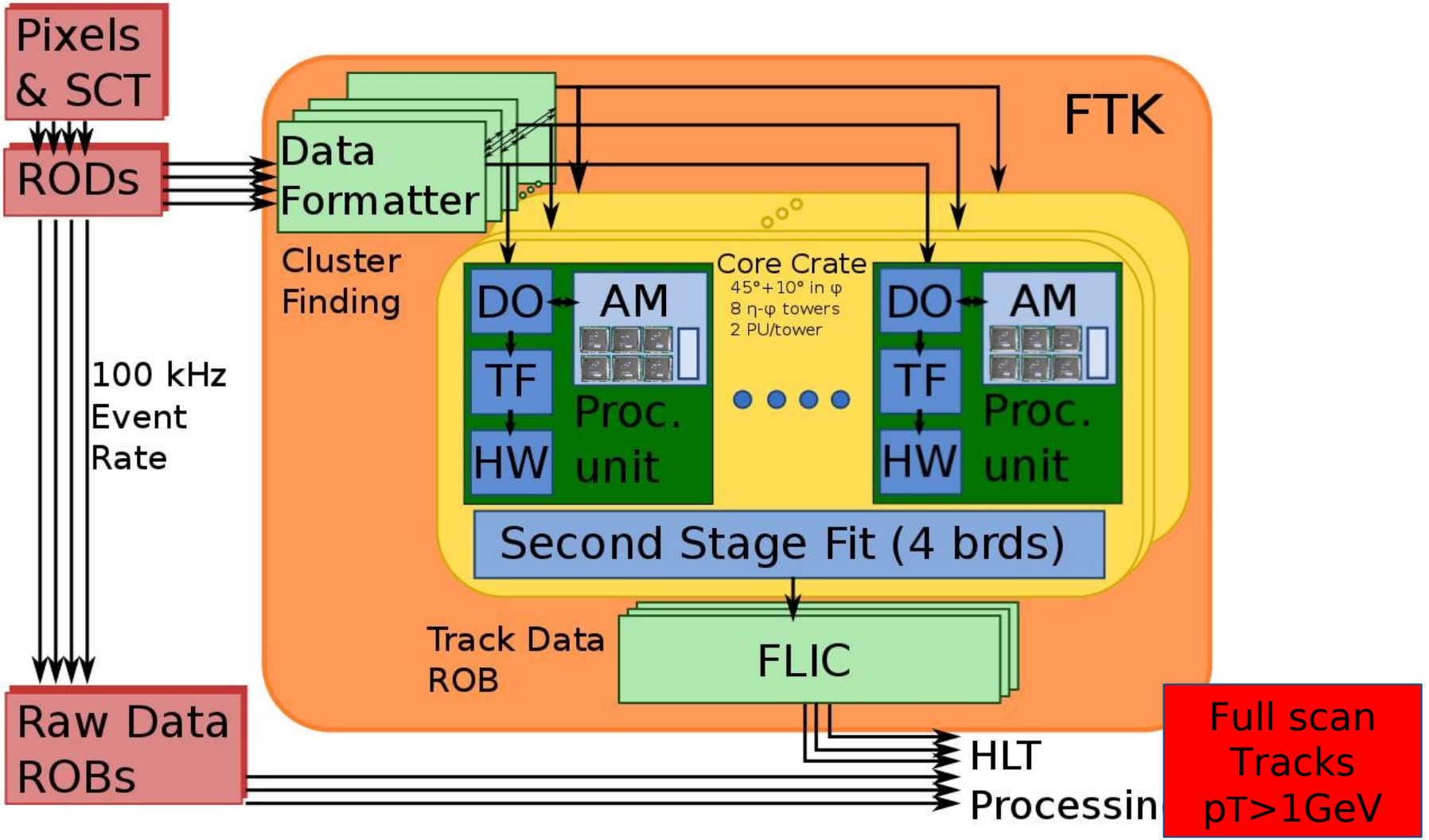
Thank you

Hope you have enjoyed the school,
learned a lot,
and got triggered (and intrigued) to
look up things on your own now

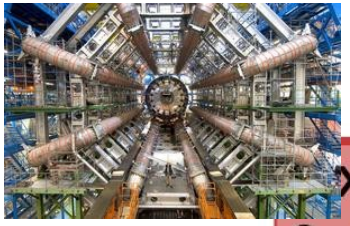
4. FTK: the system components

The FTK system – components & relation to rest of TDAQ



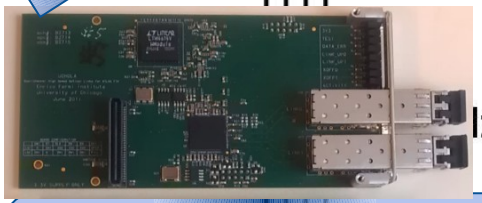


Data in FTK from Si RODs in dual-output card



xels & SCT

RODs



lz

Sends SCT & pixel data to DAQ & FTK.

Raw Data ROBs

FTK

Data Formatter

Cluster Finding

Core Crate

45°+10° in φ

8 η-φ towers

2 PU/tower

DO AM TF HW Proc. unit

DO AM TF HW Proc. unit

Second Stage Fit (4 brds)

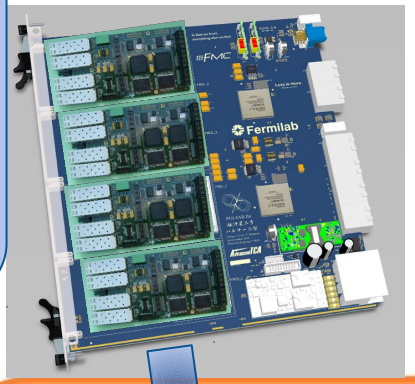
Track Data ROB

FLIC

HLT Processin

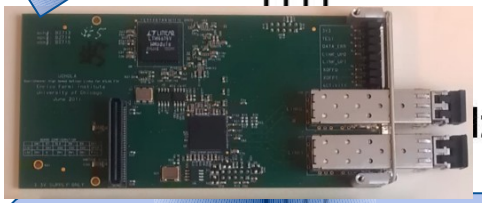
Full scan Tracks pT>1GeV

4 FTK_IM/DF do cluster finding. ATCA DF distributes clusters to 64 FTK η–ϕ towers.



Pixels & SCT

RODs



Sends SCT & pixel data to DAQ & FTK.

Raw Data ROBs

FTK

Data Formatter

Cluster Finding

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

DO AM TF HW Proc. unit

DO AM TF HW Proc. unit

Second Stage Fit (4 brds)

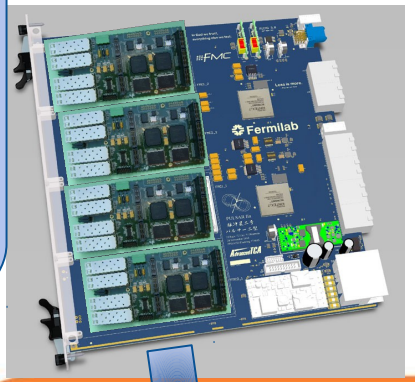
Track Data ROB

FLIC

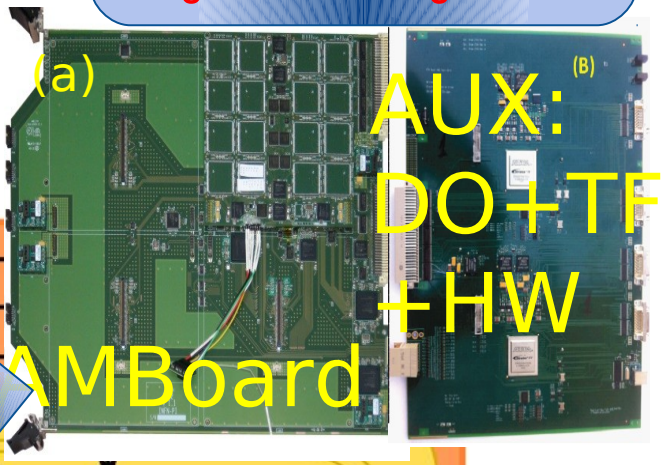
HLT Processin

Full scan Tracks pT>1GeV

4 FTK_IM/DF do cluster finding. ATCA DF distributes clusters to 64 FTK η–ϕ towers.



128 PUs do pattern matching and the 1st stage track fitting.



Pixels & SCT

RODs

Data Formatter

Cluster Finding

DO AM TF HW Proc. unit

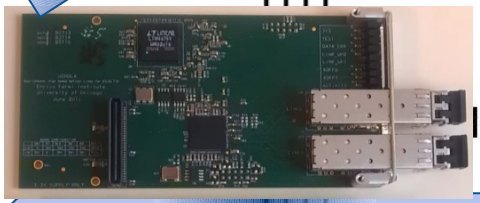
Core Crate

DO AM TF HW Proc. unit

Second Stage Fit (4 brds)

Track Data ROB

FLIC



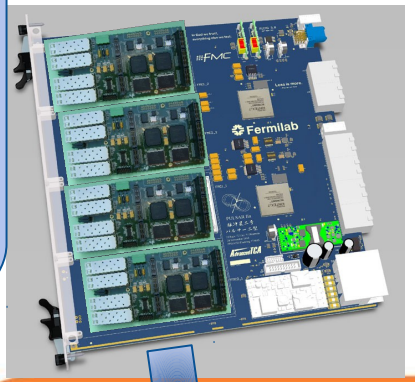
Sends SCT & pixel data to DAQ & FTK.

Raw Data ROBs

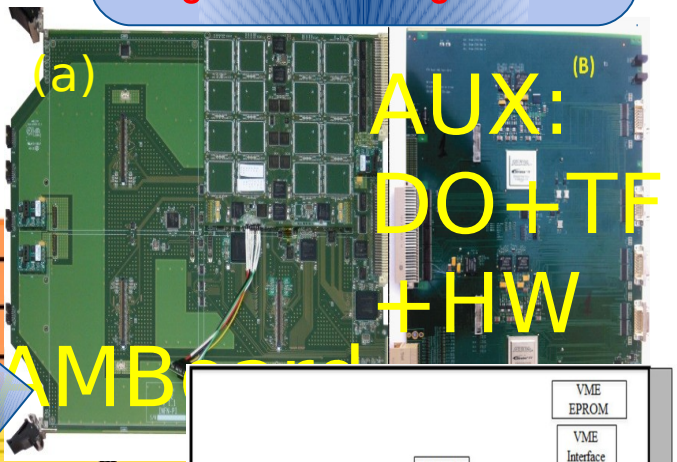
HLT Processin

Full scan Tracks pT>1GeV

4 FTK_IM/DF do cluster finding. ATCA DF distributes clusters to 64 FTK η–ϕ towers.



128 PUs do pattern matching and the 1st stage track fitting.



AUX: DO+TF +HW

Pixels & SCT

RODs

Data Formatter

Cluster Finding

DO

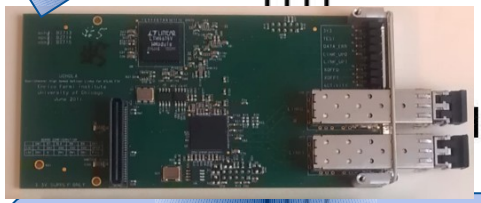
AM

TF

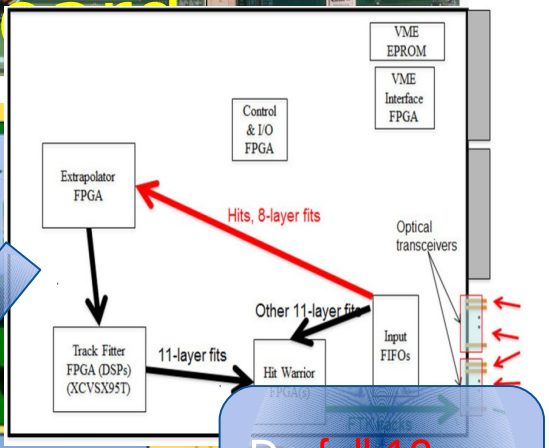
Proc. unit

HW

Core Crate



Sends SCT & pixel data to DAQ & FTK.



Second Stage Fit (4 brds)

Do full 12 layer fit

Track Data ROB

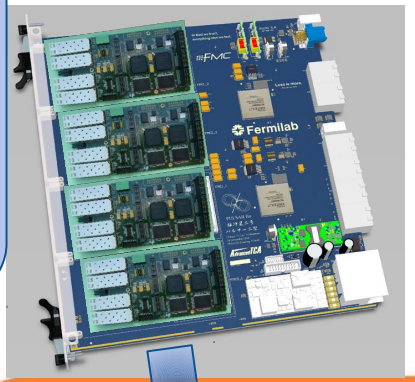
FLIC

Raw Data ROBs

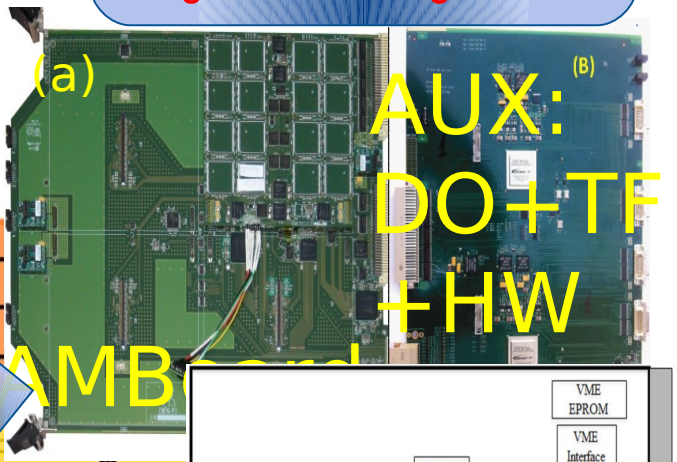
HLT Processin

Full scan Tracks pT>1GeV

4 FTK_IM/DF do cluster finding. ATCA DF distributes clusters to 64 FTK η–ϕ towers.



128 PUs do pattern matching and the 1st stage track fitting.



AUX: DO+TF +HW

Pixels & SCT

RODs

Data Formatter

Cluster Finding

DO

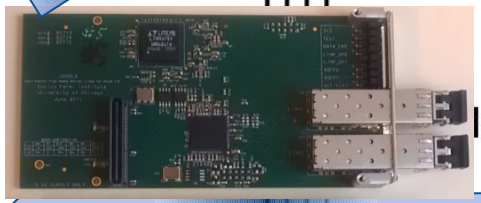
AM

TF

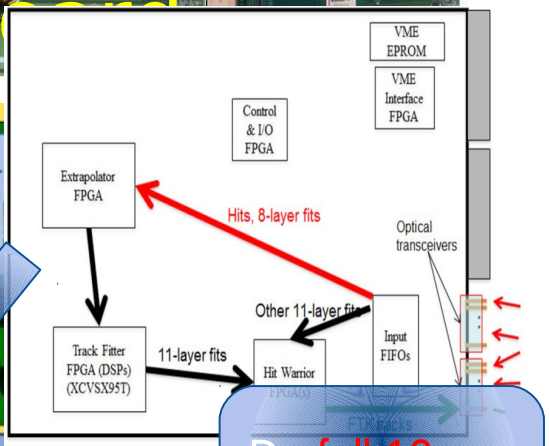
Proc. unit

HW

Core Crate

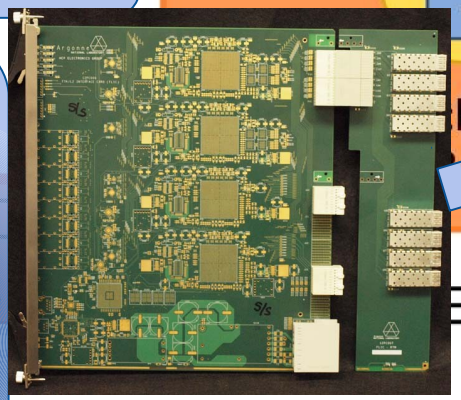


Sends SCT & pixel data to DAQ & FTK.



Do full 12 layer fit

Second Stage Fit (4 brds)



FLIC sends tracks to ROS's. ATCA for global function TBD

FLIC

HLT Processin

Full scan Tracks pT>1GeV

5. Timeline

Start partial installation by mid 2015,
cover barrel region,
then add units to cover end-cap and
accommodate more patterns,
needed for higher Lumi