

Intelligent triggering: pattern recognition with Associative Memories and other tools (FPGAs) - the Fast Tracker example in ATLAS

Kostas Kordas

Aristotle University of Thessaloniki



ISOTDAQ2015 , Rio de Janeiro, January 28 - February 5, 2015

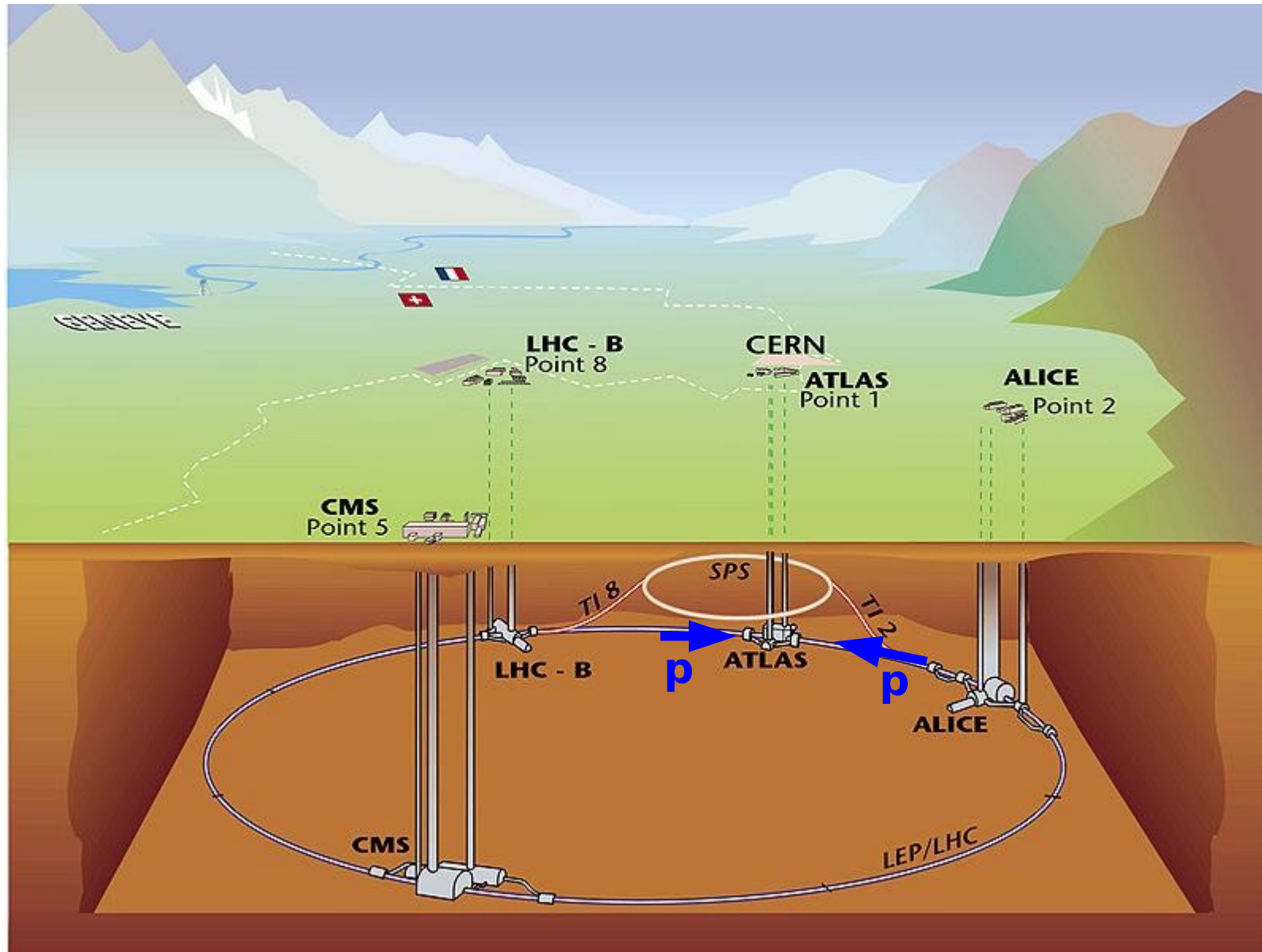
Overview

- Trigger and DAQ in HEP experiments
 - The need for tracking info at the Trigger and how to do it fast
- FTK (Fast Tracker): tracking in two stages
 - Pattern matching with Associative Memories
 - Refined fitting with FPGAs

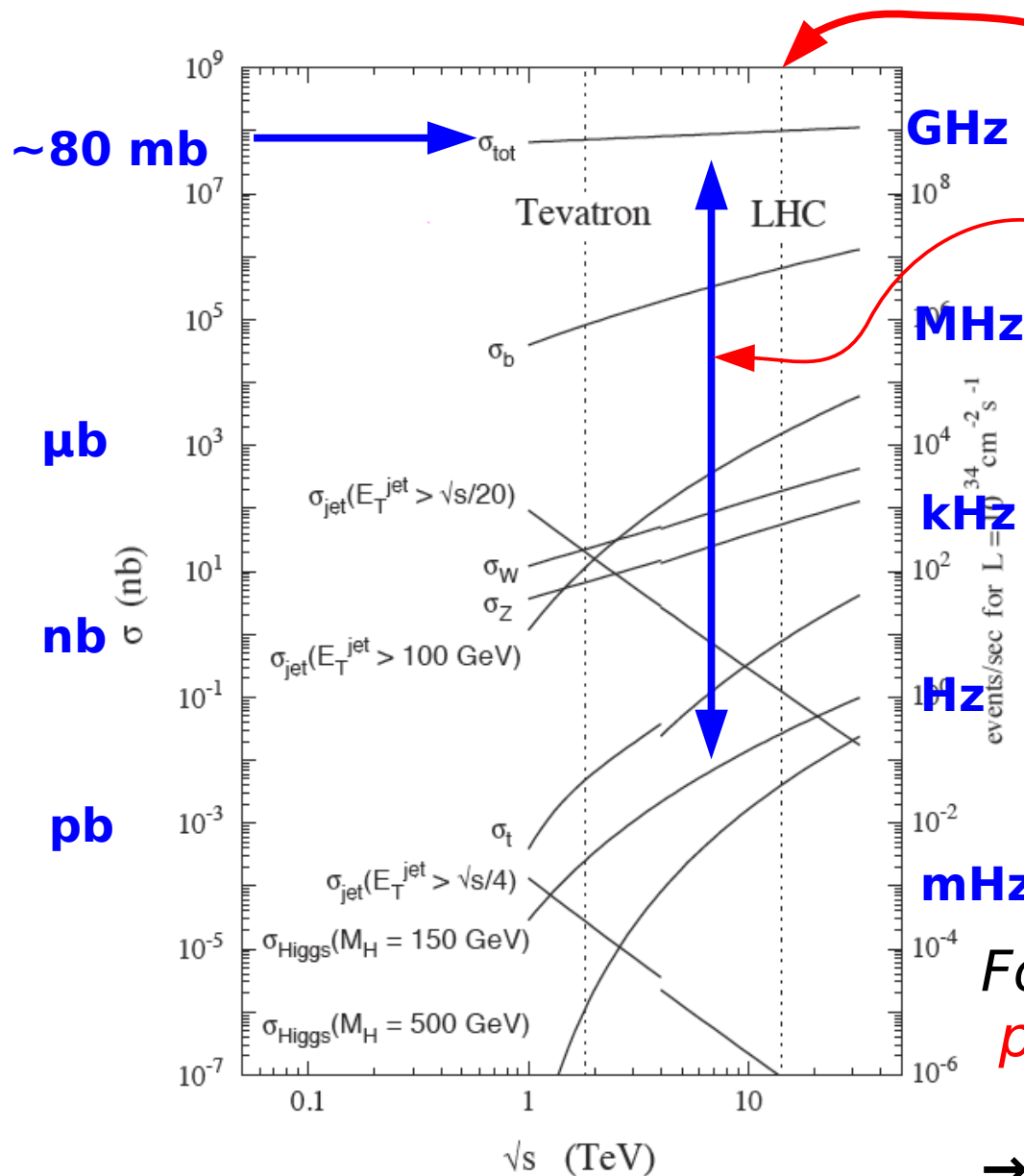
Basically you'll see that:

if you want to avoid or cannot afford calculating something time consuming: **split the problem and use pre-calculated patterns and quantities.**

Experiments at the LHC - pp collisions



The energy frontier - interested in rare processes



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

>10 orders of magnitude!
 (>12 orders when including branching ratios to leptons, e.g. $H \rightarrow ZZ \rightarrow \mu^+ \mu^- \mu^+ \mu^-$) !!!

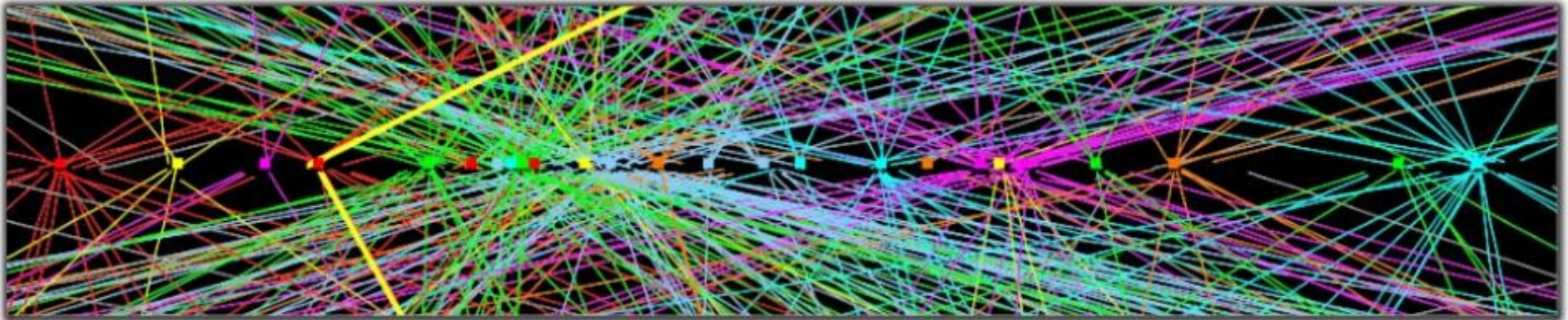
In order to have a reasonable number of observed events, we need high luminosity colliders

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

For 25ns bunch spacing:
pp interactions/bunch crossing:
 $\langle \mu \rangle \sim 25$

→ **soon: L = 3-5 times more!**

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.

~ 100 M channels in modern HEP experiments, recording signals from these, produces:

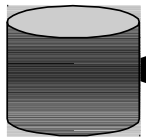
$\rightarrow 1.5$ MB of info per “event”, i.e., every time I read this info

- * The Trigger and Data Acquisition system, watches $O(1)$ billion pp collisions per second (40M proton bunch crossings / sec),
- * **select online “the most interesting” $O(1k)$ events/sec,**
- * **and log them for offline use with a resolution of a ~ 100 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.

Permanent storage



High Level Triggers: PC farms
~0.2s, ~1 kHz

L1: hardware
{2.5 μ s, 100 kHz}



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

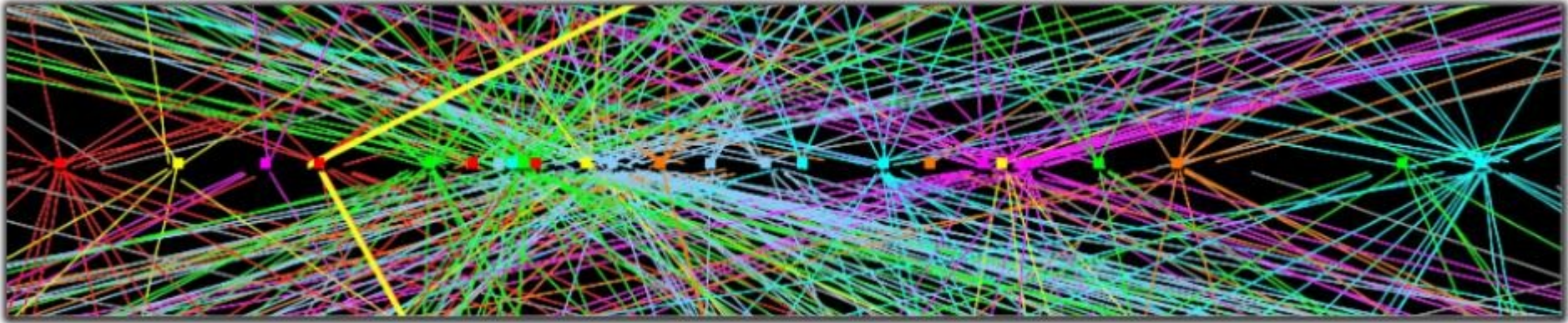
When there is limited time budget (L1 trigger): decide based only on the muon and calorimeter systems

There are channels though, where you need information from the inner tracker as early as possible in order 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

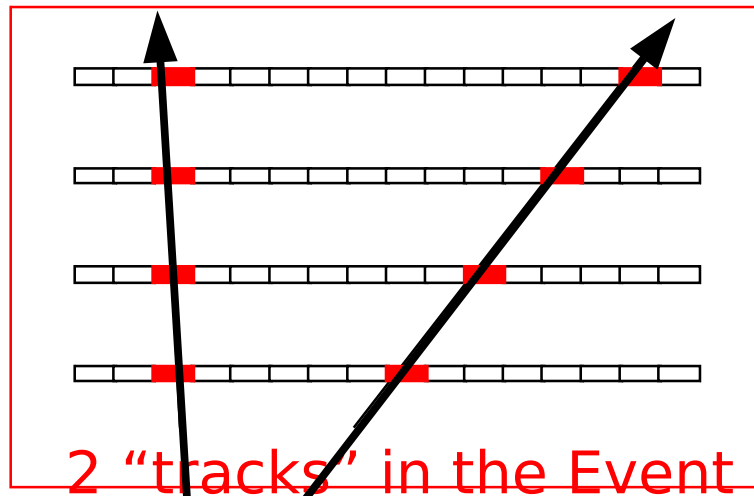
$$H \rightarrow b \bar{b} \quad \text{or} \quad H \rightarrow \tau^+ \tau^-$$

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.

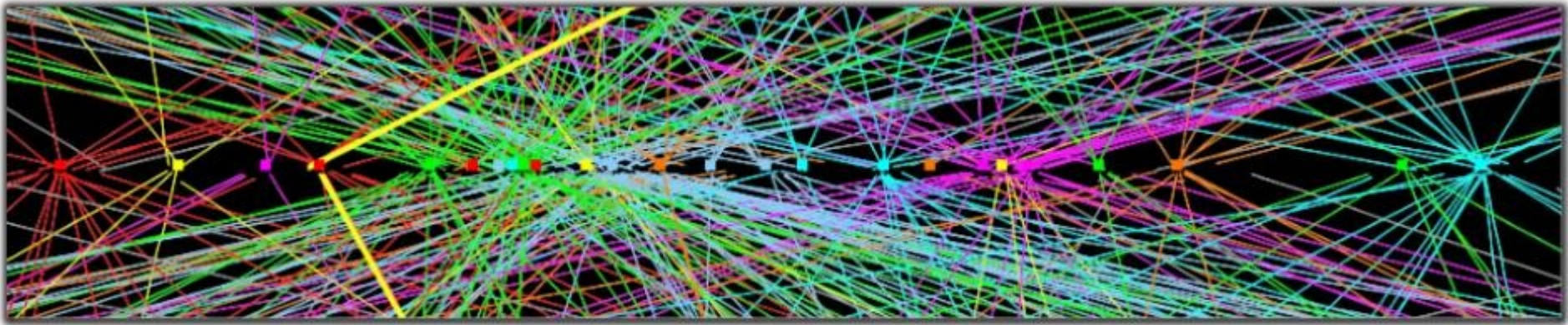
Each charged particle leaves a trace in the detector, a.k.a “a track”



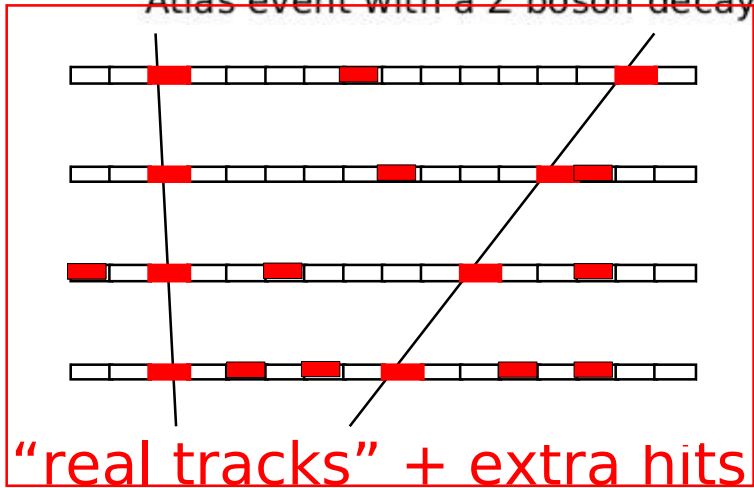
- Connecting the “hit” readout cells from one detection layer to the other
- traces the charged particle's path as it moves radially outward and its' position is measured in each detector layer



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

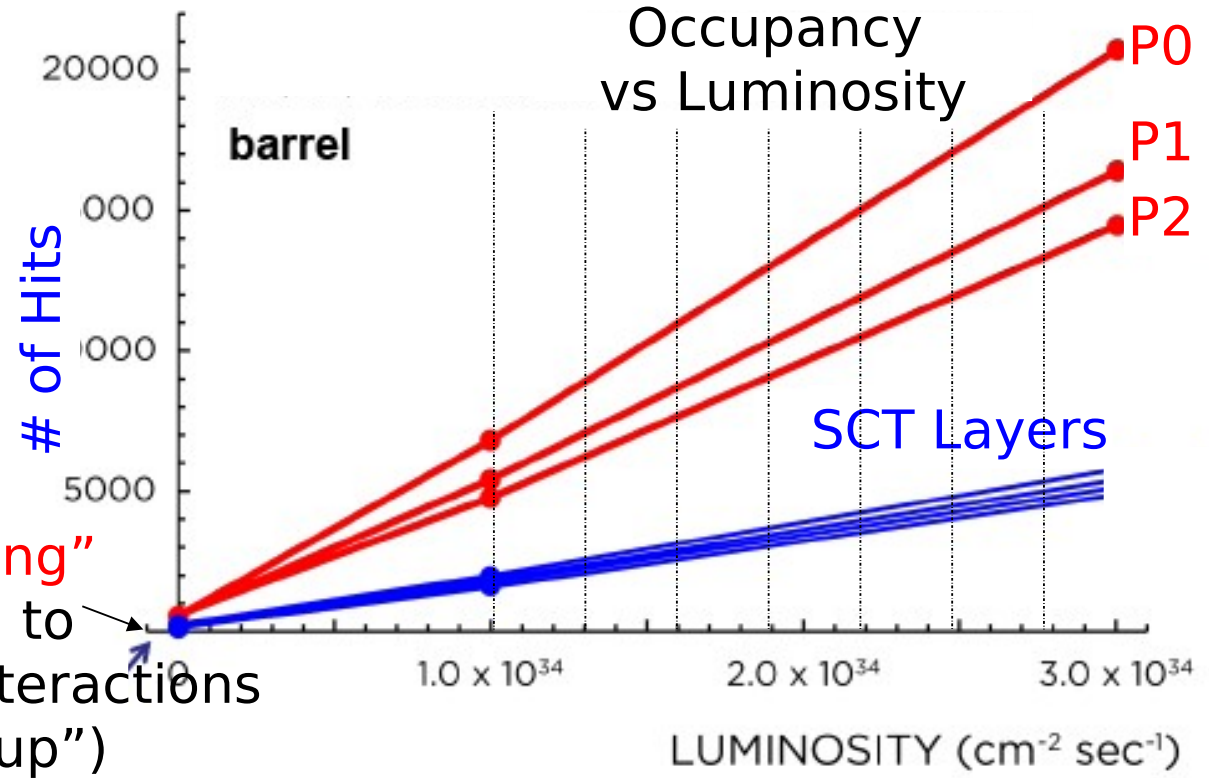


Atlas event with a Z boson decaying to two muons and 24 additional interaction vertices.



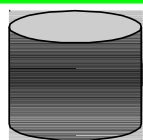
2 "real tracks" + extra hits

Hits from the "Hard pp scattering"
→ negligible fraction compared to
the non-interesting extra pp interactions
at same bunch crossing ("pile-up")



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

**Permanent
storage**



FTK

High Level Triggers: PC farms
~0.2s, ~1 kHz

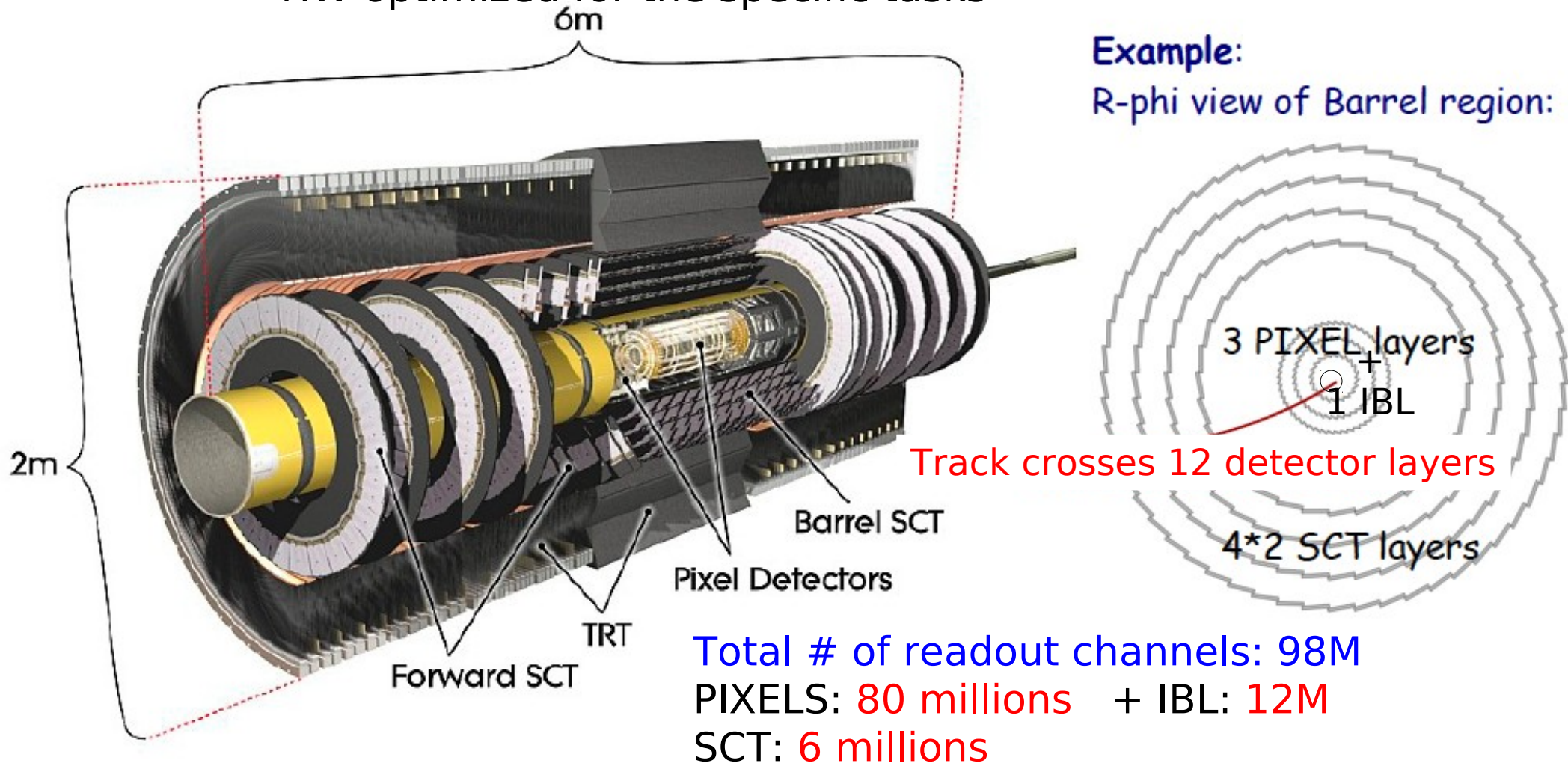
L1: hardware
{2.5 μ s, 100 kHz}



FTK: Tracking particles in the Silicon Detectors

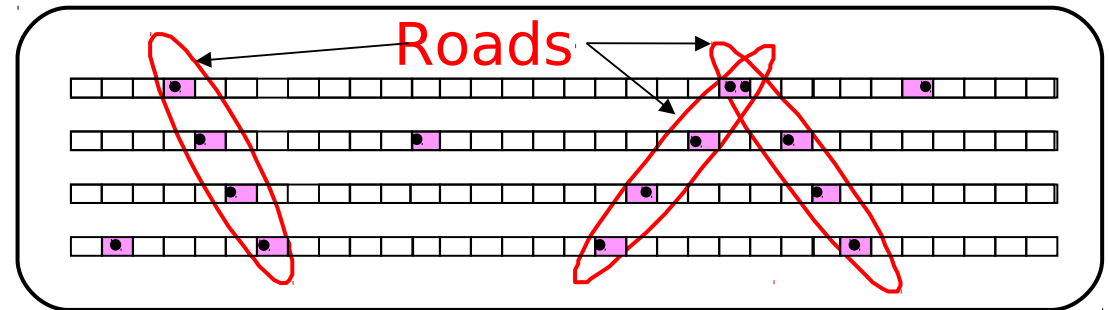
ATLAS' Fast Tracker (FTK) processes all Level-1 accepted events (100kHz)
Output: all tracks w/ $p_T > 1$ GeV available to HLT. **Typical FTK latency $\sim 100\mu\text{s}$** , compared to **$O(50\text{ms})$ HLT**

Advantages: high-bandwidth connection with detector & HW optimized for the specific tasks



Very fast tracking - tracking in 2 steps

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

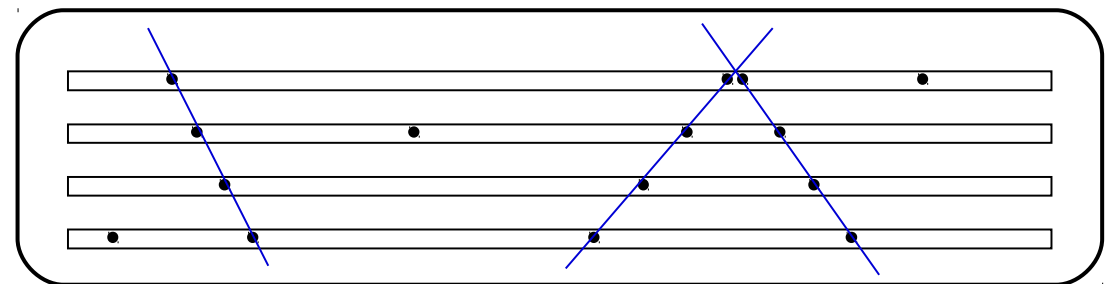


Pattern recognition w/ Associative Memory

Originally:

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

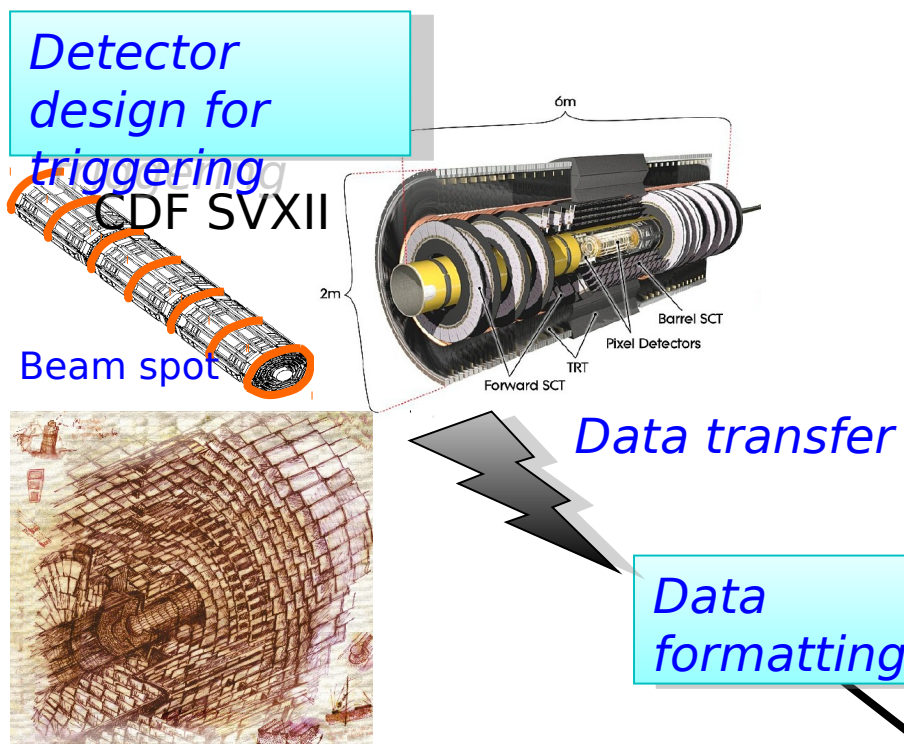
2. Then track fitting inside the roads. Thanks to 1st step, this is much easier.



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

Excellent results with linear approximation!

From hits to tracks ready for the HLT in $< 100 \mu\text{s}$



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

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

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

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

Pattern Recognition

Associative Memory approach discussed here

Track Fitting

FPGAs

HLT

1. Data Formatting:

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

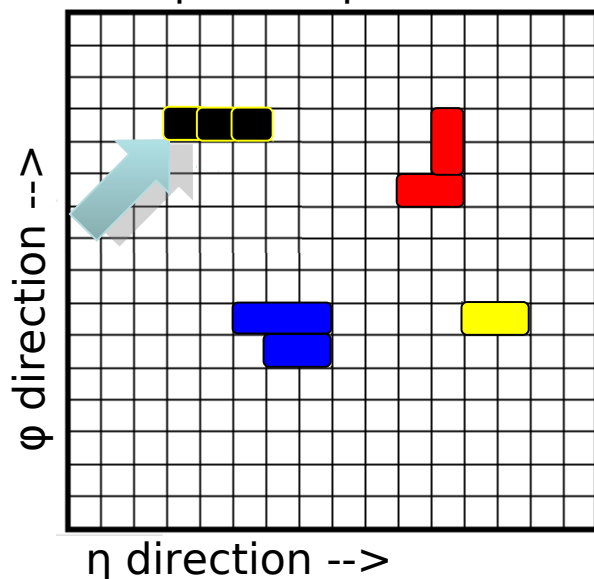
Clustering algorithm: how-to

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

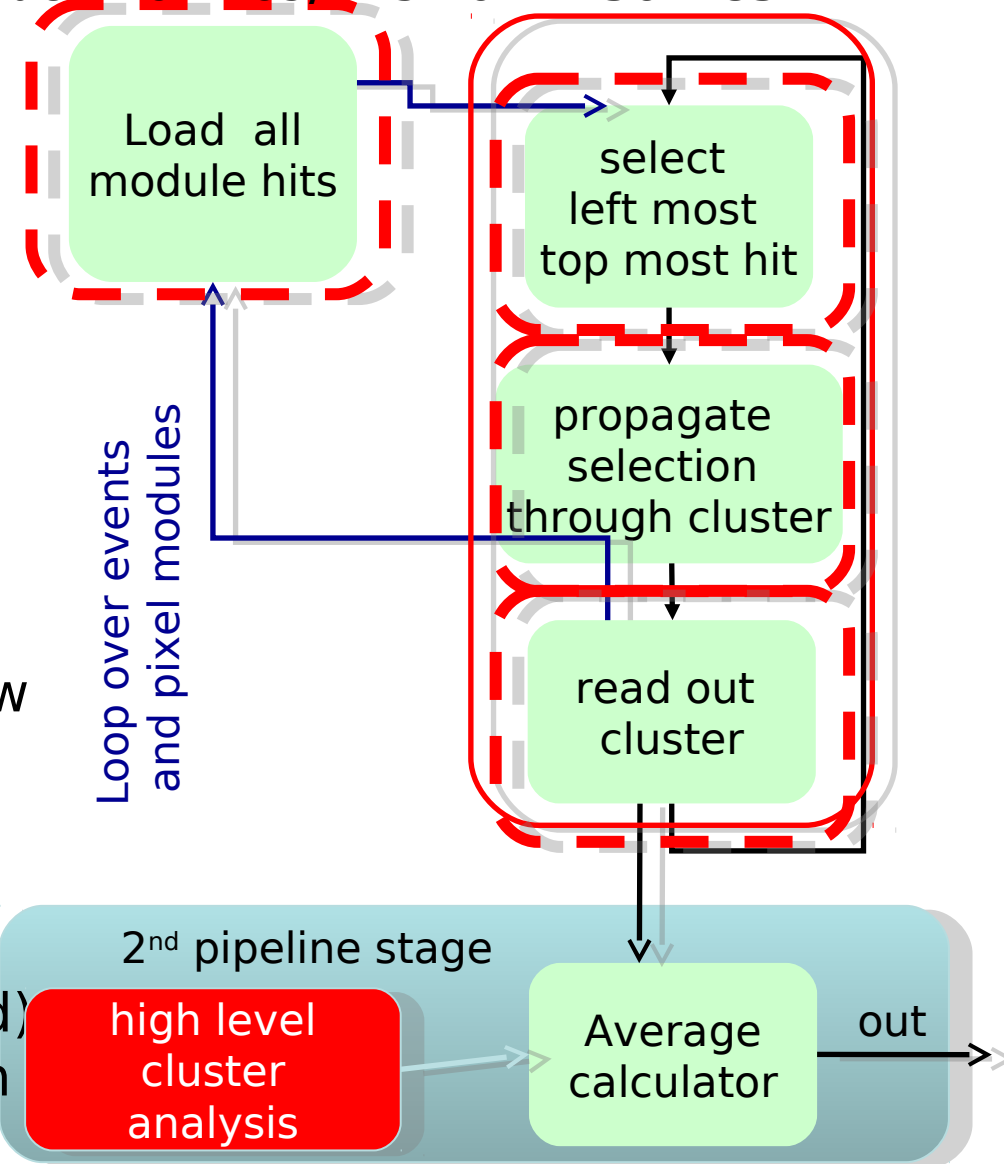


1st phase:

- 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!

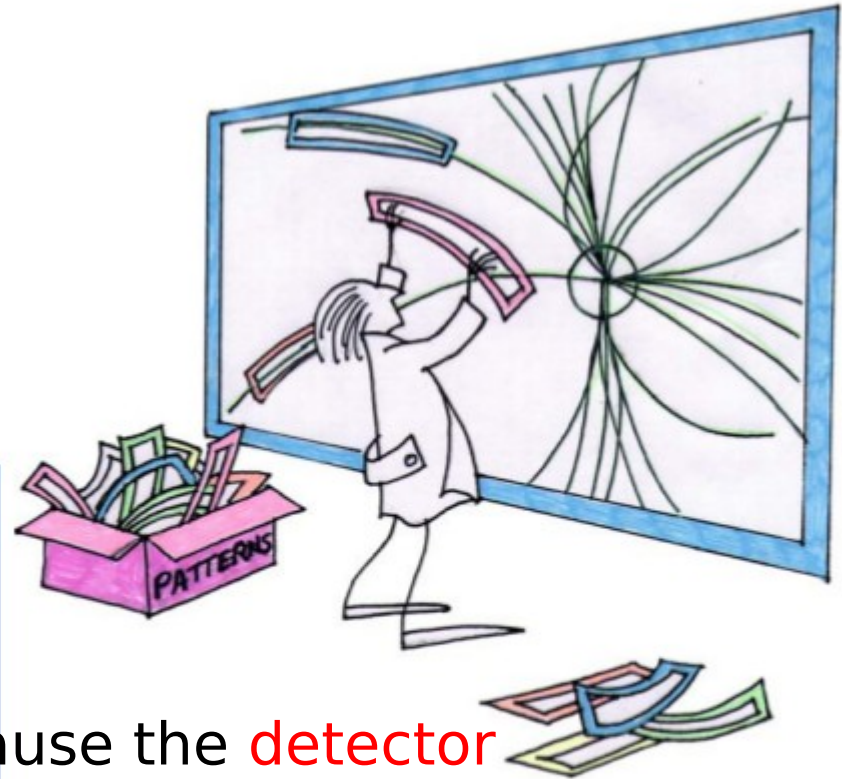
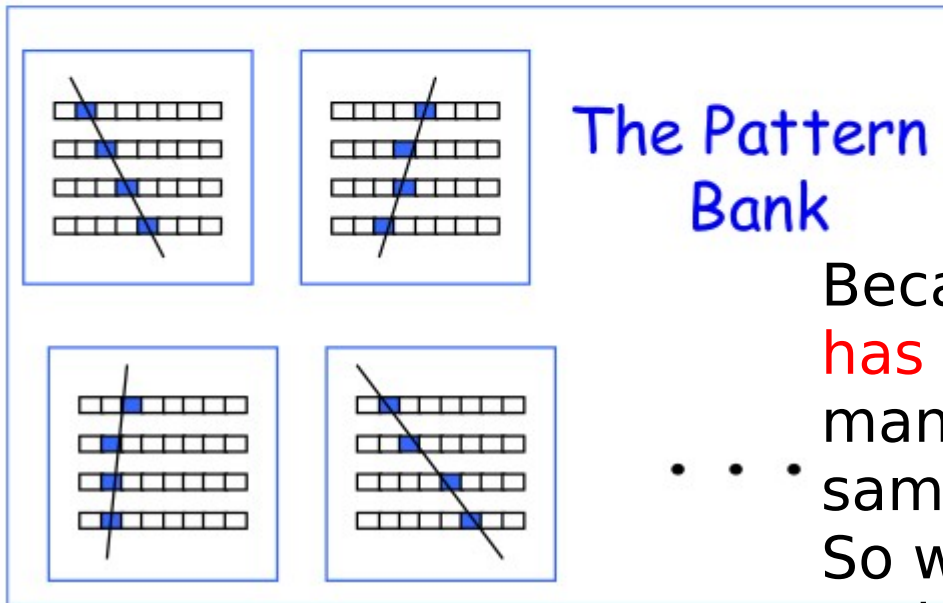
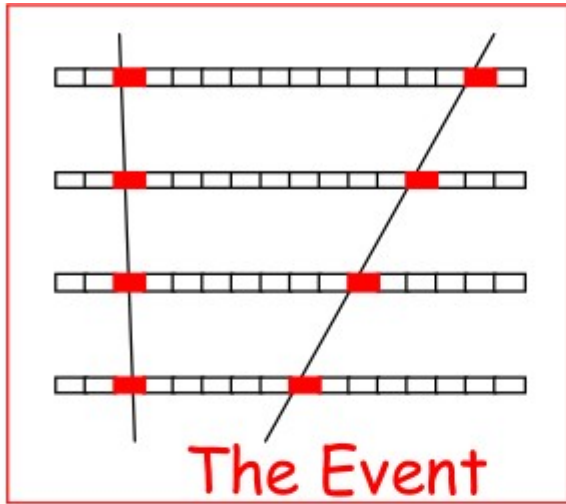


2.

The coarse pattern matching first

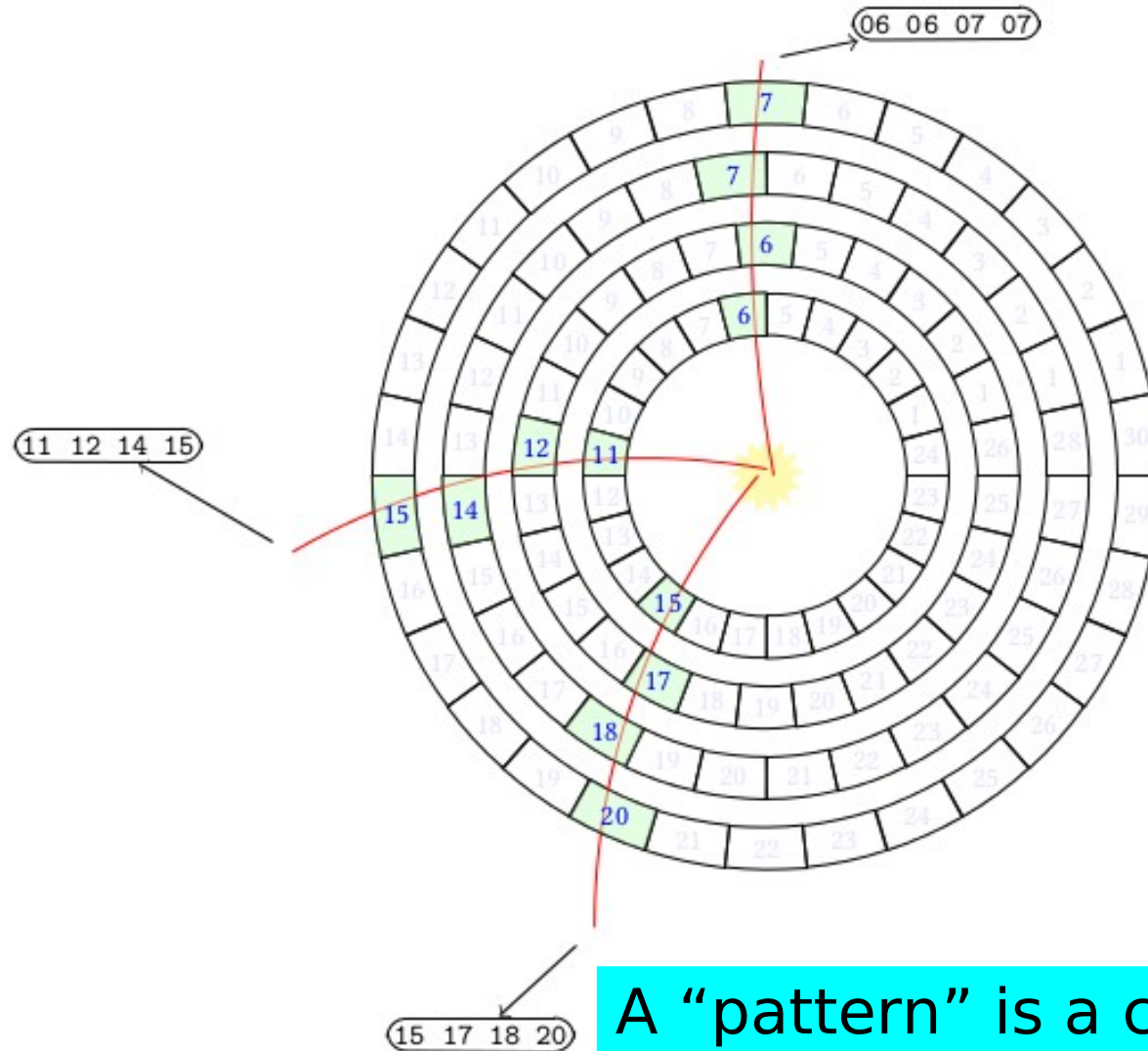
Using 8 out of the 12 silicon layers

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 **finite-size pattern-bank**.

Each track gives a possible pattern

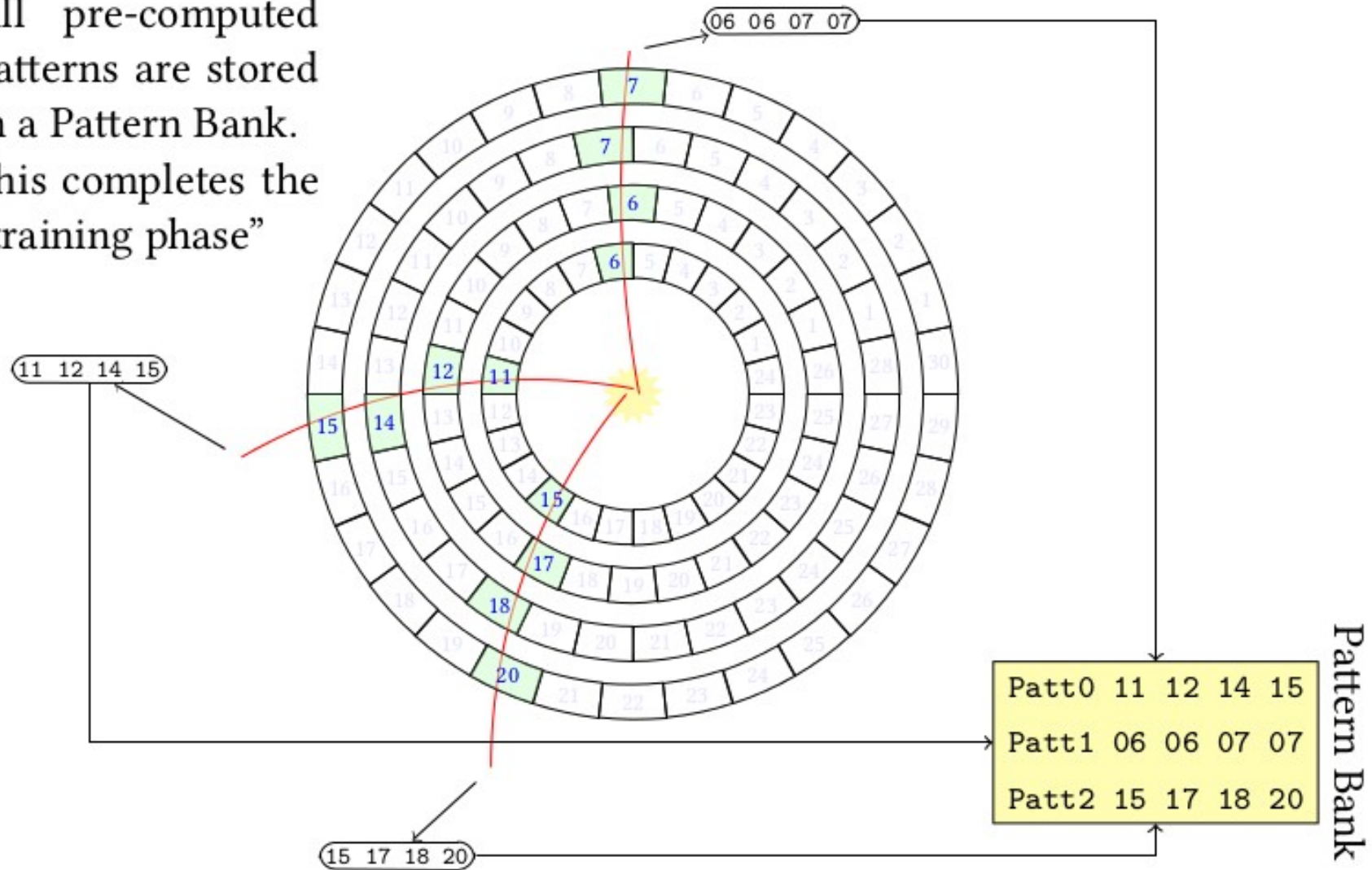


Each track is associated to an unique combination of coordinates per layer.

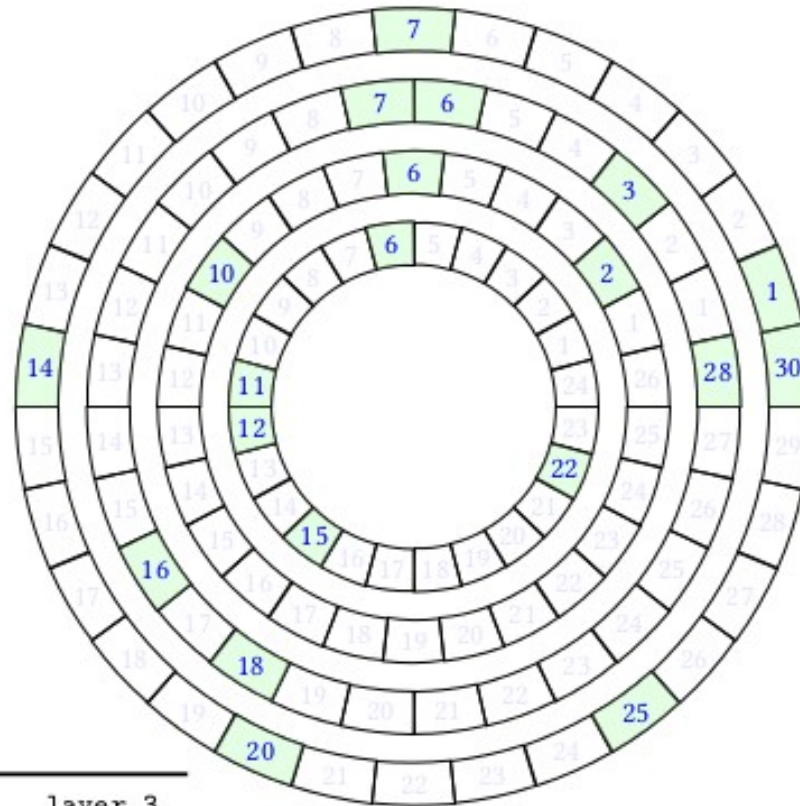
A “pattern” is a collection of numbers: one coordinate for each detector layer

Training: Make a "bank" of all patterns

All pre-computed patterns are stored in a Pattern Bank. This completes the "training phase"



Compare your data (the “event”) to all patterns in the bank



An event appears as a list of sparse hits on each layer. The hits are compared with the pattern bank.

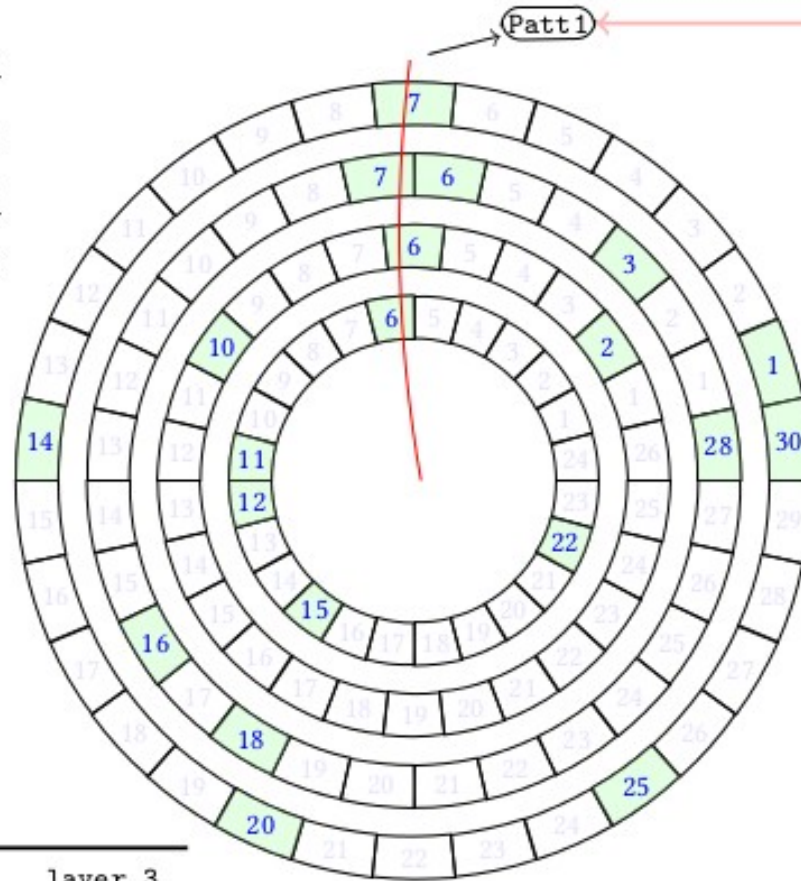
layer 0	layer 1	layer 2	layer 3
6	2	3	1
11	6	6	7
12	10	7	14
15		16	20
22		18	25
		28	30

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

Pattern Bank

Data vs. precalculated patterns → find matching pattern

In this example:
 After the pattern matching one track has been recognized in 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

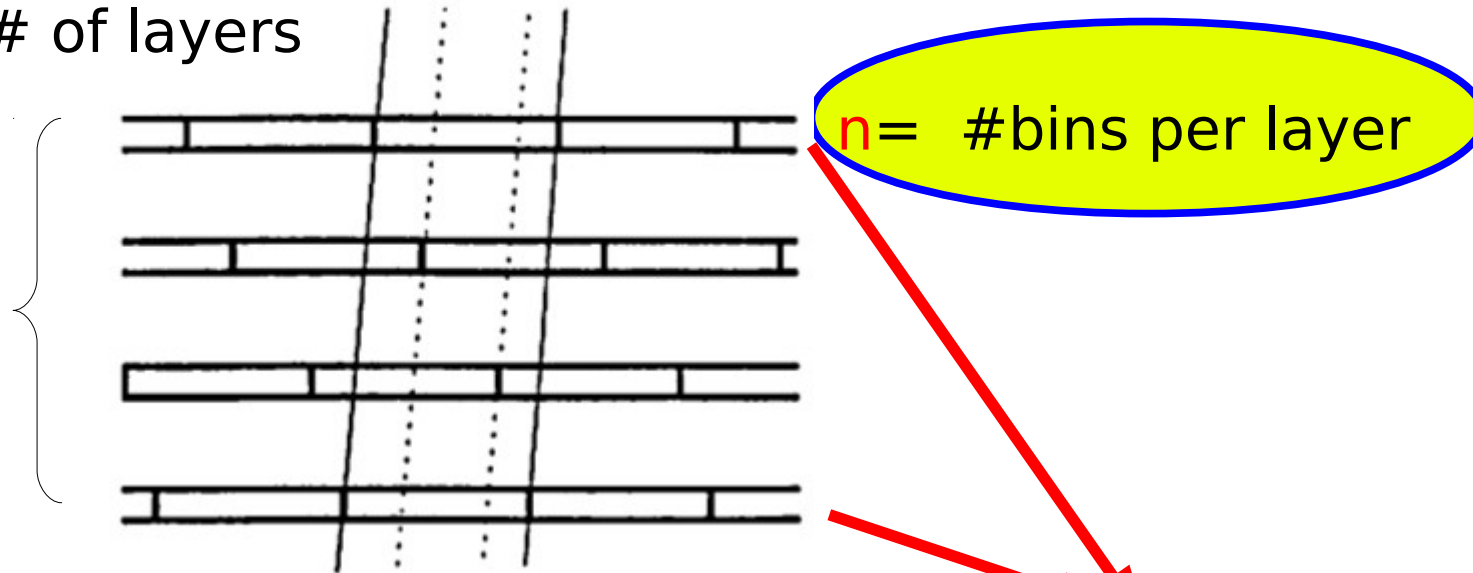
Patt0	11	12	14	15
Patt1	06	06	07	07
Patt2	15	17	18	20

Pattern Bank

How many patterns you need in your bank?

N_p = number of straight lines crossing the detector layers

m = # of layers



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

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

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

(Re-bining with 2-channels per bin: $n \rightarrow n/2$ means $N_p \rightarrow \frac{1}{4} N_p$)

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.

Example: The list to be searched: L = 1 3 4 6 8 9 11. The value to be found: X = 4.

Compare X to 6. X is smaller. Repeat with L = 1 3 4.

Compare X to 3. X is bigger. Repeat with L = 4.

Compare X to 4. They are equal. We're done, we found X.

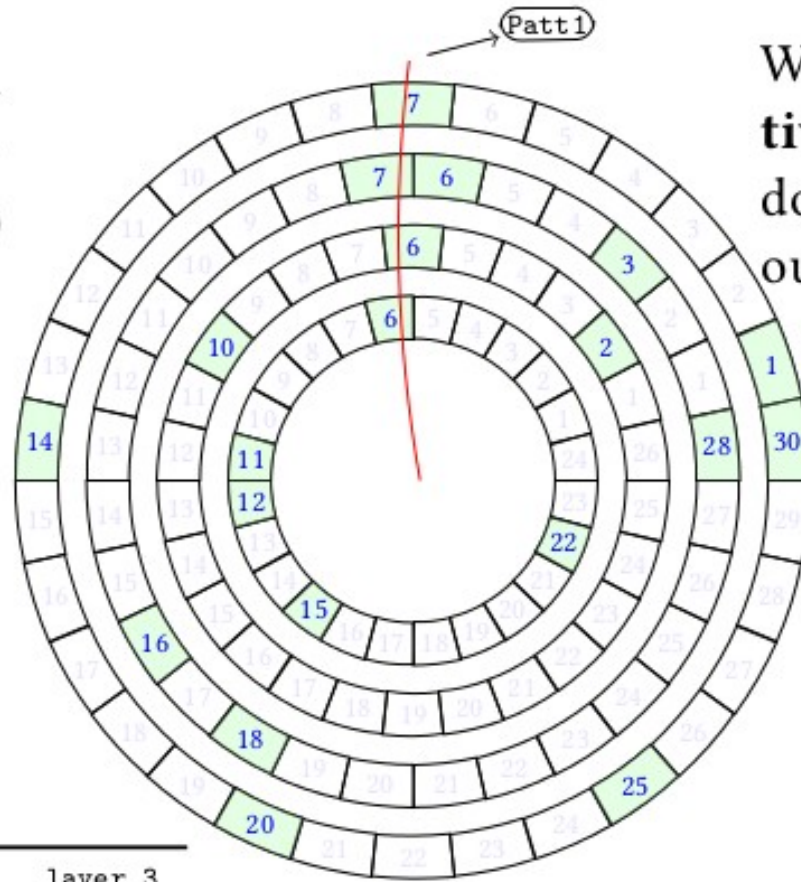
2a.

Speed is extremely important at triggering, so let's pay attention to this problem first:

Find track candidates very very fast

How to match data to patterns fast?

Compare all combinations of hits?
 $5 \times 3 \times 6 \times 6 = 540$
 operations.



With the **Associative Memory** it's done during the read-out!

Ultimate speed for pattern matching

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

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

Pattern Bank

which matching technique?

Ultimate speed: Associative Memories

436

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

October 24, 1988

VLSI STRUCTURES FOR TRACK FINDING

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.

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

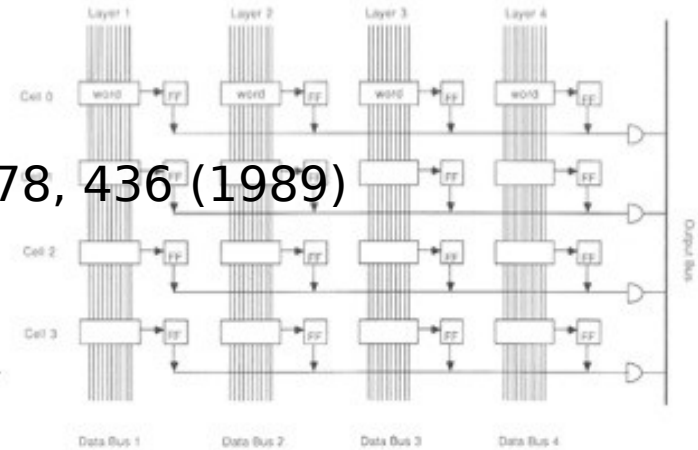


Fig. 3. Associative memory architecture.

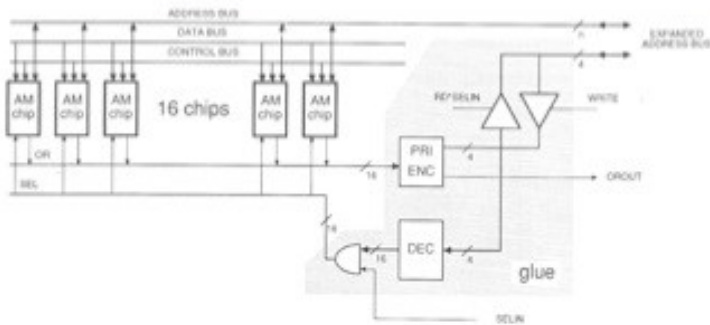


Fig. 5. 16 AM chips tied by the "glue".

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.

Associative Memory (AM) = a kind of 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 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”

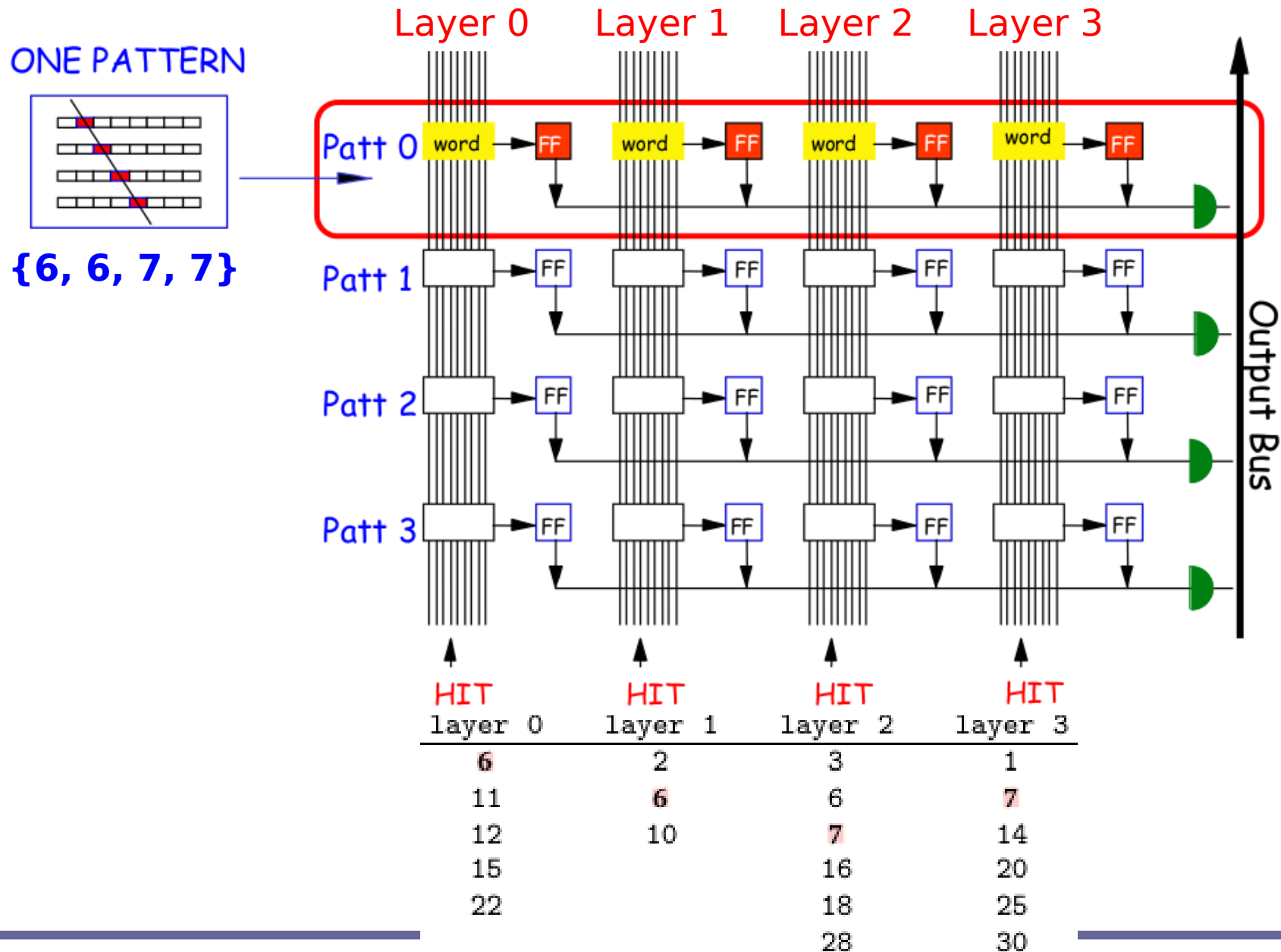
Ternary CAMs

- **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:

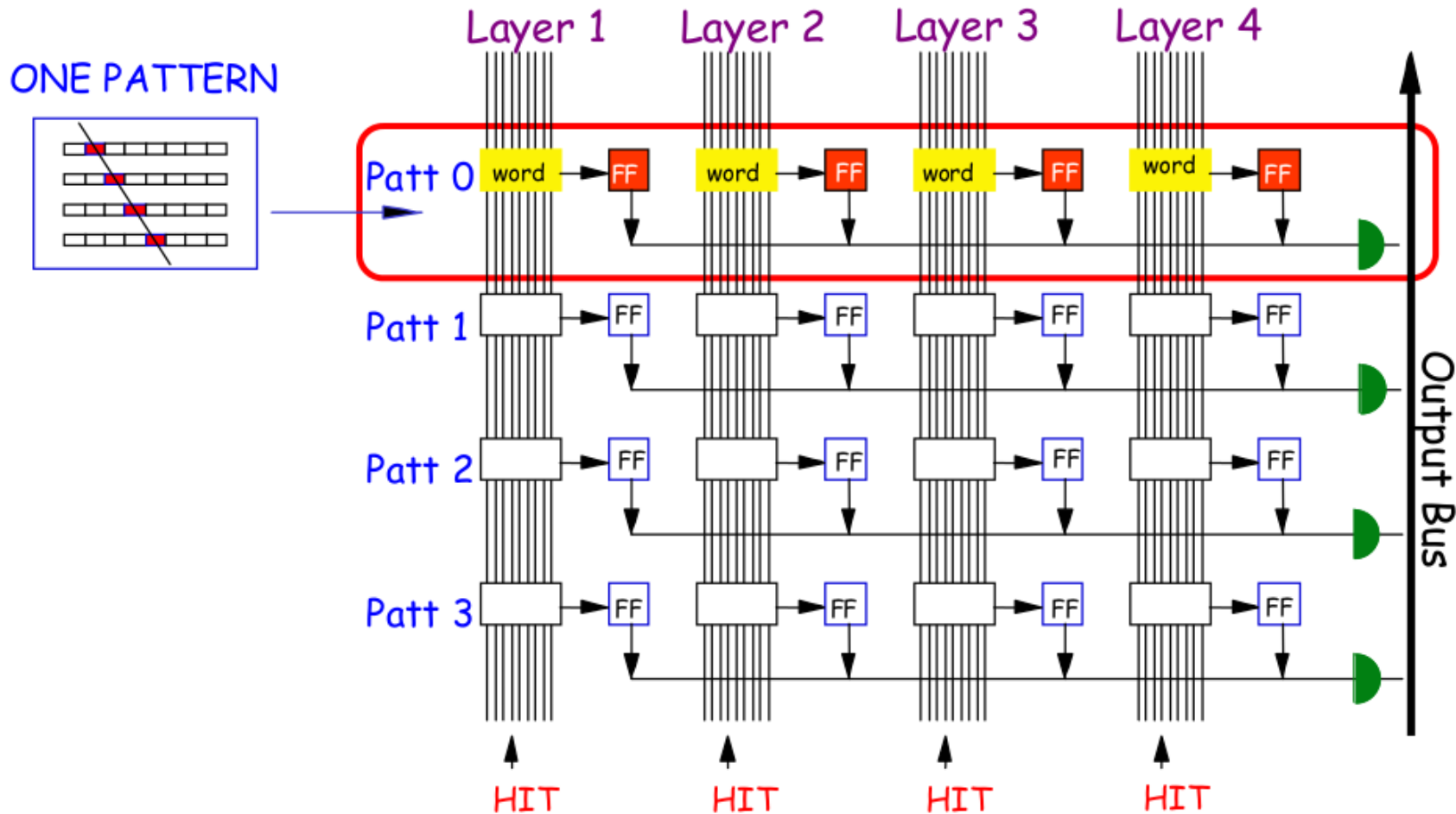
- the internal memory cell must now encode *three possible states instead of the two of binary CAM*. This additional state is typically implemented *by adding a mask bit ("care" or "don't care" bit) to every memory cell.*

Associative Memory: a CAM checking matching of each hit independently



AM chip working principle

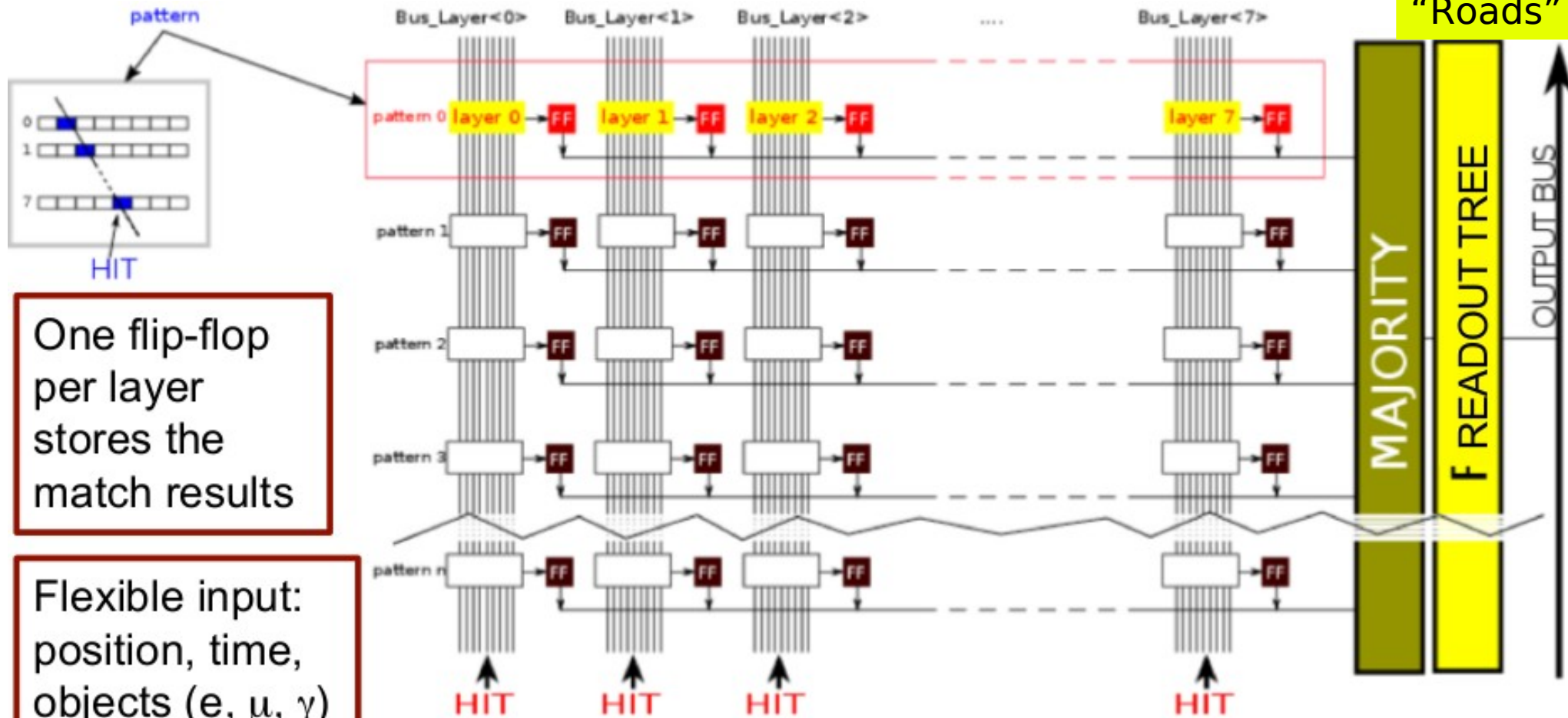
1. Load the patterns: each active bin in each layer is a number



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 in each hit independently**
→ **AND all flags to get a complete pattern matching.**

Track trigger w/ pattern matching AM

Result:
Matched
Patterns
"Roads"



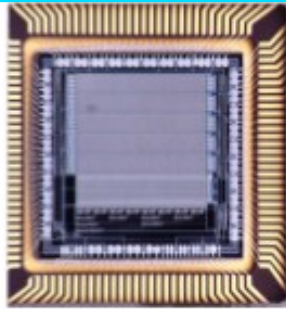
One flip-flop per layer stores the match results

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 evolution: mainly ASICs

SVT
AM chip



- (90's) **Full custom VLSI chip** - $0.7\mu\text{m}$ (INFN-Pisa)
- **128 patterns, 6x12bit words each, 30MHz**

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

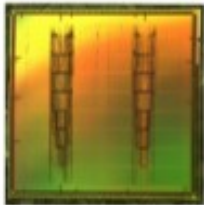


Alternative **FPGA** implementation of SVT AM chip

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

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

SVT upgrade

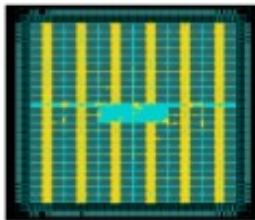


Standard Cell $0.18\mu\text{m}$ \rightarrow **5000 pattern/AM chip**

SVT upgrade total: 6M pattern, 40MHz

A. Annovi et al., **IEEE TNS**, Vol 53, Issue 4, Part 2, **2006**

FTK R&D

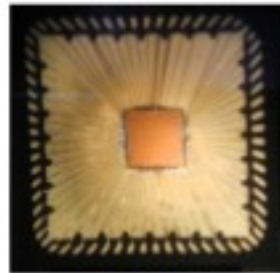


AMchip04 –65nm technology, std cell & full custom, 100MHz
Power/pattern/MHz \sim 30 times less. Pattern density x12.

First variable resolution implementation!

F. Alberti et al 2013 **JINST** **8 C01040**, doi:[10.1088/1748-0221/8/01/C01040](https://doi.org/10.1088/1748-0221/8/01/C01040)

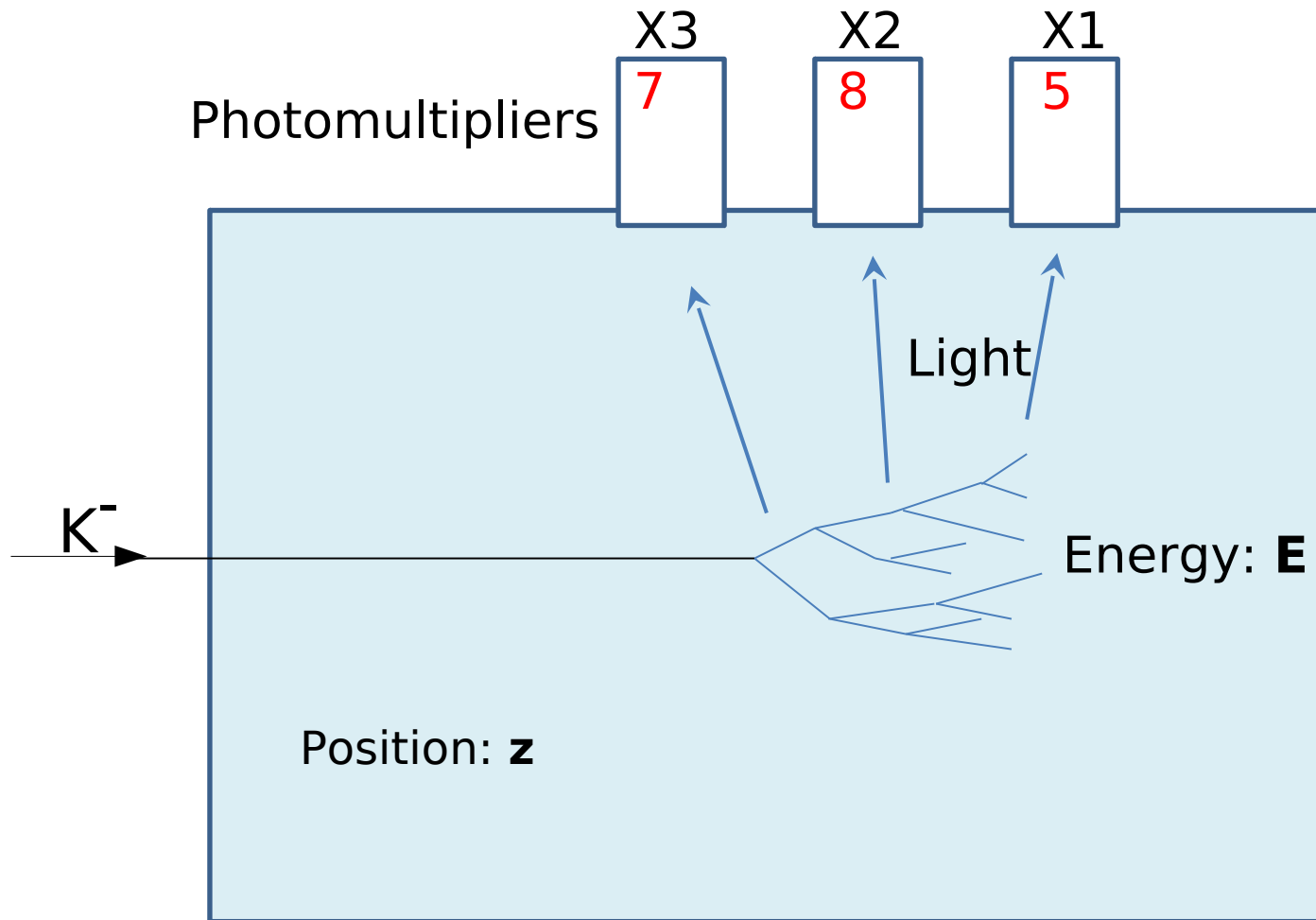
AM chip for FTK: AMchip06



- ▶ 90's Full custom VLSI chip - 0.7 μ m (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. Instr. 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 μ m 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 .

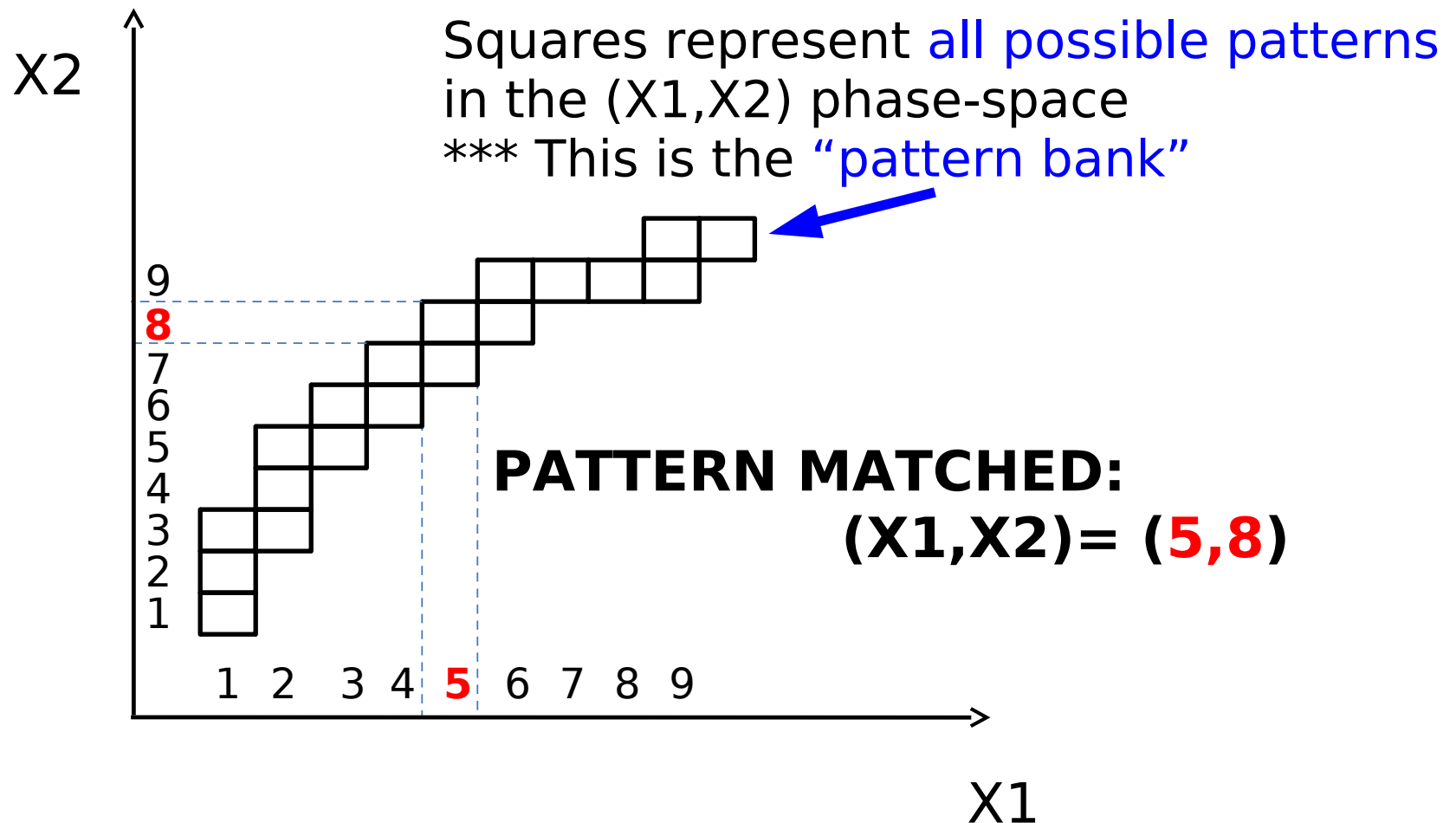
AMchip05: switched to serialized IO (11*2Gbs)
AMchip06: the FTK AM chip with 128k patterns/chip

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

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



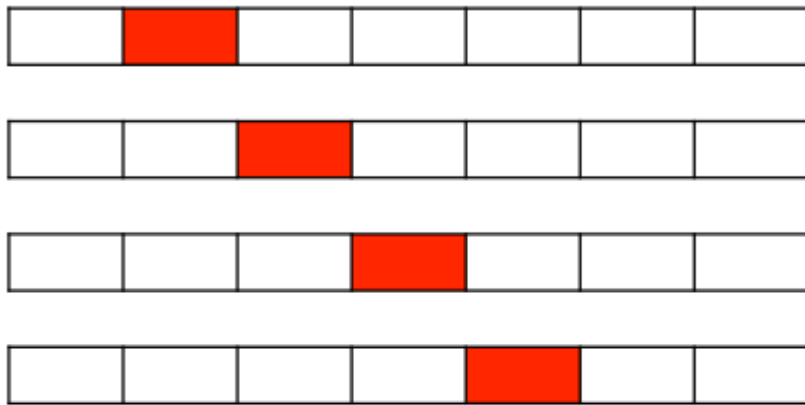
2b.

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

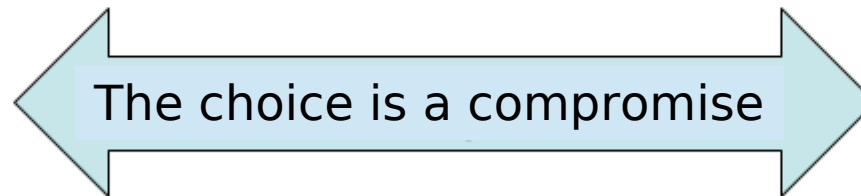
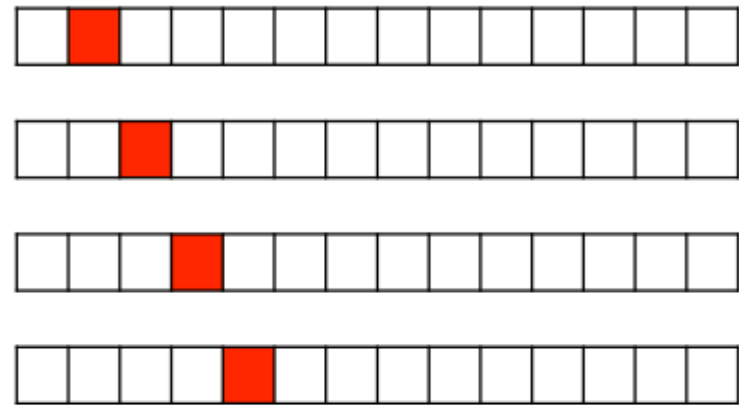
how do we deal with the number of patterns which can be big in the high-granularity detectors?

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

Wide patterns



Thin patterns



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

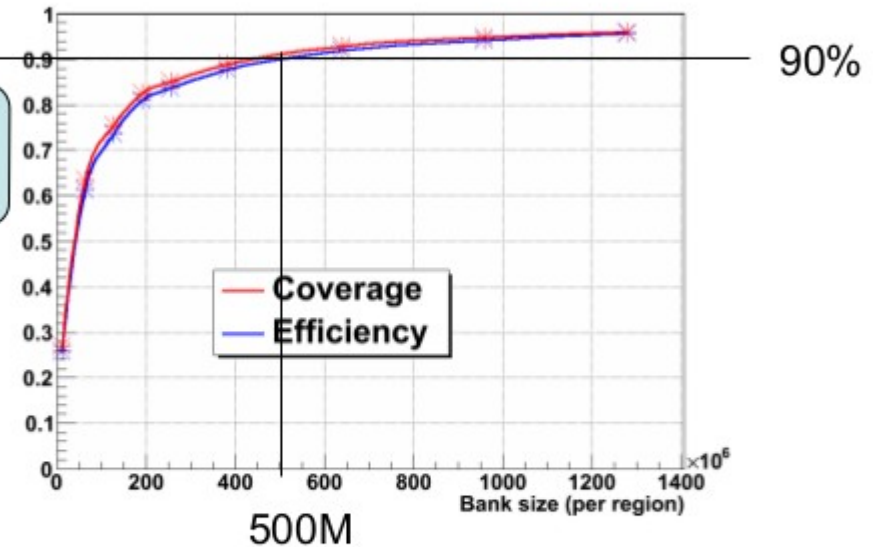
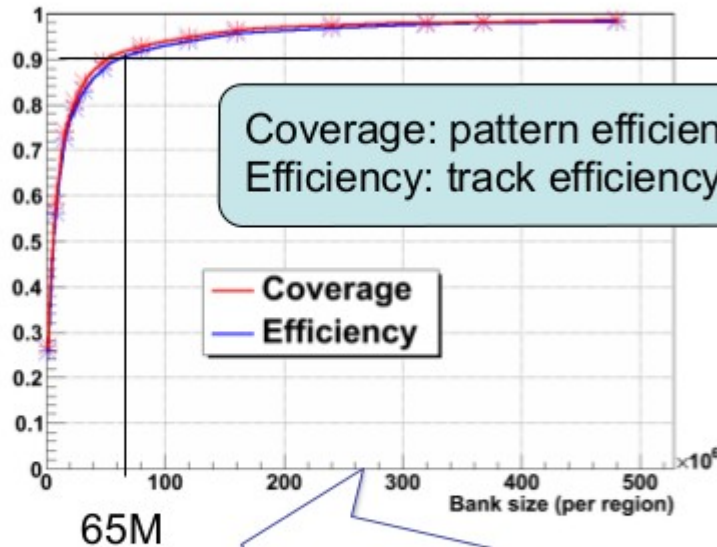
High efficiency with large bins (small number of patterns) → but, lots of fake tracks found → lots of work for detailed track fitting!

We prefer “few patterns” - “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 degrees)

<# 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

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

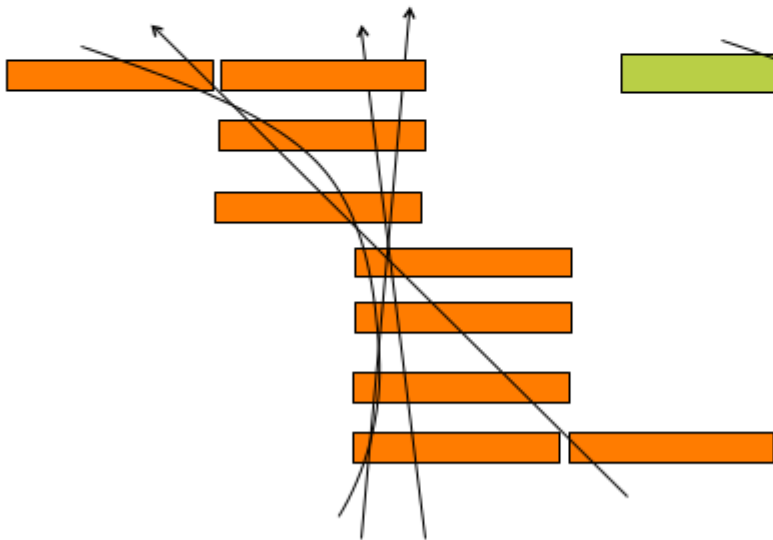
“0x11x” selects bins 6,7,14, or 15

“111xx” selects bins 28 to 31

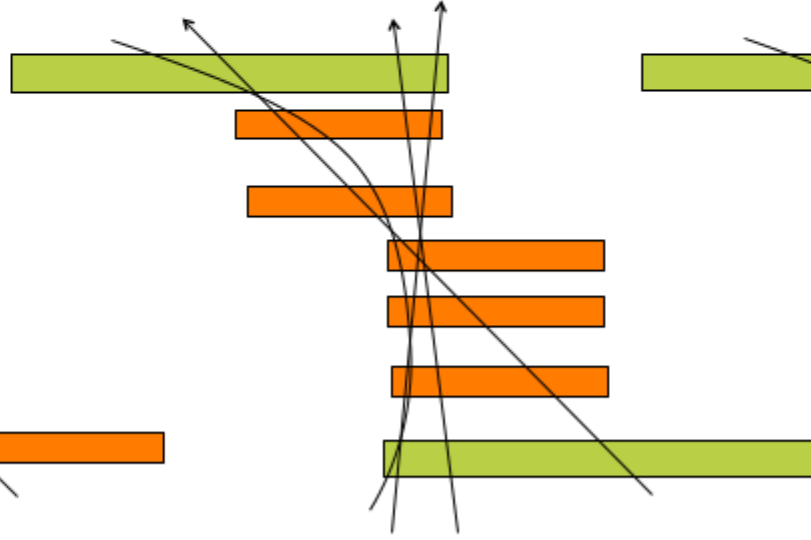
Refinements: majority & variable widths

- Majority Logic: Only require N out of M layers have a match
 - Gains efficiency
- Variable Resolution Patterns (Don't Care Bits)
 - Reduces the number of patterns and fake matches

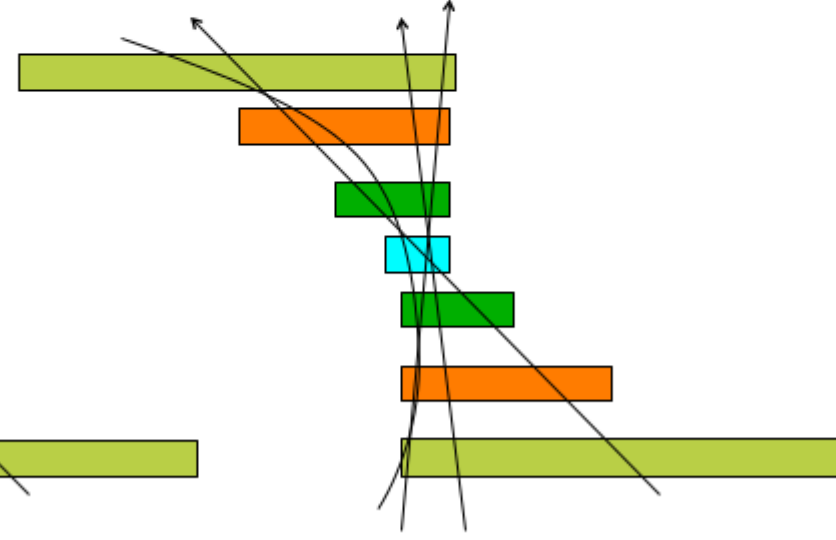
No variable resolution:
3 patterns needed



1 bit variable resolution:
1 pattern needed



3 bit variable resolution:
1 pattern with 1/16th volume



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

Alberto Annovi

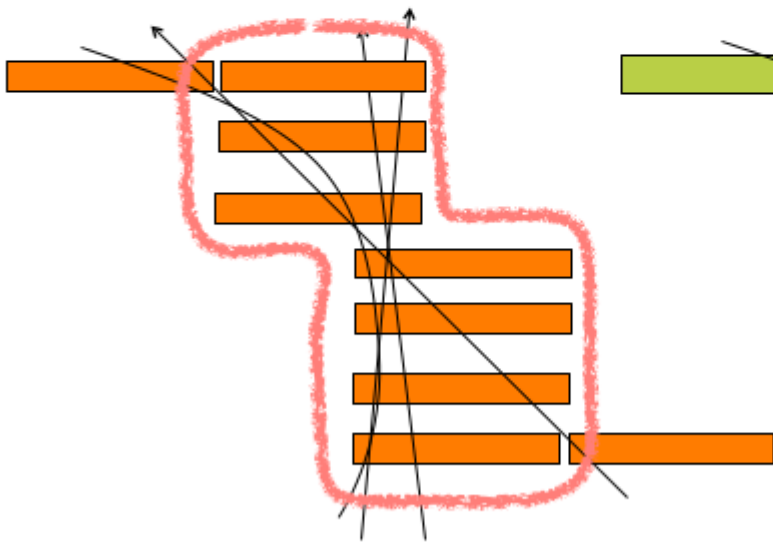
* ANIMMA - A new "Variable Resolution Associative Memory" for High Energy Physics
ATL-UPGRADE-PROC-2011-004, doi:10.1109/ANIMMA.2011.6172856

* "Variable resolution Associative Memory for the Fast Tracker ATLAS upgrade", ICATTP 2013

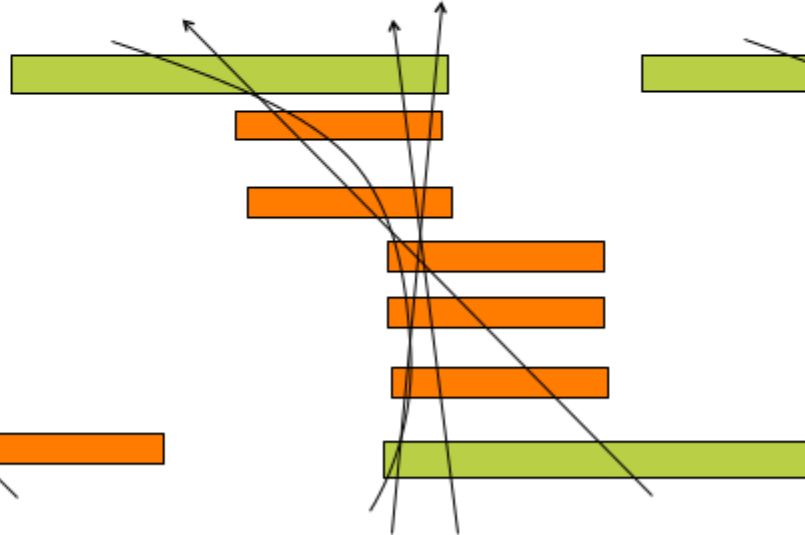
Refinements: majority & variable widths

- 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

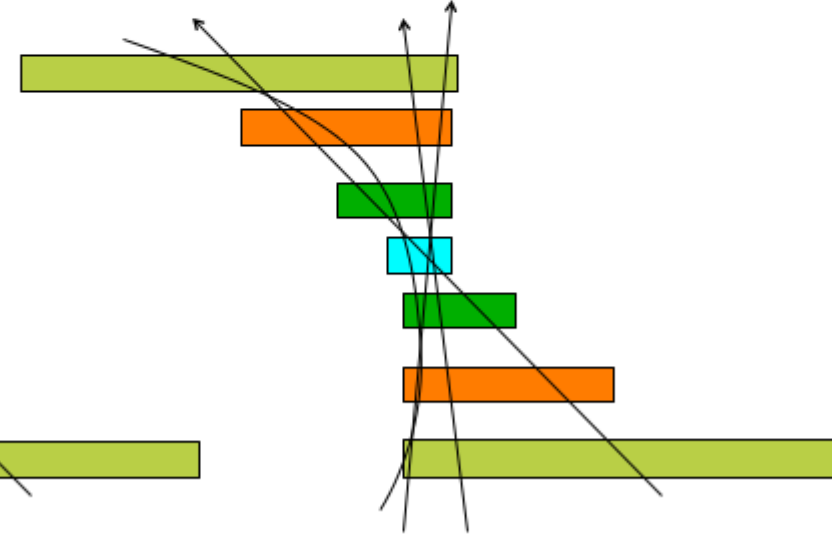
No variable resolution:
3 patterns needed



1 bit variable resolution:
1 pattern needed



3 bit variable resolution:
1 pattern with 1/16th volume



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

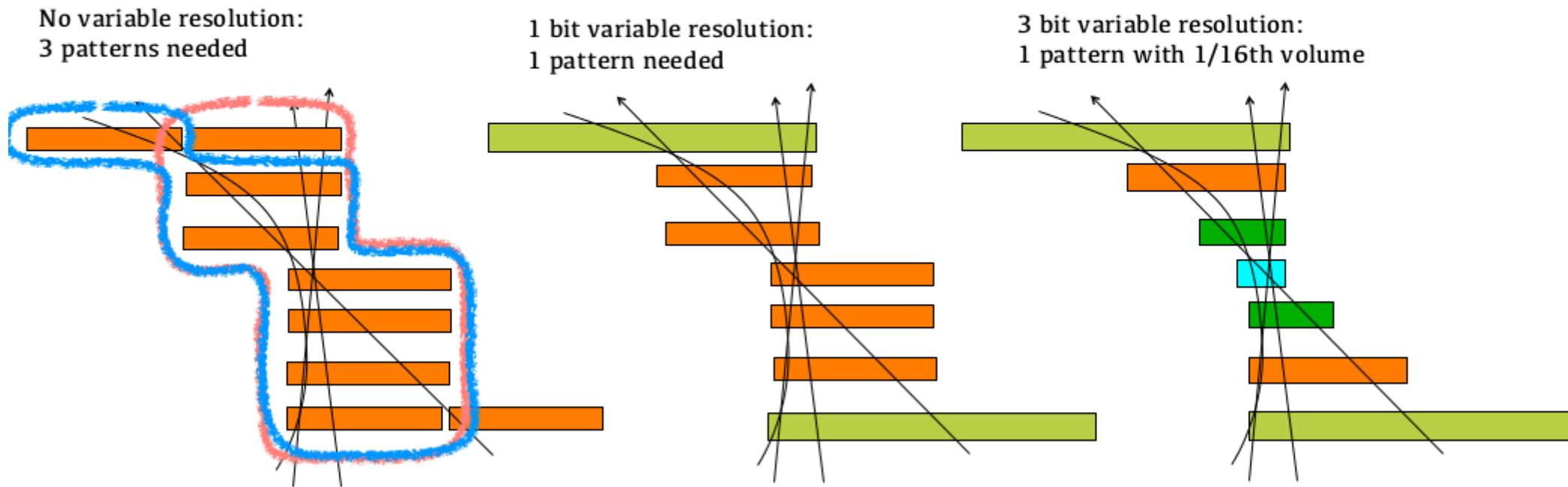
Alberto Annovi

* 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

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

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

Refinements: majority & variable widths

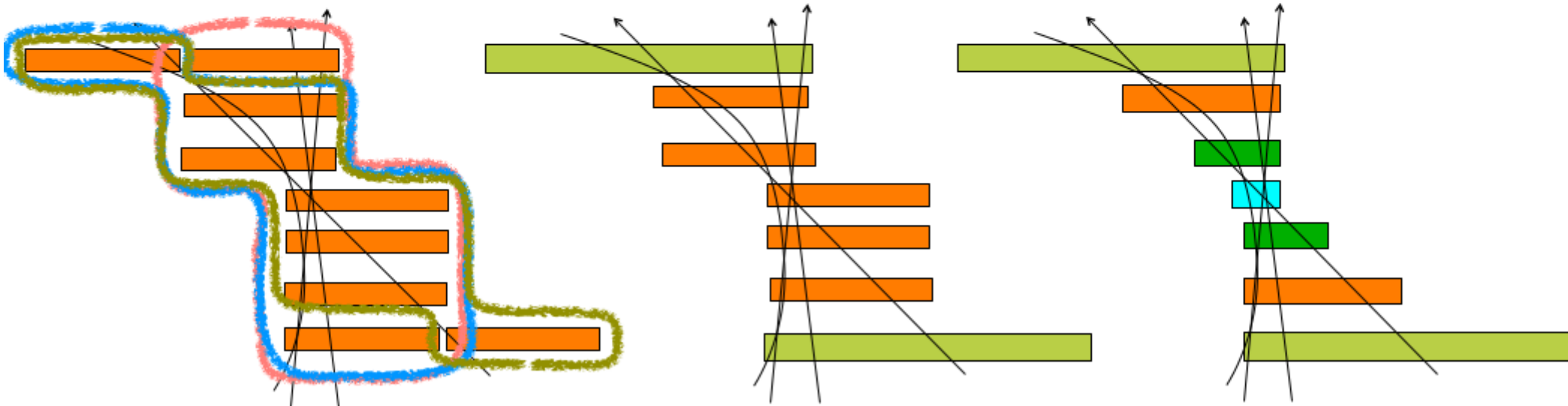
- Majority Logic: Only require N out of M layers have a match
 - Gains efficiency
- Variable Resolution Patterns (Don't Care Bits)

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

No variable resolution:
3 patterns needed

1 bit variable resolution:
1 pattern needed

3 bit variable resolution:
1 pattern with 1/16th volume



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

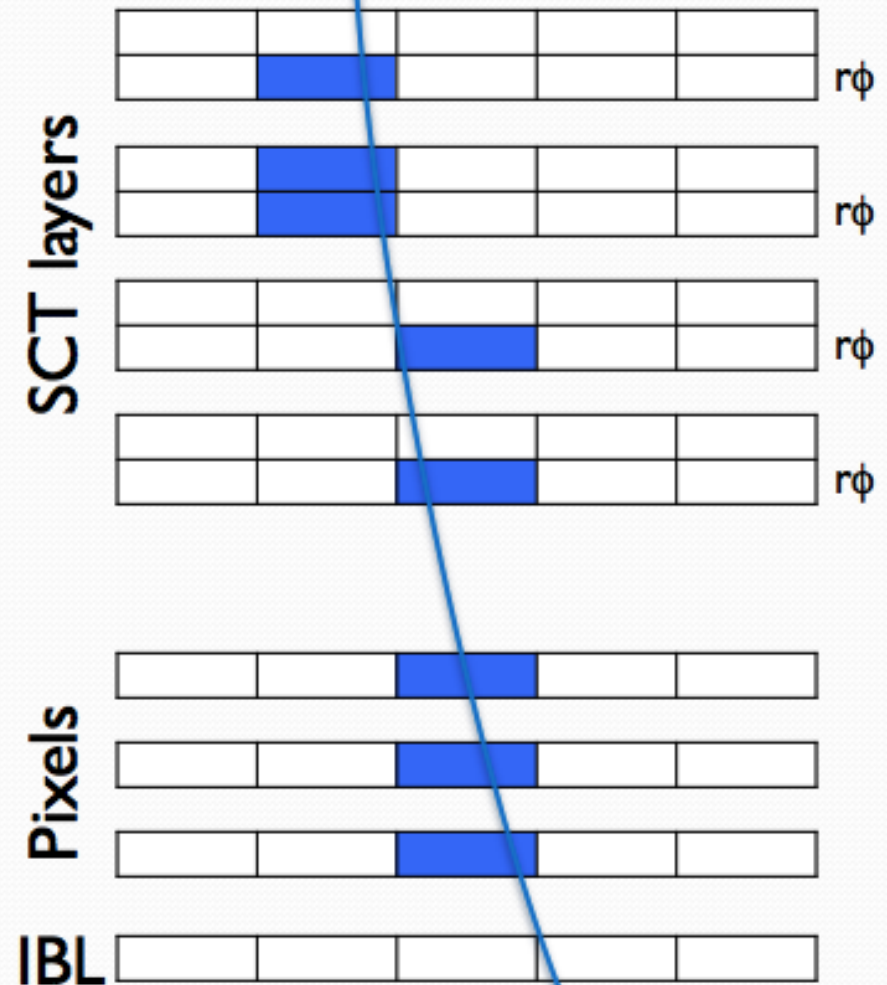
3.

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)

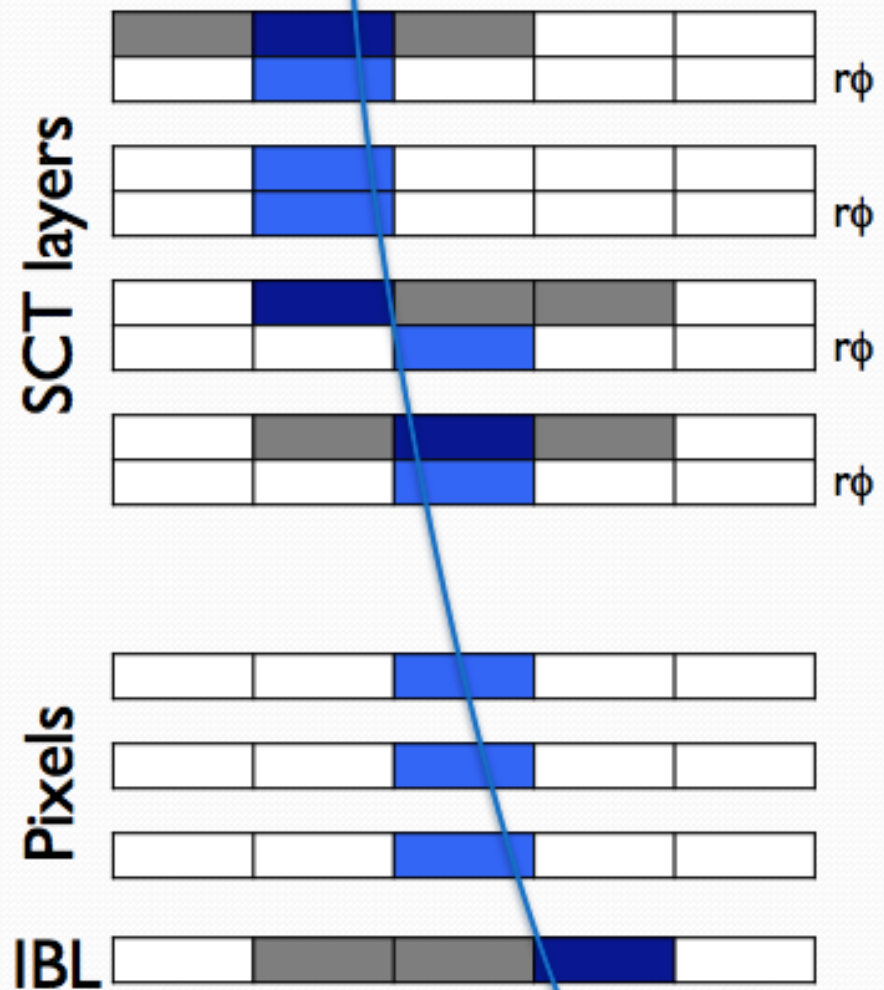
BUT: Linear approximation: get a set of **linear equations**

“each parameter depends linearly on the hits” → **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 



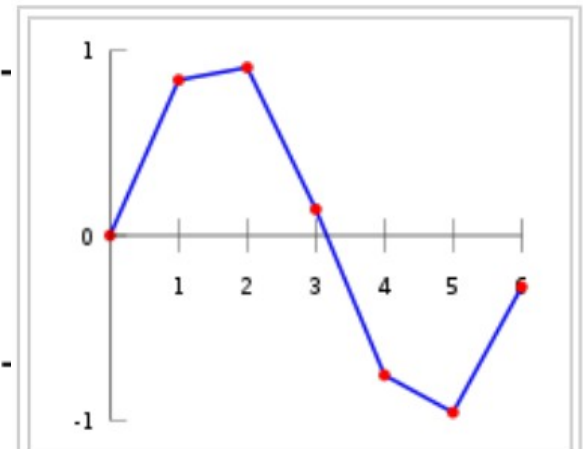
Done on FPGAs, on a “2nd stage” board

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

$$\sin(x) \approx x - \frac{x^3}{6} + \frac{x^5}{120}$$

```
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: $(15 \times 36)_{\text{pix}} \times 16_{\text{sct}}$
 - 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/cm²
- Hardware constraints (for each of 64 η - ϕ towers)
 - # AM patterns $< 16.8 \times 10^6$
 - # roads/event $< 16 \times 10^3$
 - # fits/event $< 80 \times 10^3$

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 * 10⁶	Efficiency %	roads / evt * 10³	fits / evt * 10³
Barrel	$(30 \times 72)_{\text{pix}} \times 32_{\text{sct}}$	$2_{\text{pix}} \times 1_{\text{sct}}$	16.8	93.3%	3.2	26
Endcap	$(30 \times 72)_{\text{pix}} \times 32_{\text{sct}}$	$2_{\text{pix}} \times 1_{\text{sct}}$	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

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>

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

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 matching with “AM chip” (ASIC)**,
 - **refined track-fitting (and almost everything else needed, from formatting to smart databases, to I/O) in powerful modern FPGAs**

From the FTK to the future

- These ideas and technology will not be restricted to FTK in ATLAS
 - Future applications in HEP (tracking in L1 for ATLAS and CMS...)
 - Applications outside HEP (medical imaging, smart cameras, genomics, ...)
- FTK uses the AMchip06, an Associative Memory with
 - 128k patterns of 8 words \times 18 bits each word
 - high speed serial links
 - variable resolution (up to 6 ternary bits)
 - low power
 - 8 \times 16 bit comparisons at 100 MHz

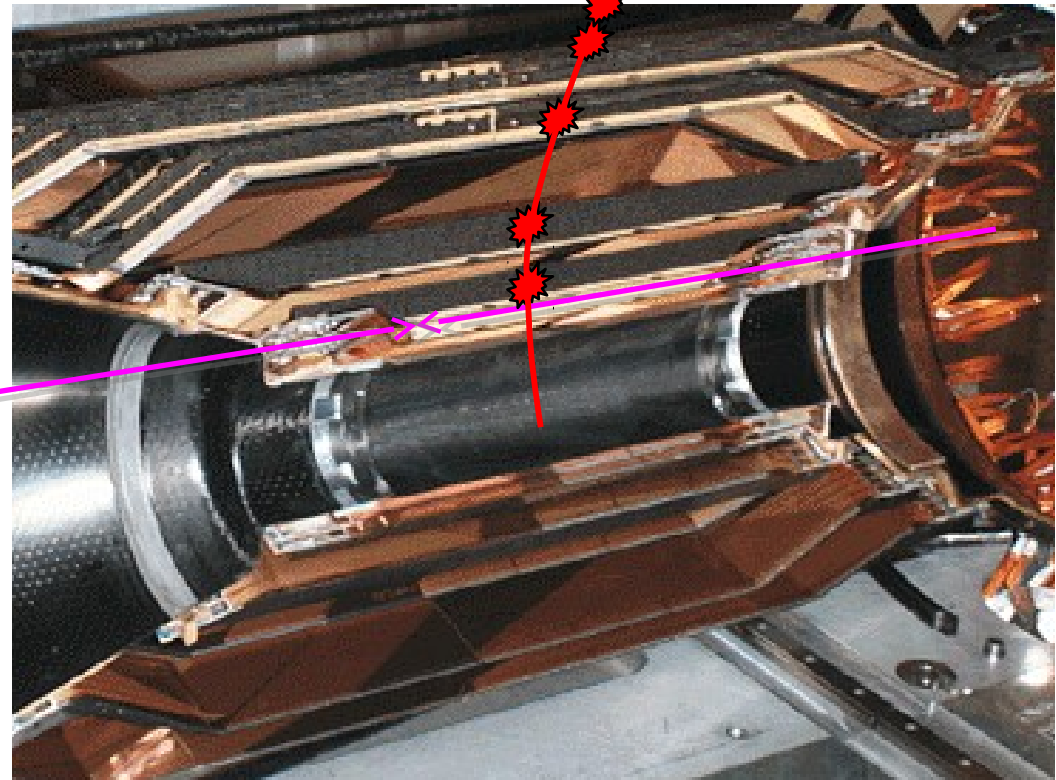
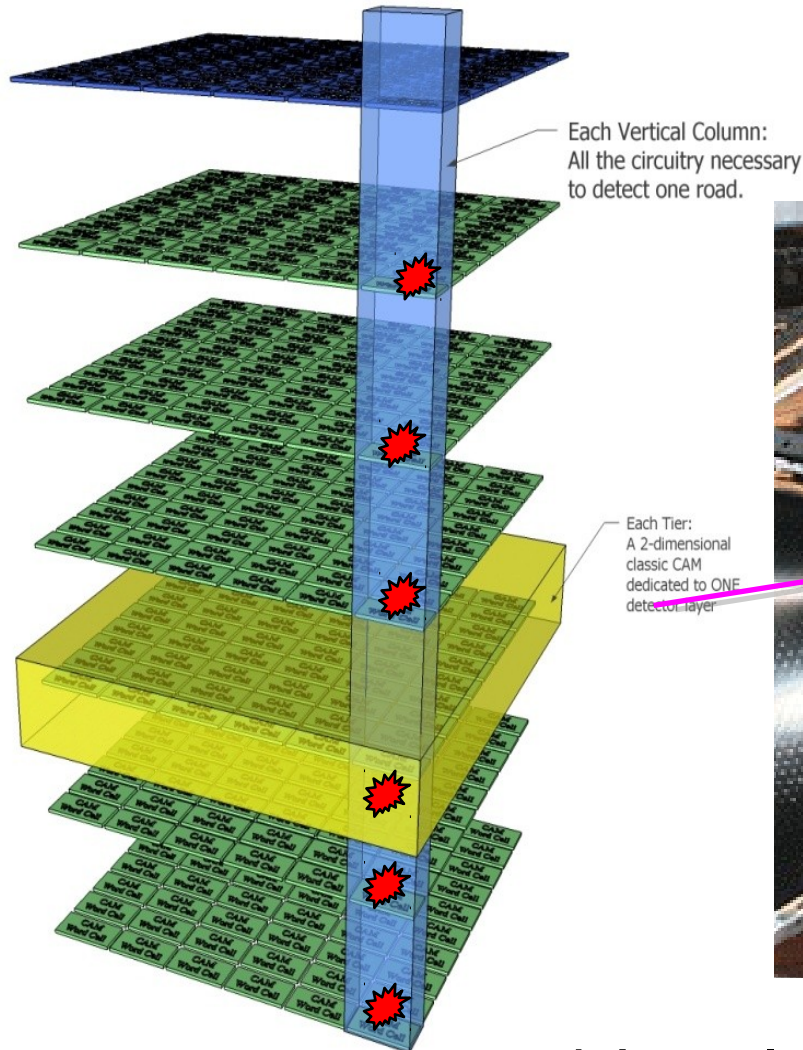
...and to keep an eye in what new technology can do for you:

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

Acknowledgments:

Alberto Annovi, Francesco Crescioli, Mauro Dell' Orso, Paola Giannetti,
and FTK members

Thank you!

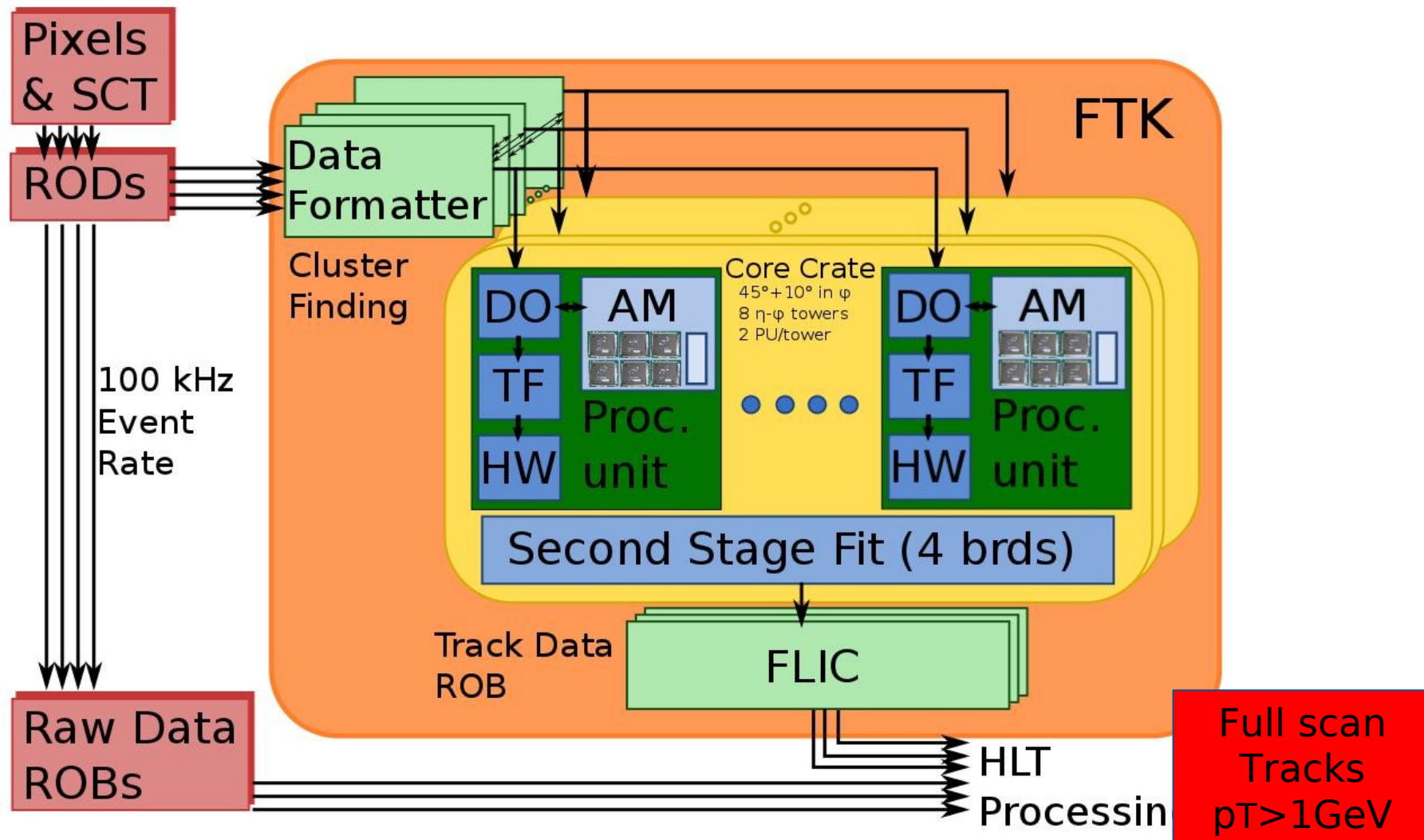
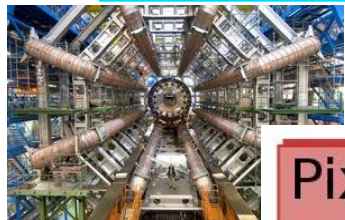
Hope you have enjoyed the lectures,
still enjoy the labs,
learned a lot,
got triggered(!) and intrigued to look up things on your own
and enjoy Rio!



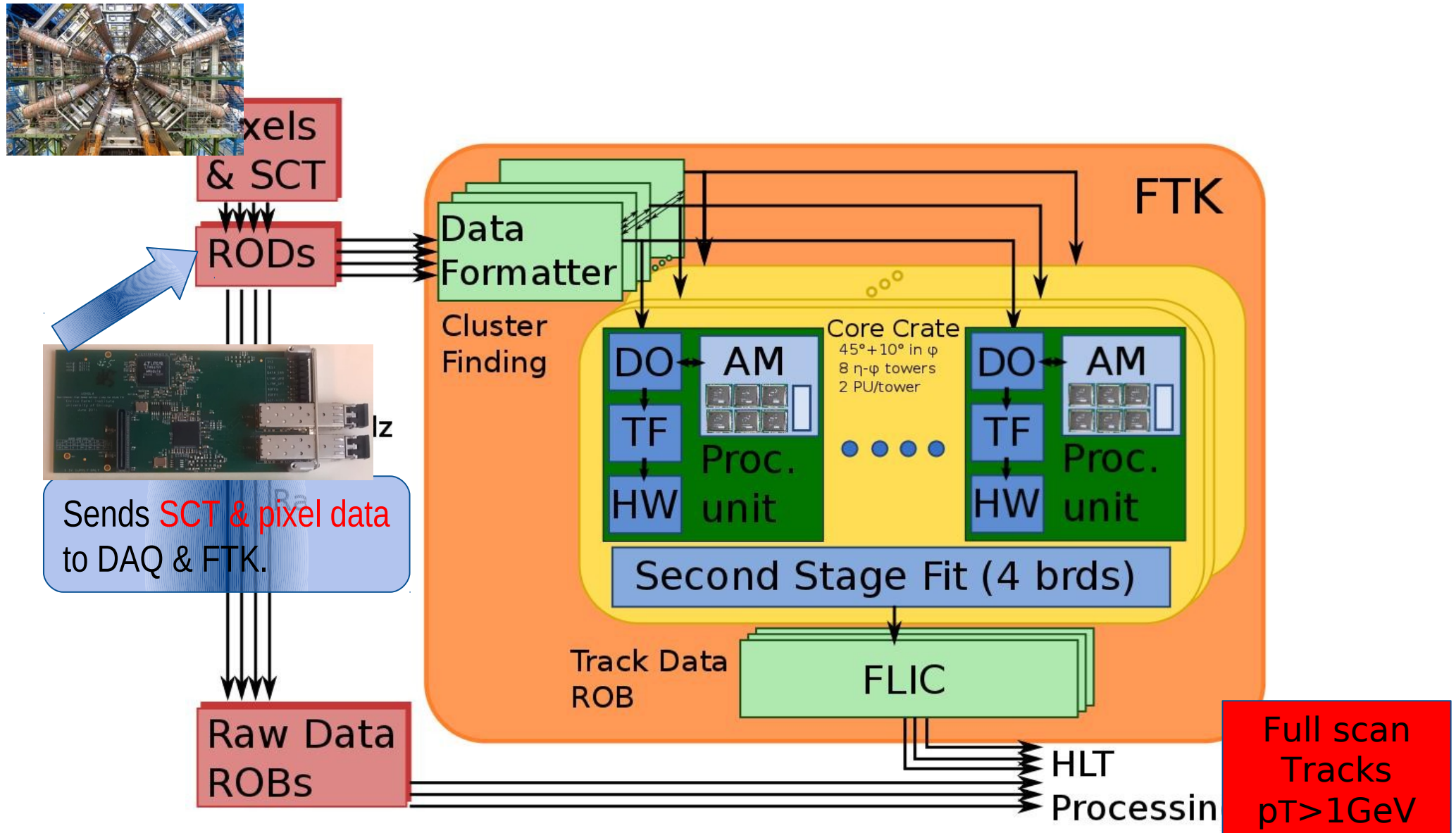
Extras...

4. FTK: the system components

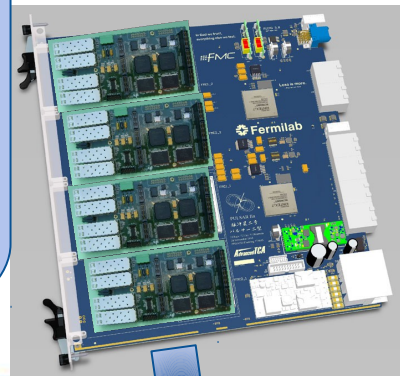
The FTK system - components & relation to rest of TDAQ



Data in FTK from Si RODs in dual-output card



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



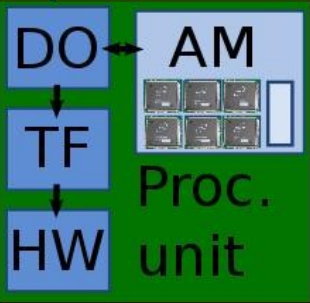
Pixels & SCT

RODs

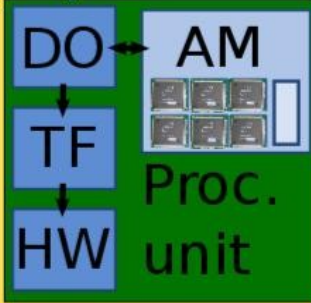
Data Formatter

FTK

Cluster Finding



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



Second Stage Fit (4 brds)

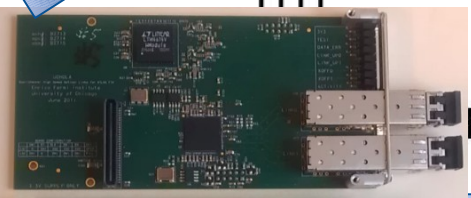
Track Data ROB

FLIC

Raw Data ROBs

HLT Processing

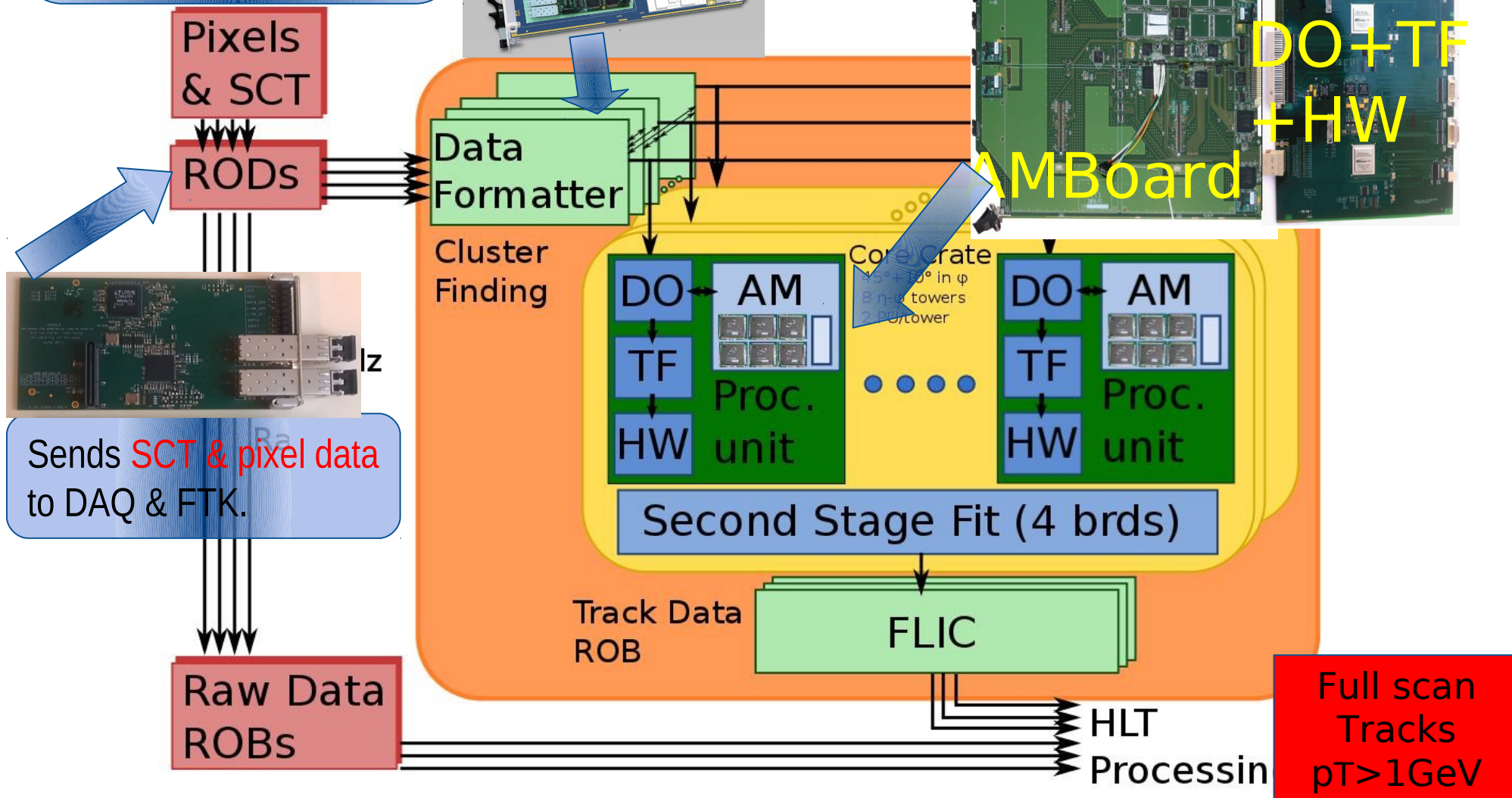
Full scan Tracks
 $p_T > 1\text{GeV}$



Sends SCT & pixel data to DAQ & FTK.

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.



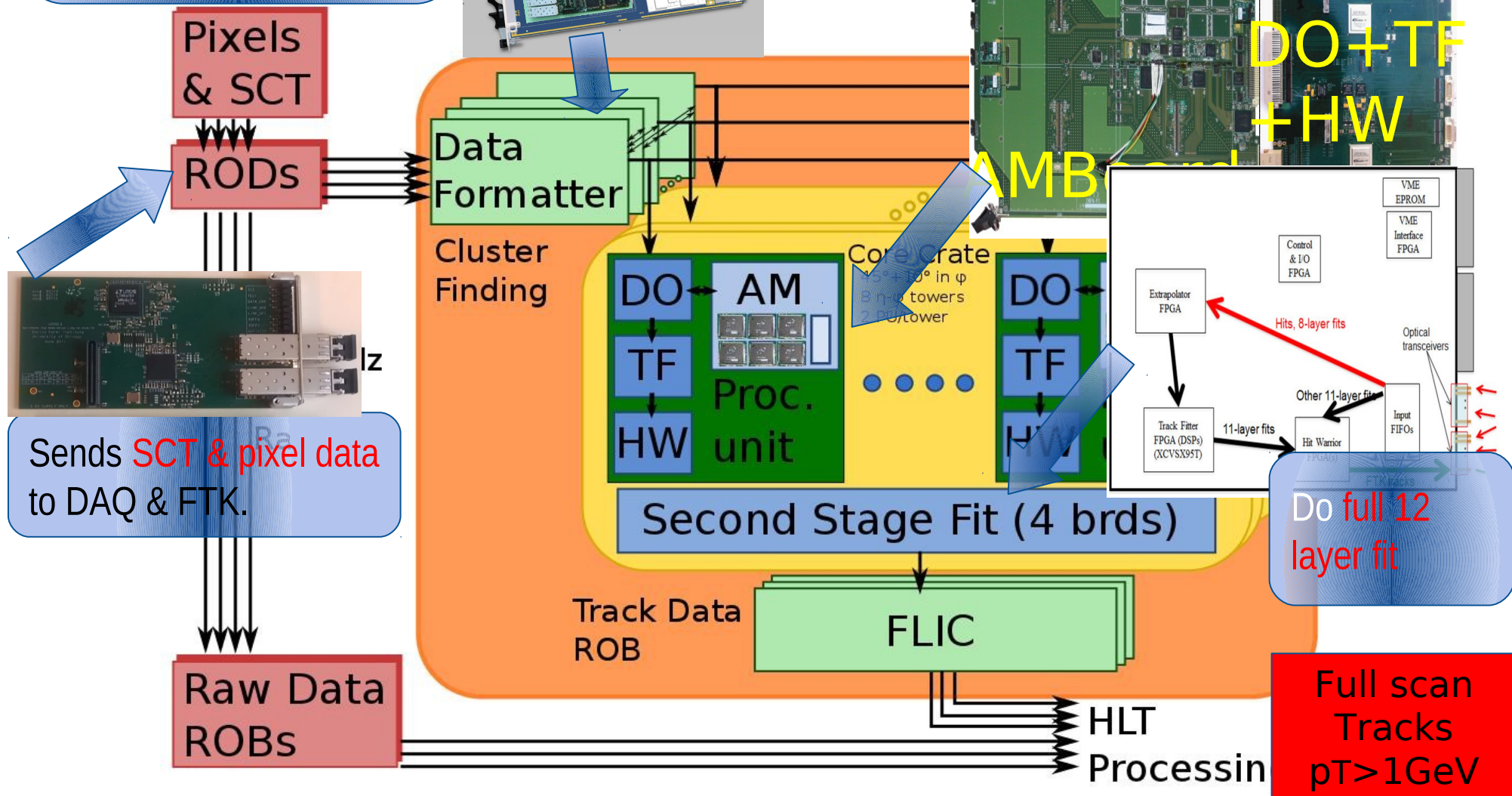
Sends SCT & pixel data to DAQ & FTK.

Raw Data ROBs

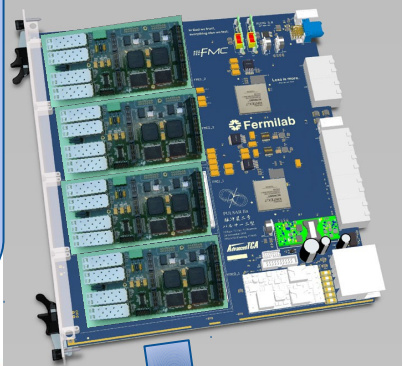
Full scan Tracks $p_T > 1 \text{ GeV}$

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

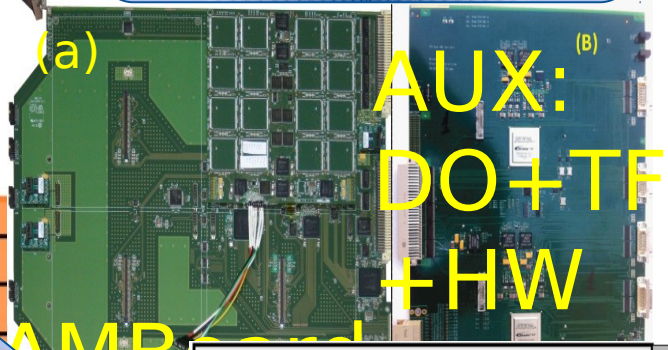
128 PUs do pattern matching and the 1st stage track fitting.



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



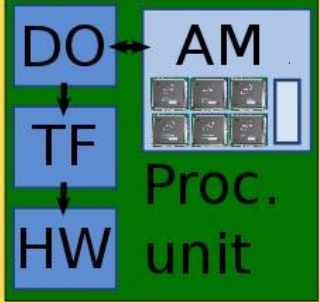
128 PUs do pattern matching and the 1st stage track fitting.



Pixels & SCT
 ↓↓↓↓
 RODs

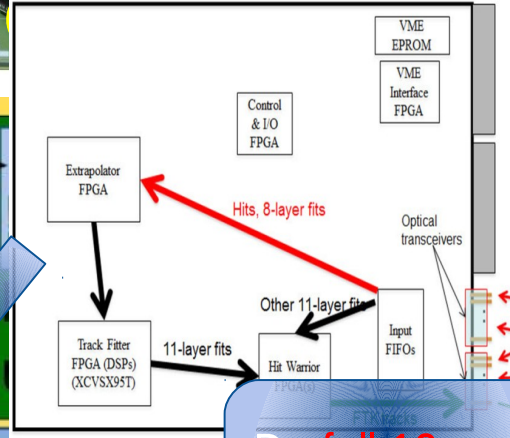
Data Formatter

Cluster Finding

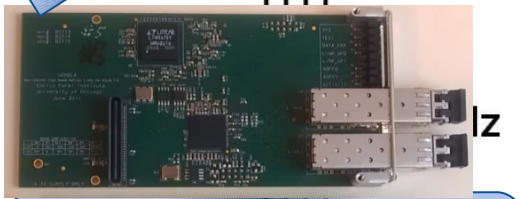


Core Crate
 $35^\circ \times 30^\circ$ in ϕ
 8 η - ϕ towers
 2 PU/tower

AMB



Do full 12 layer fit



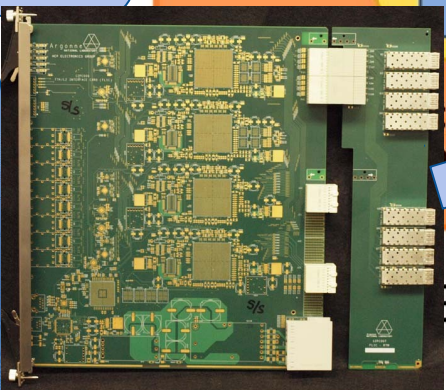
Sends SCT & pixel data to DAQ & FTK.

Second Stage Fit (4 brds)

FLIC

Full scan Tracks $p_T > 1\text{GeV}$

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



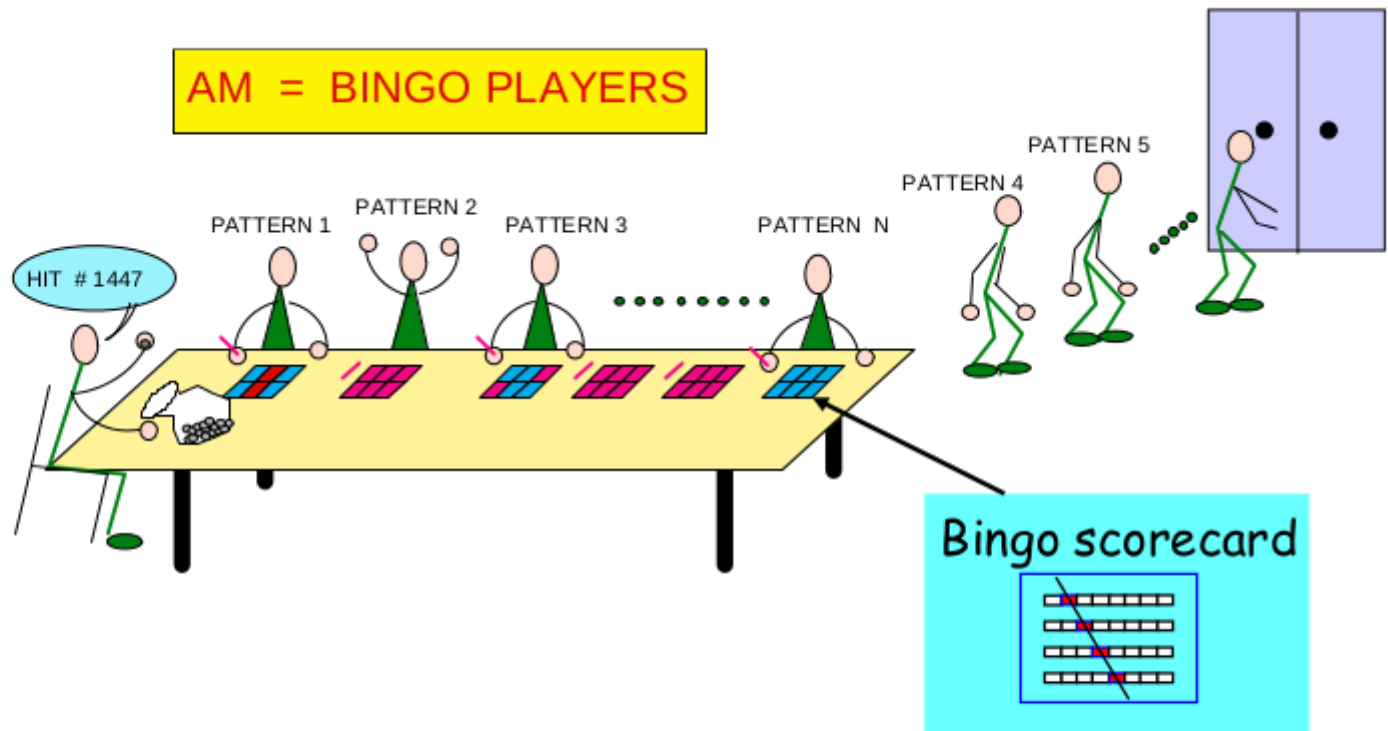
Track Data

HLT Processing

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

AM = hits compared to pre-computed patterns



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