Tracking in the Triggering System

the use of Associative Memory from CDF to the HL-LHC

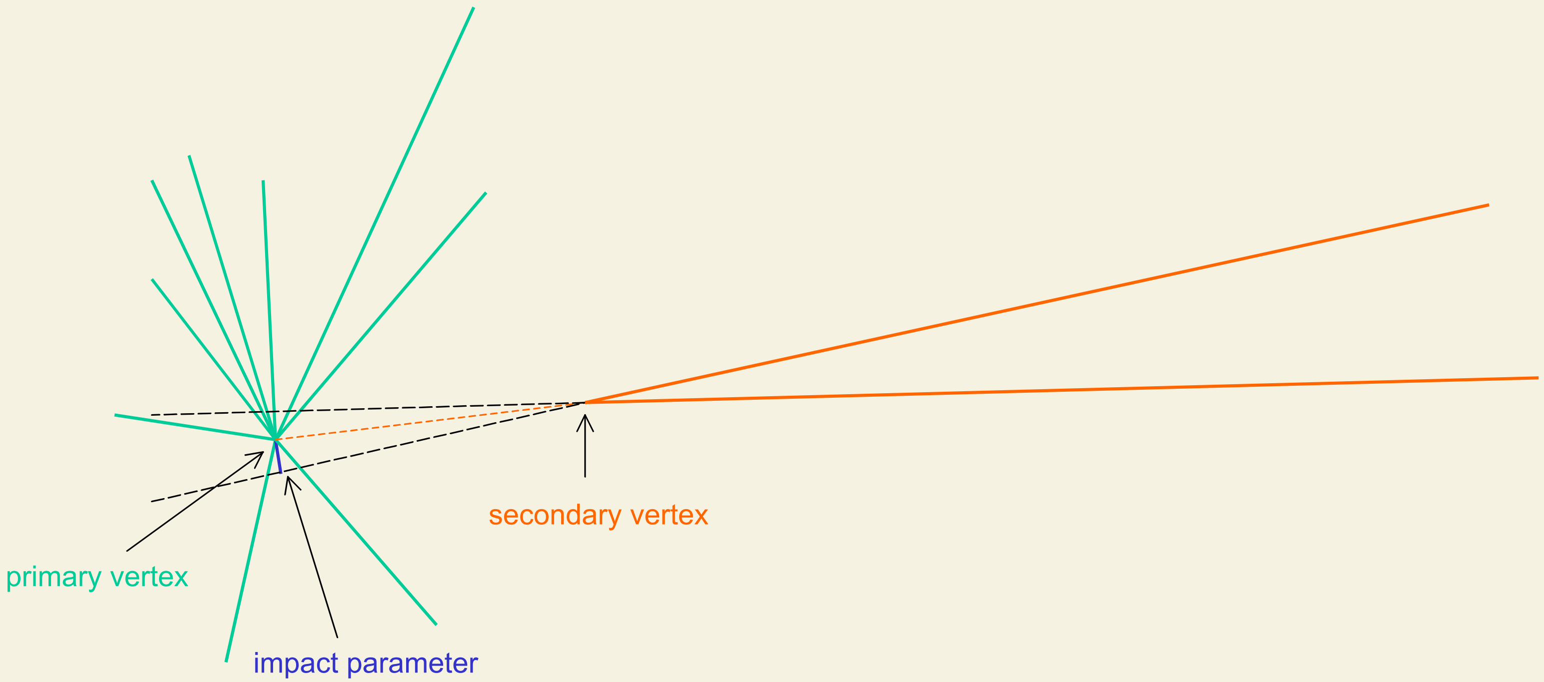
Luciano Ristori - October 19, 2016

The Associative Memory

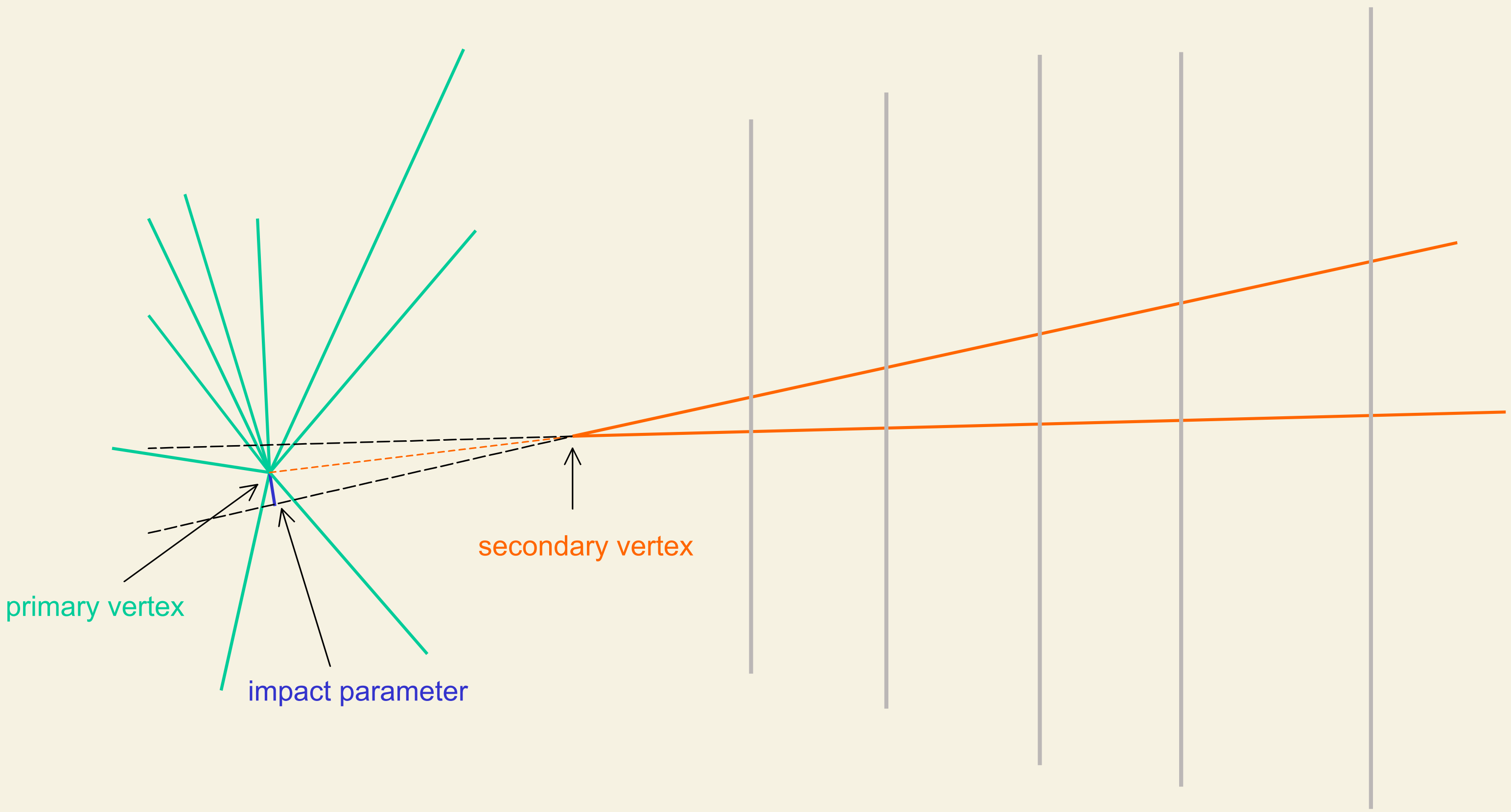
- The idea
- How does it work?
- A little history
- Recent optimizations
- Possible future developments

The idea

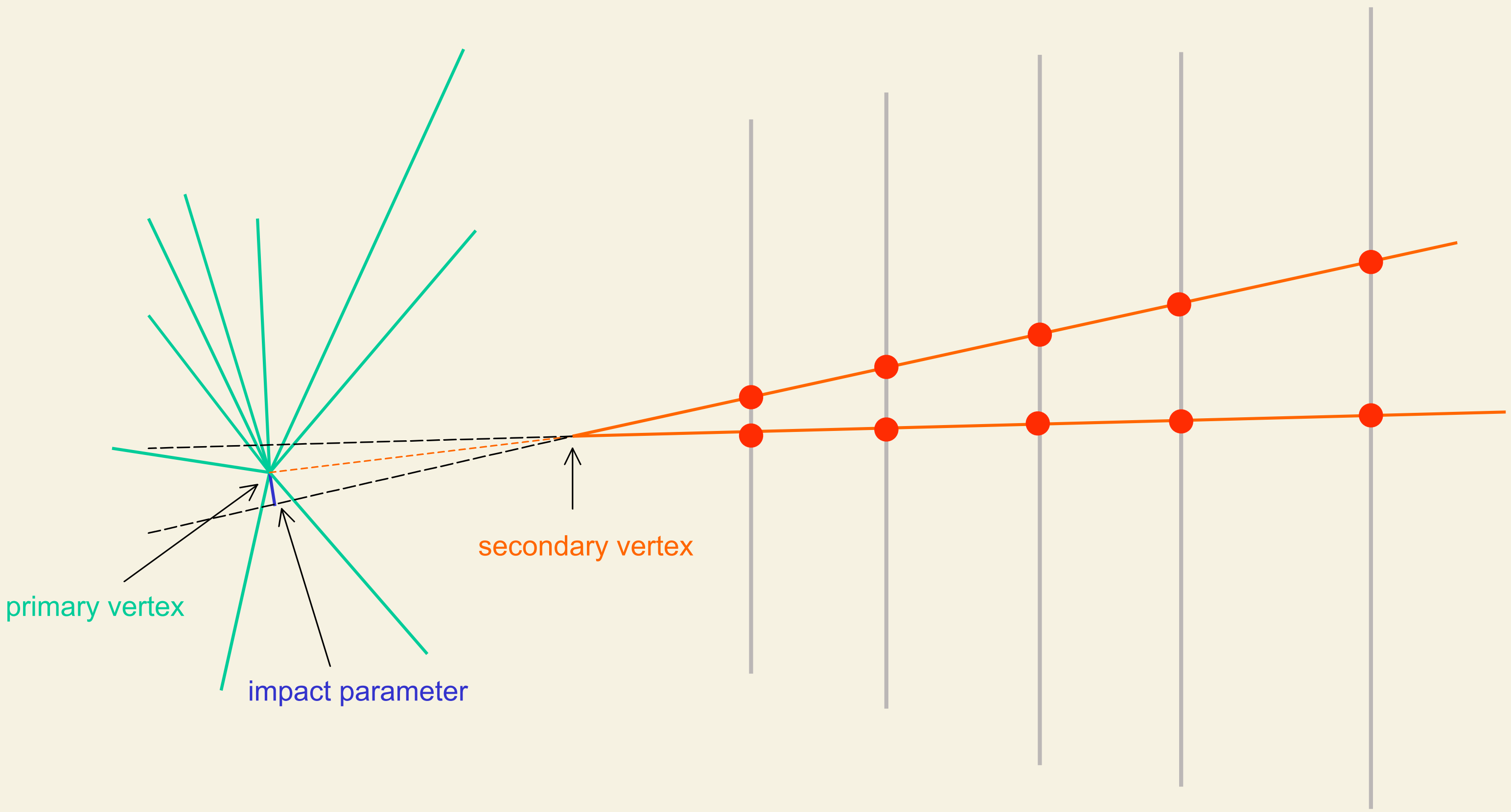
pattern recognition



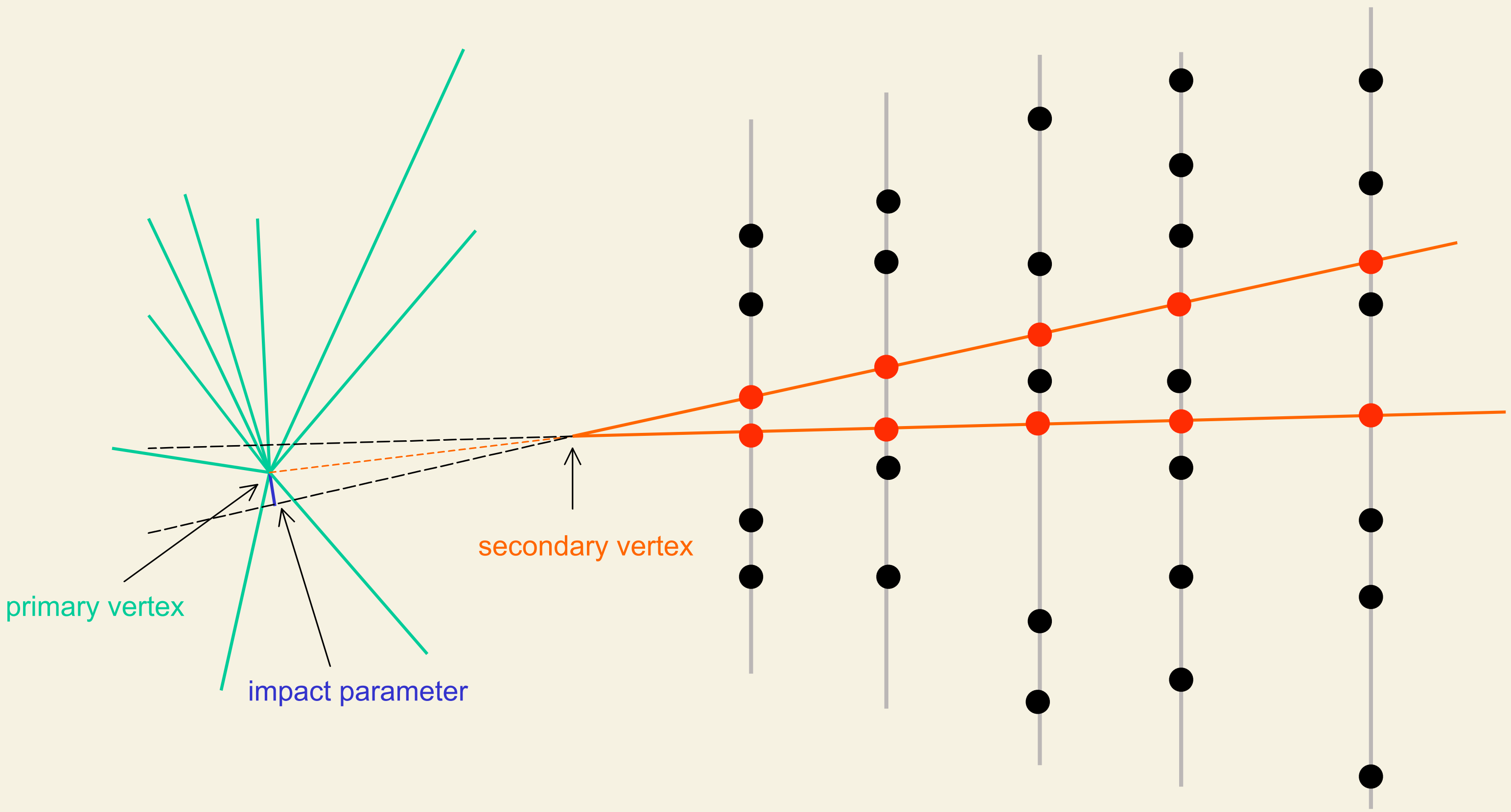
pattern recognition



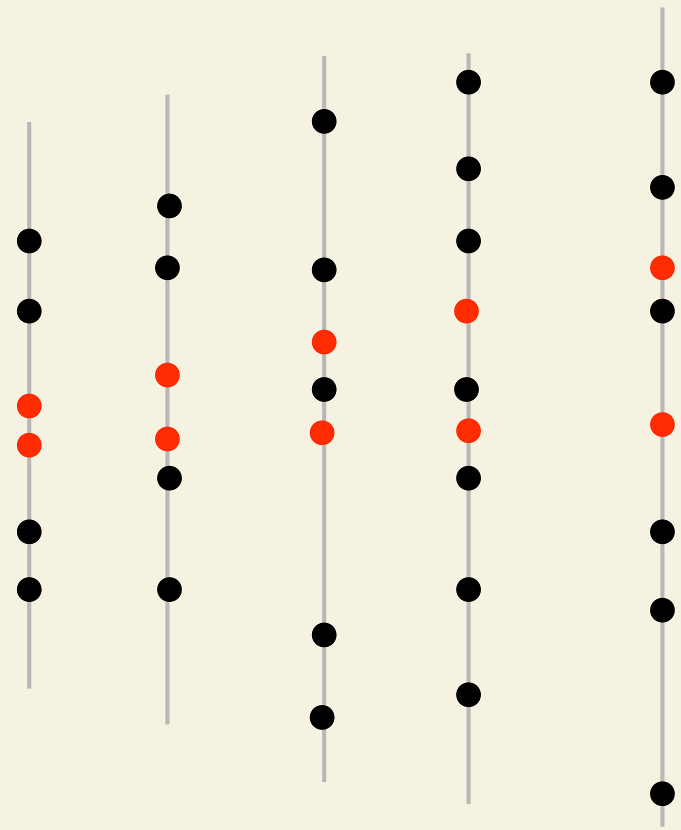
pattern recognition



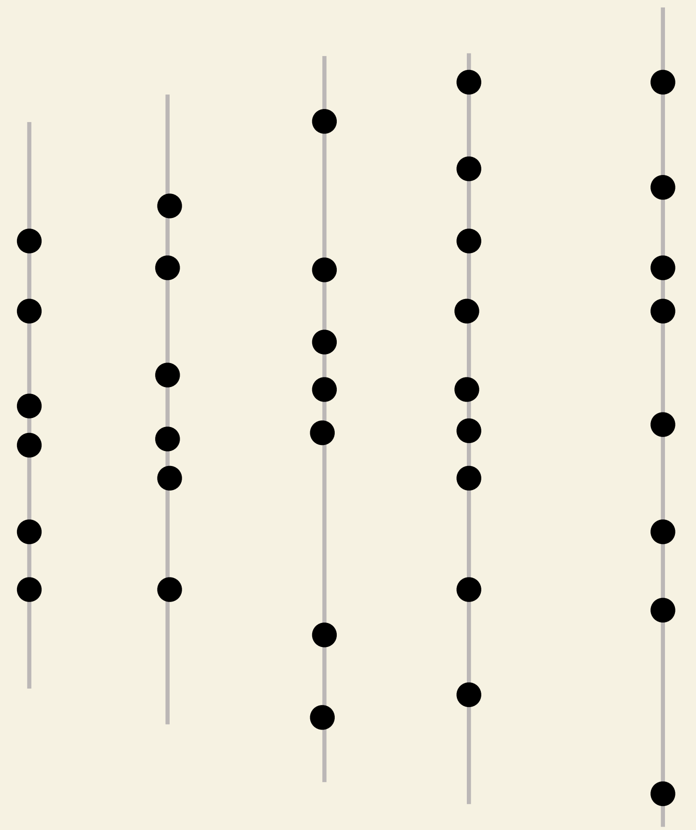
pattern recognition



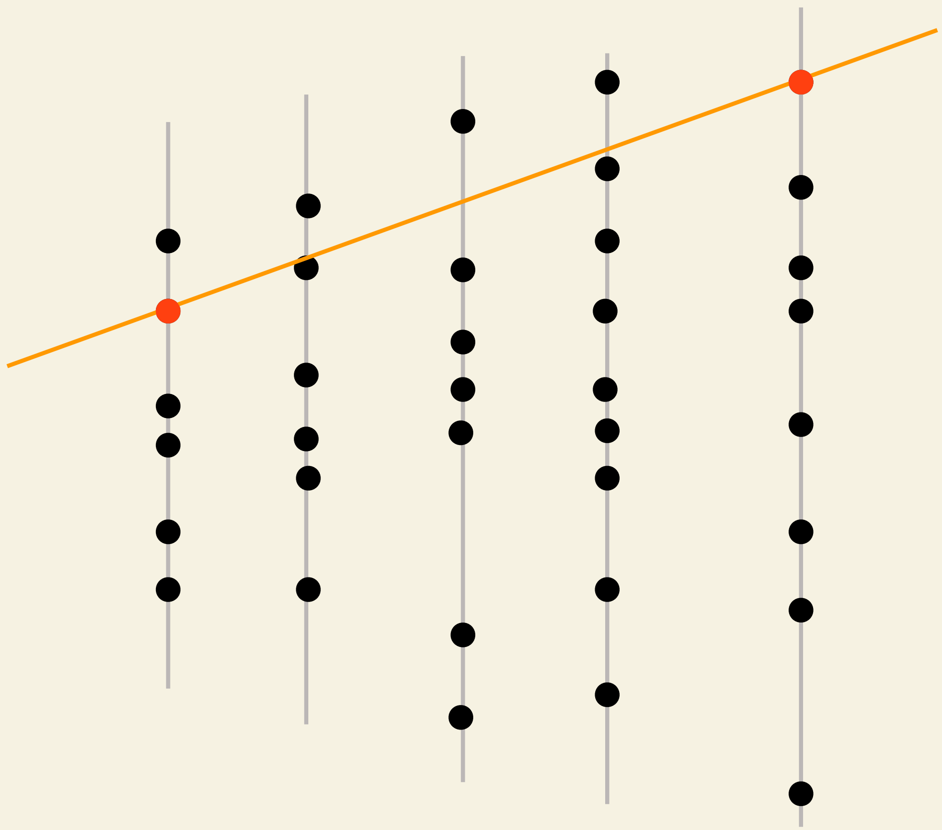
pattern recognition



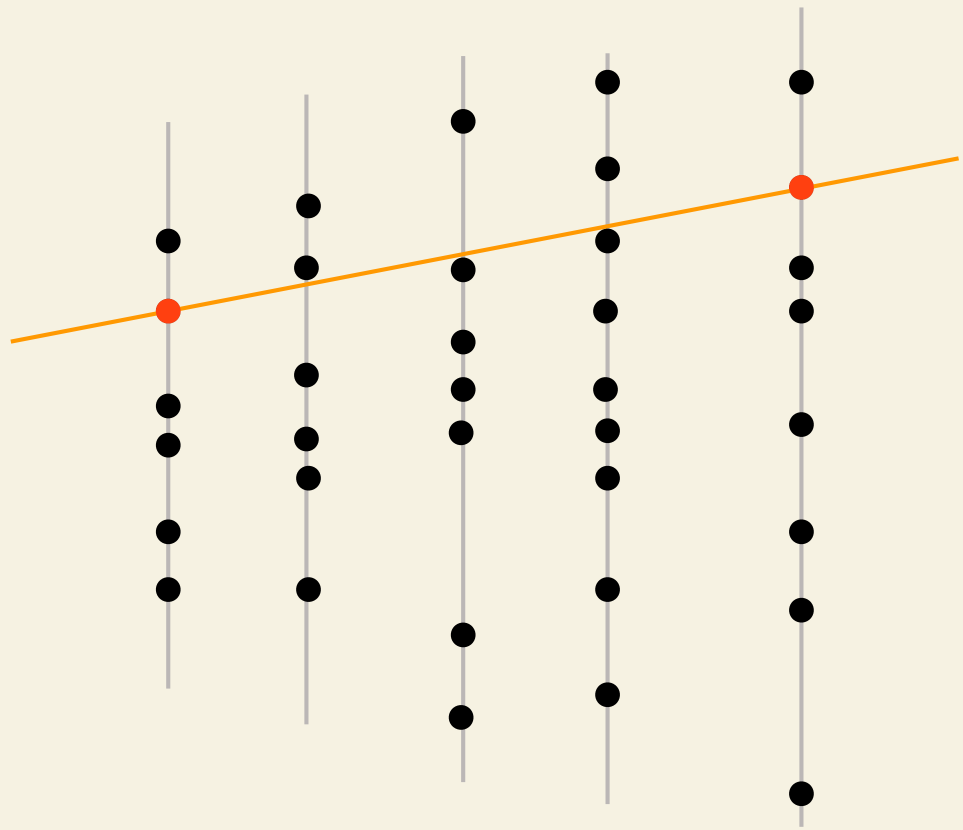
looking for the right combination of hits



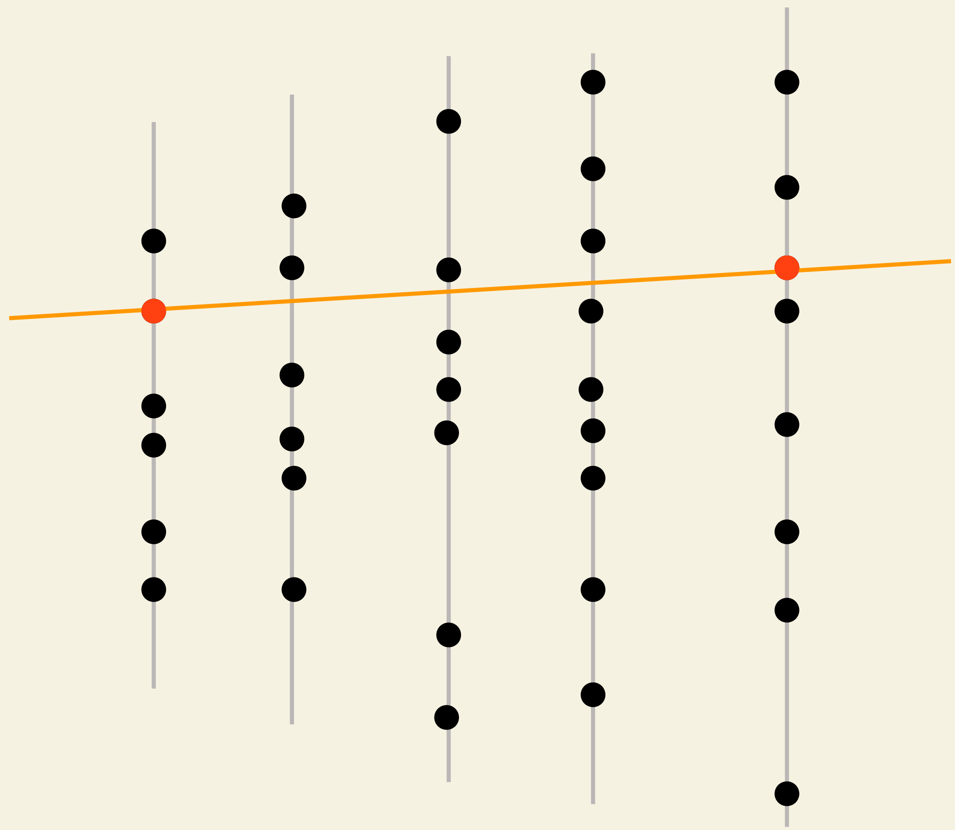
looking for the right combination of hits



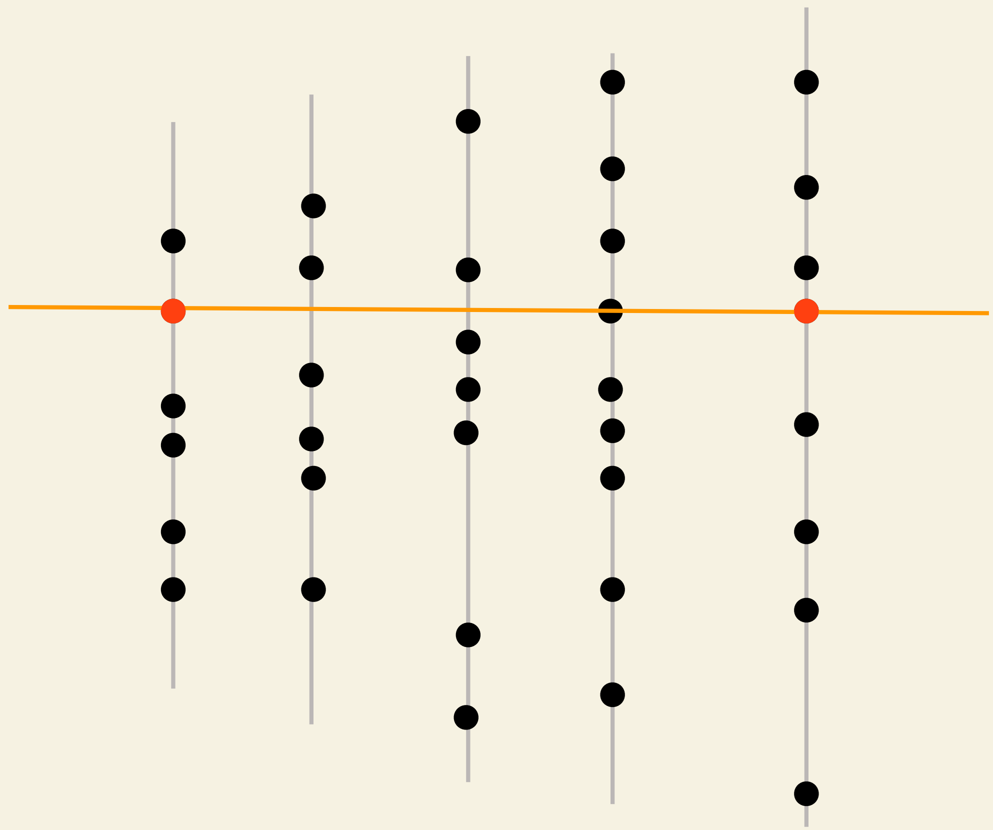
looking for the right combination of hits



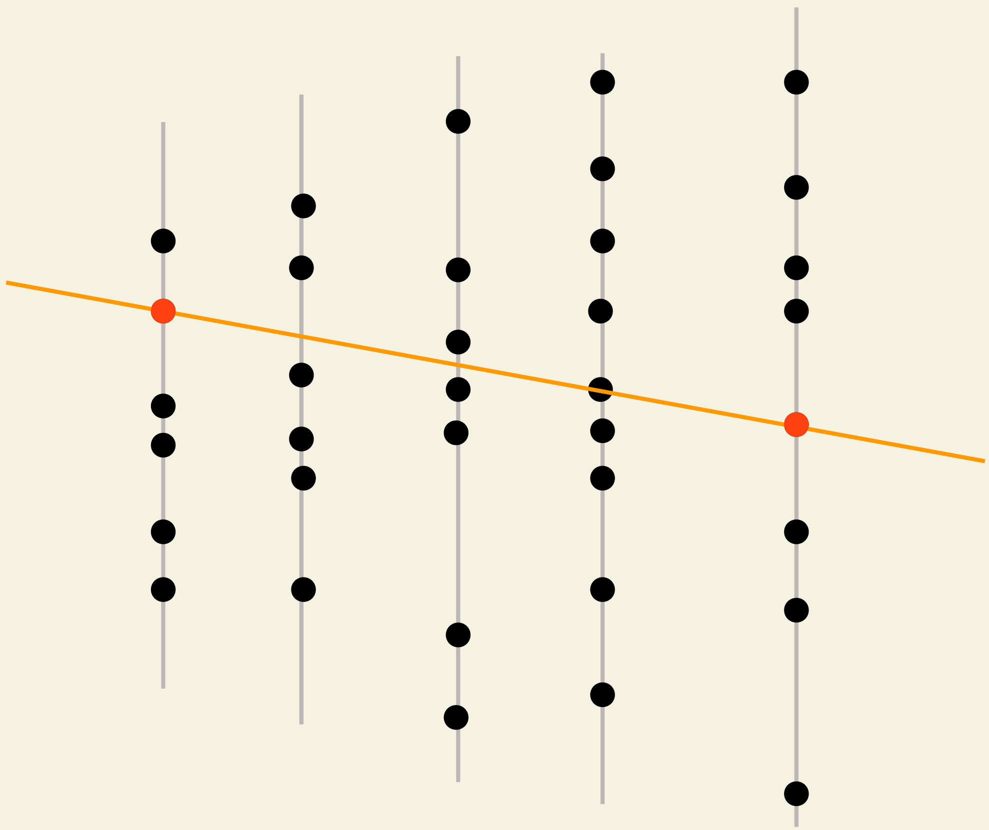
looking for the right combination of hits



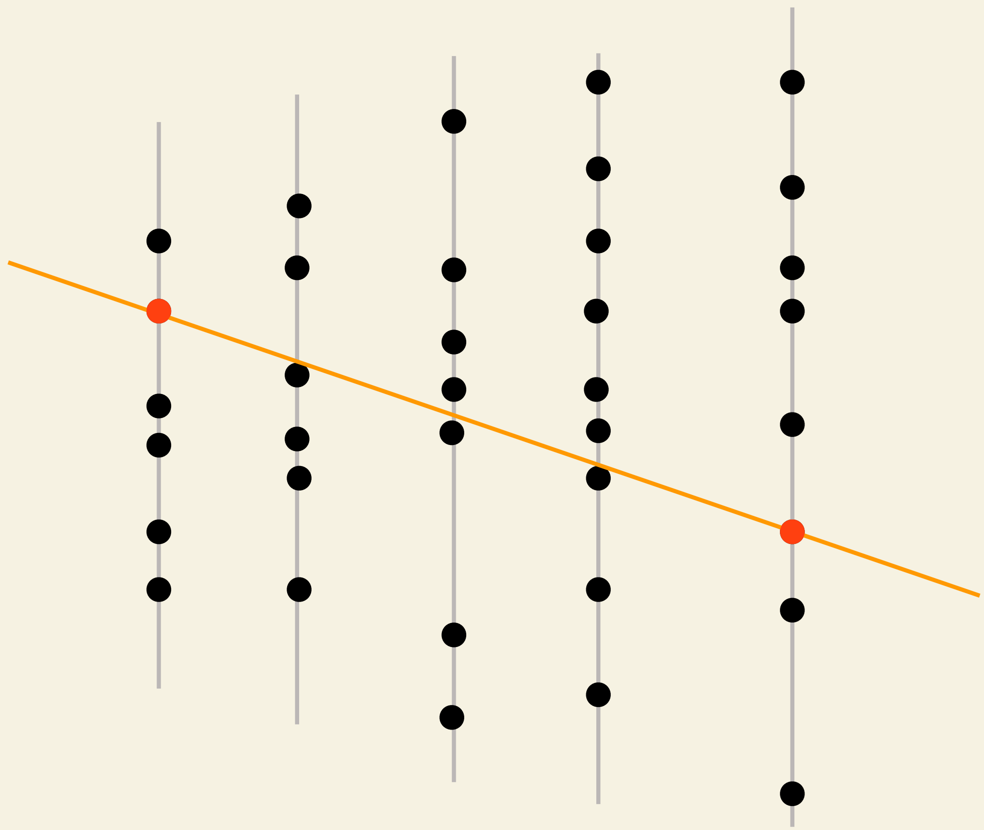
looking for the right combination of hits



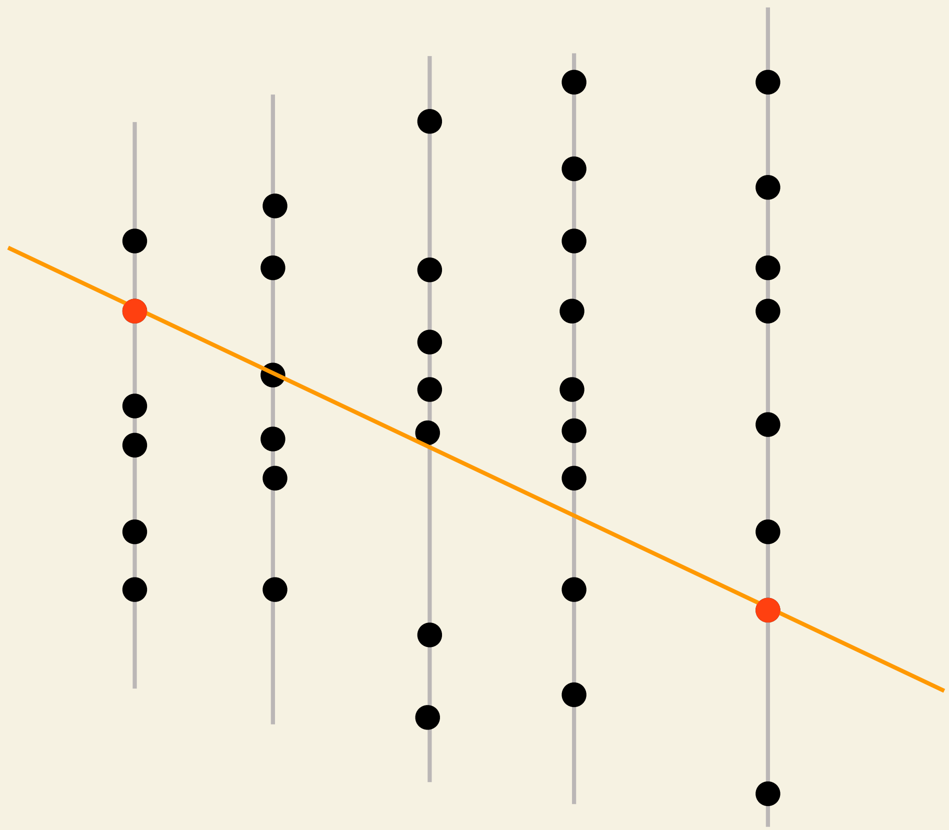
looking for the right combination of hits



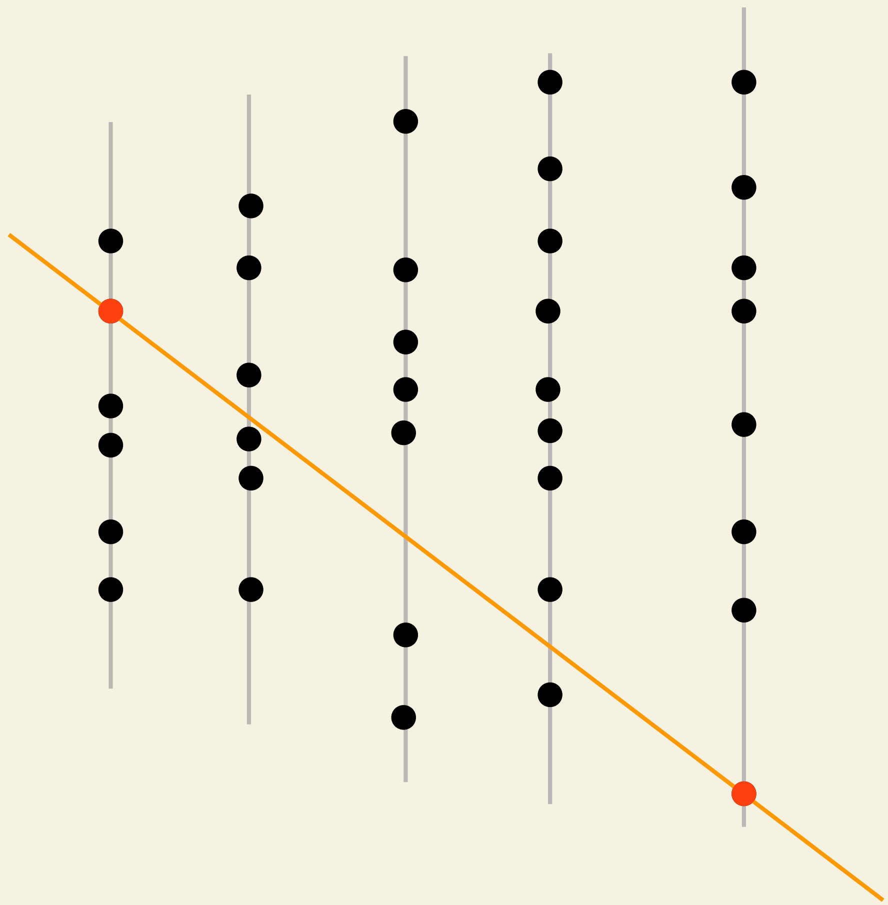
looking for the right combination of hits



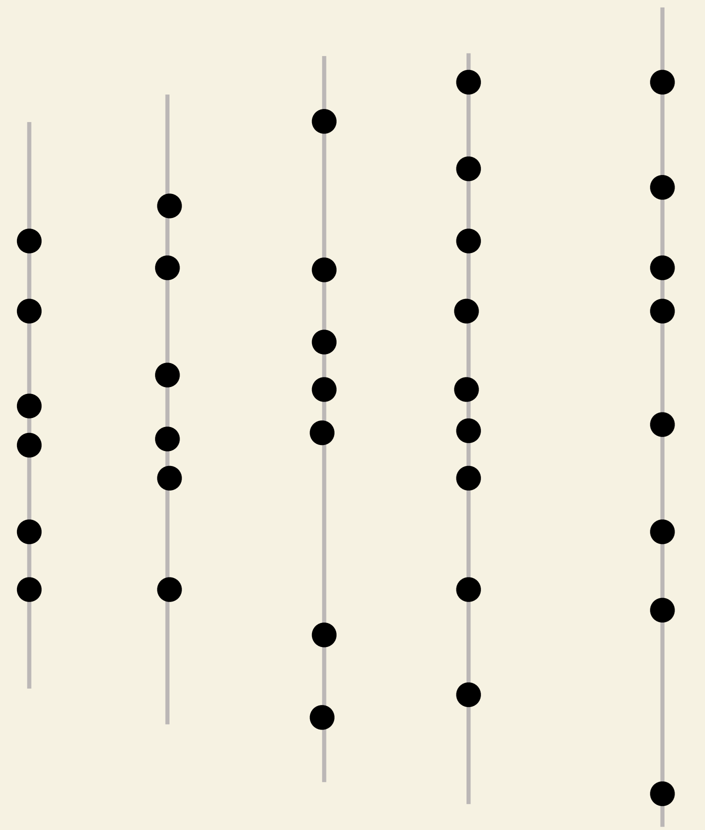
looking for the right combination of hits



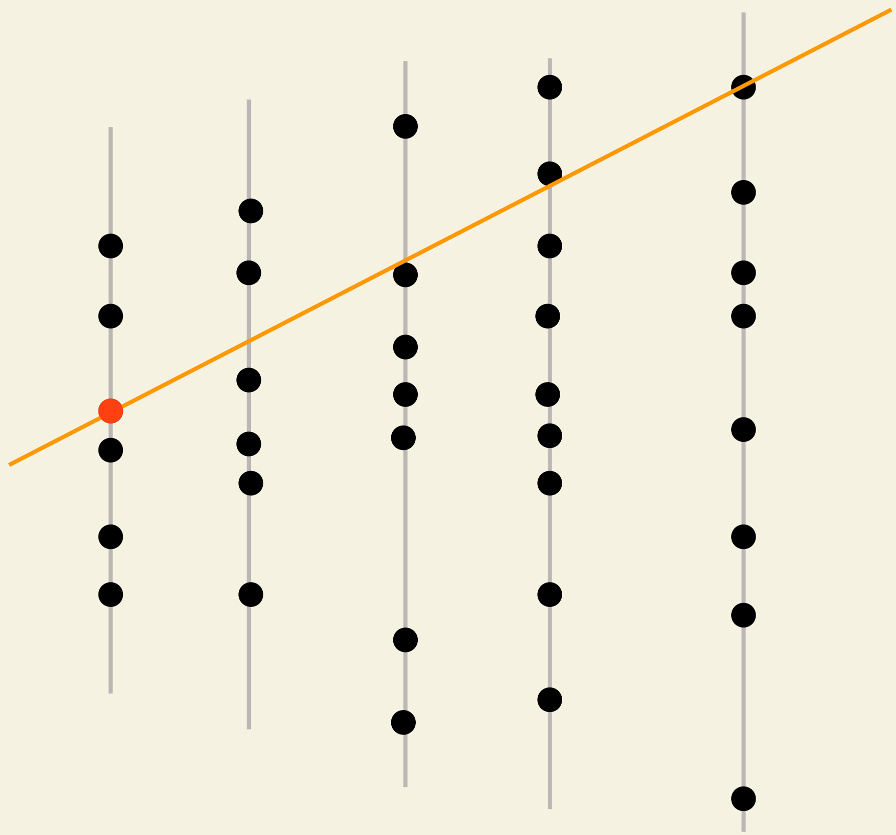
looking for the right combination of hits



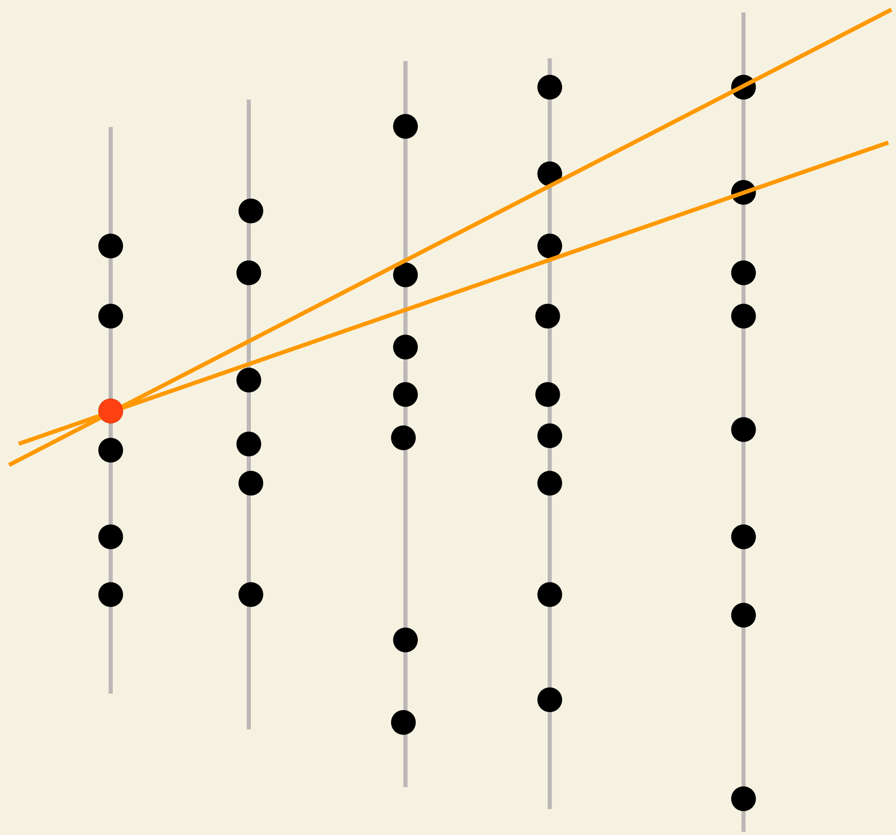
looking for the right combination of hits



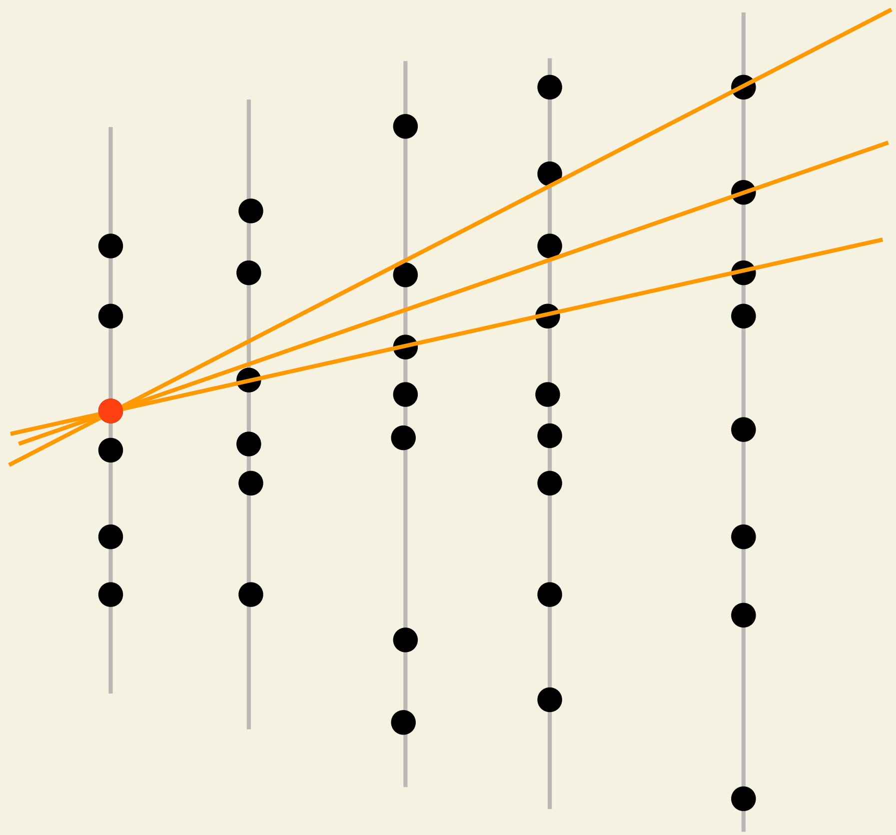
looking for the right combination of hits



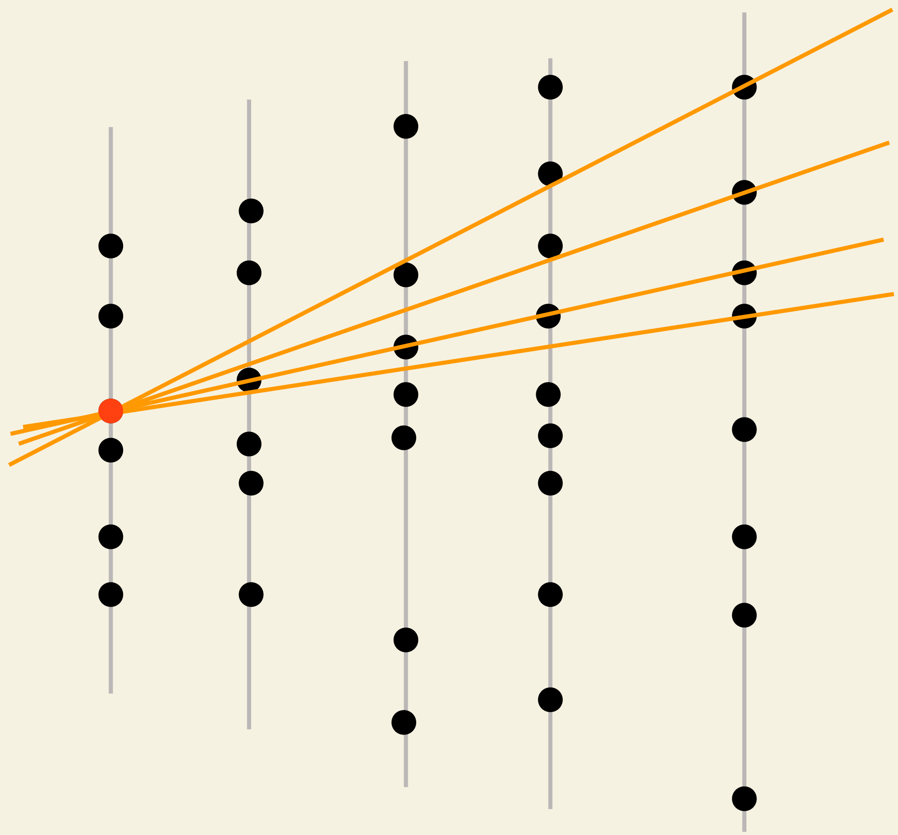
looking for the right combination of hits



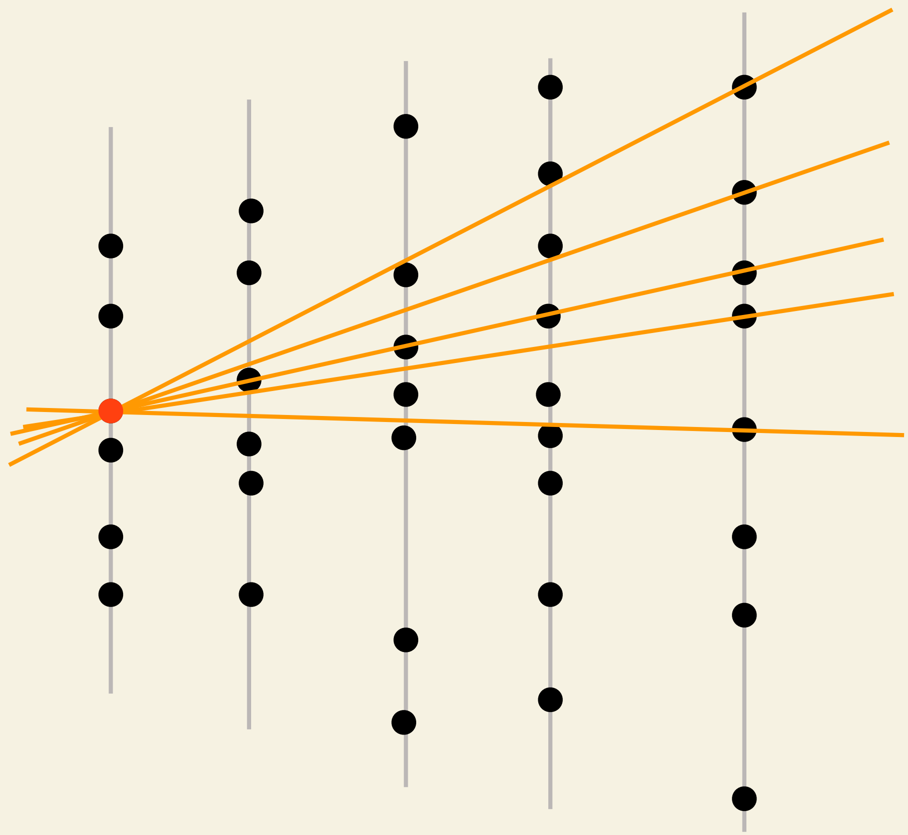
looking for the right combination of hits



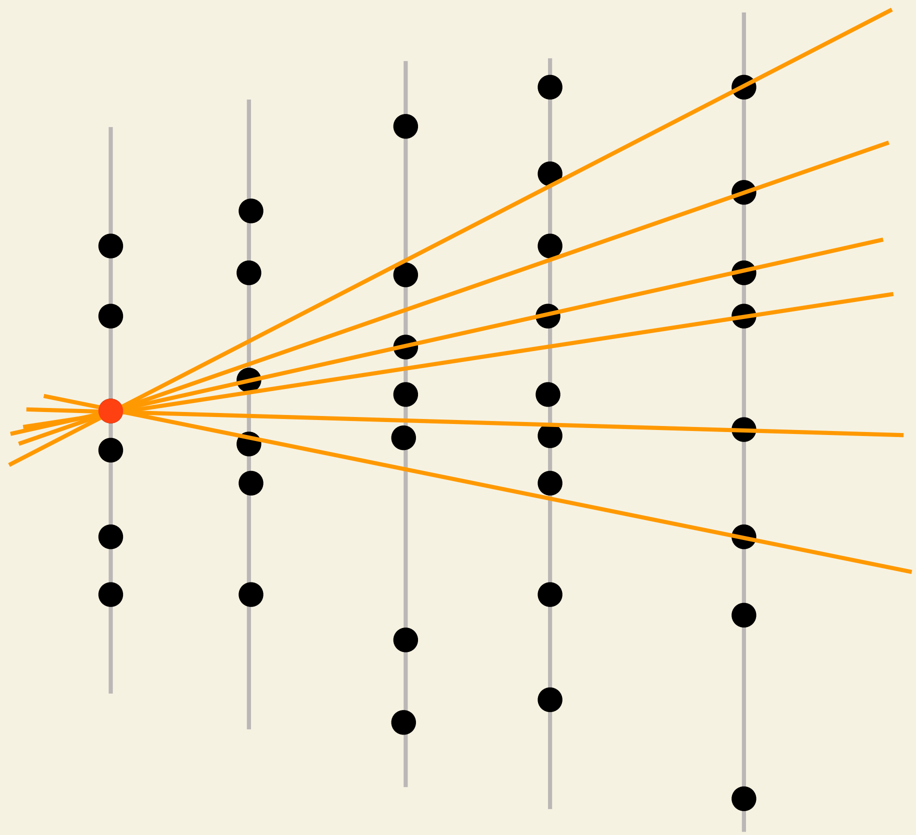
looking for the right combination of hits



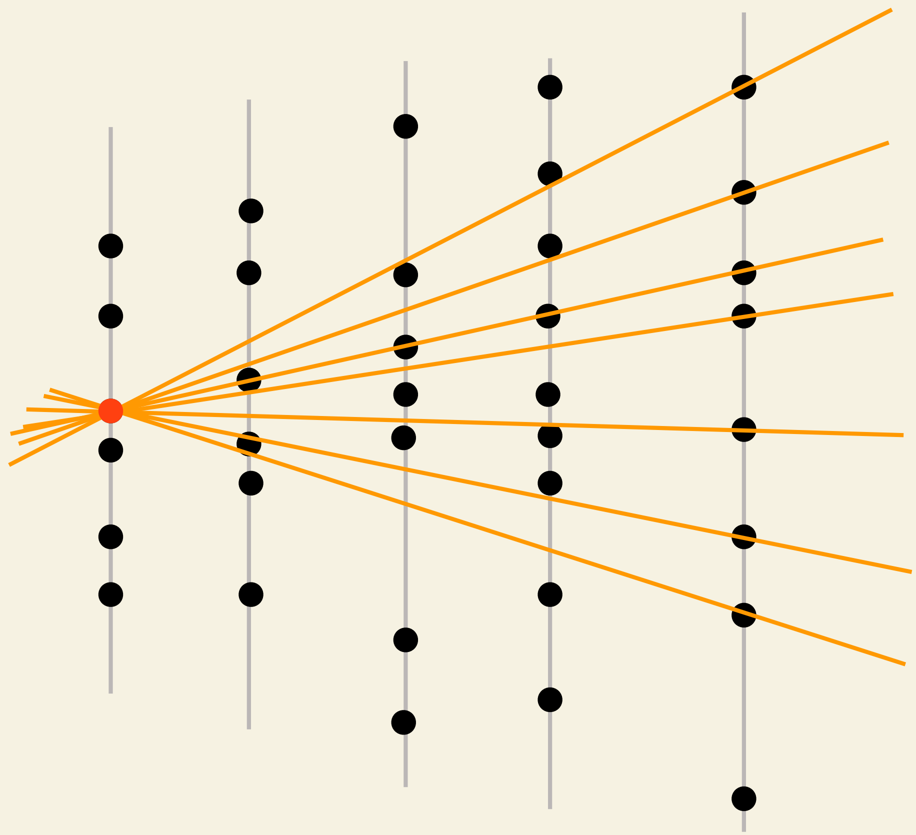
looking for the right combination of hits



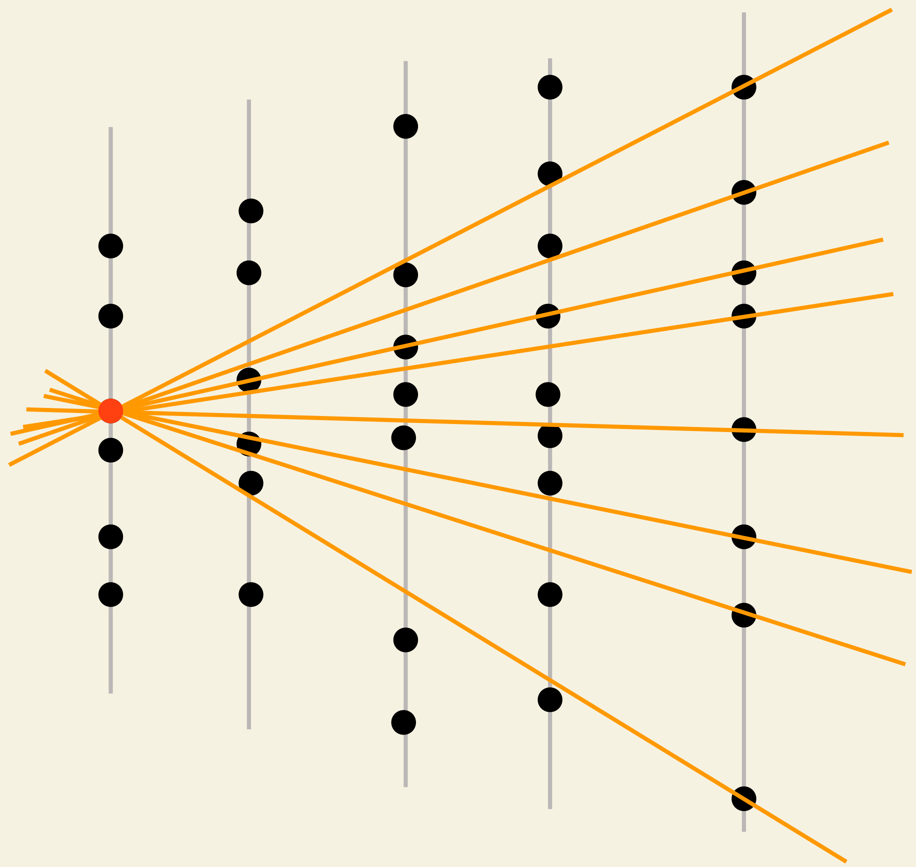
looking for the right combination of hits



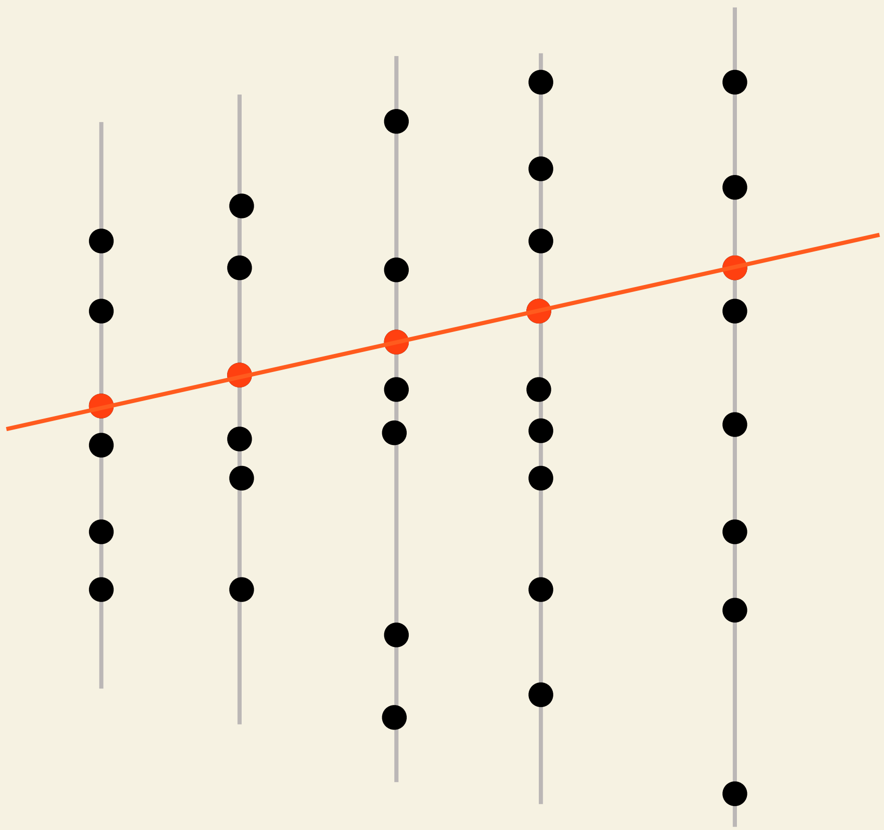
looking for the right combination of hits



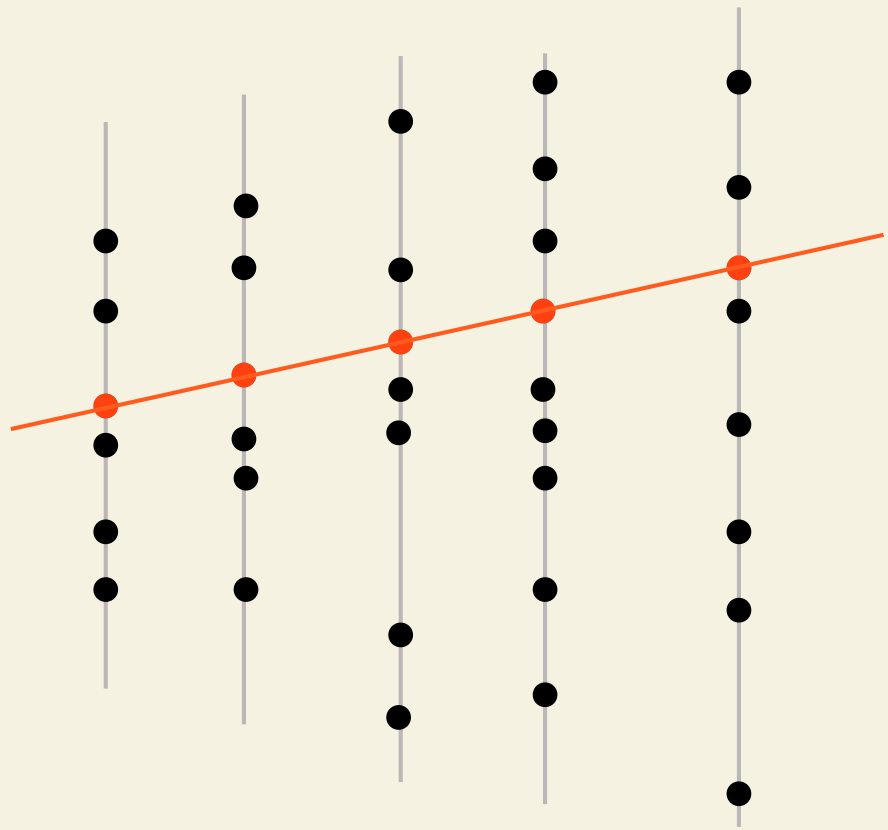
looking for the right combination of hits



looking for the right combination of hits

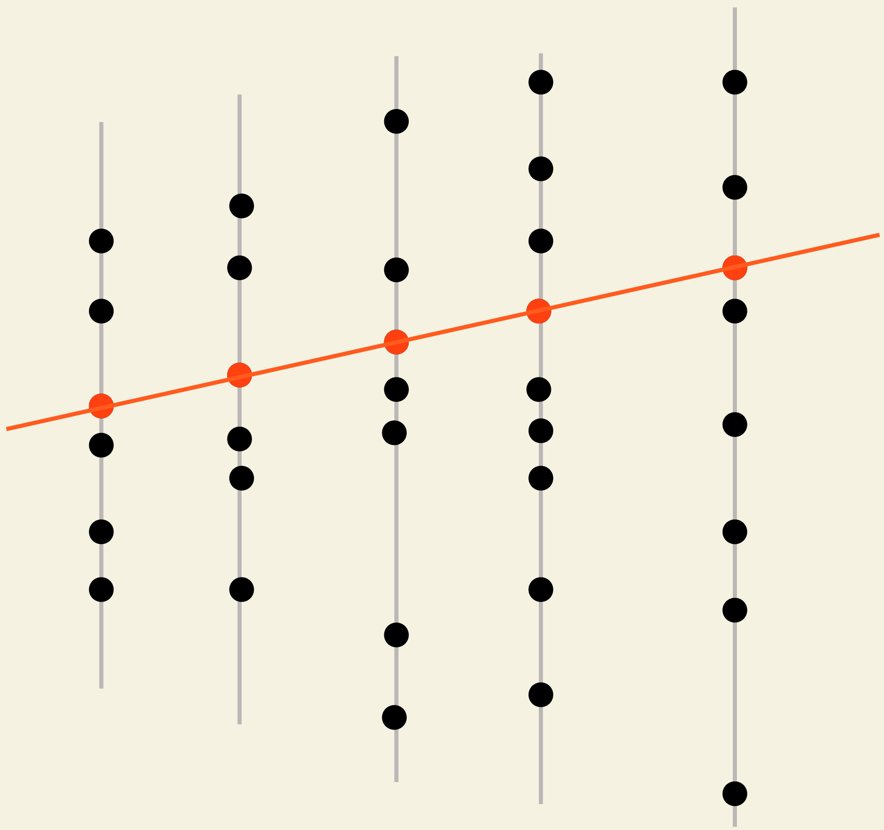


looking for the right combination of hits



number of trials $\propto$ (number of hits)^3

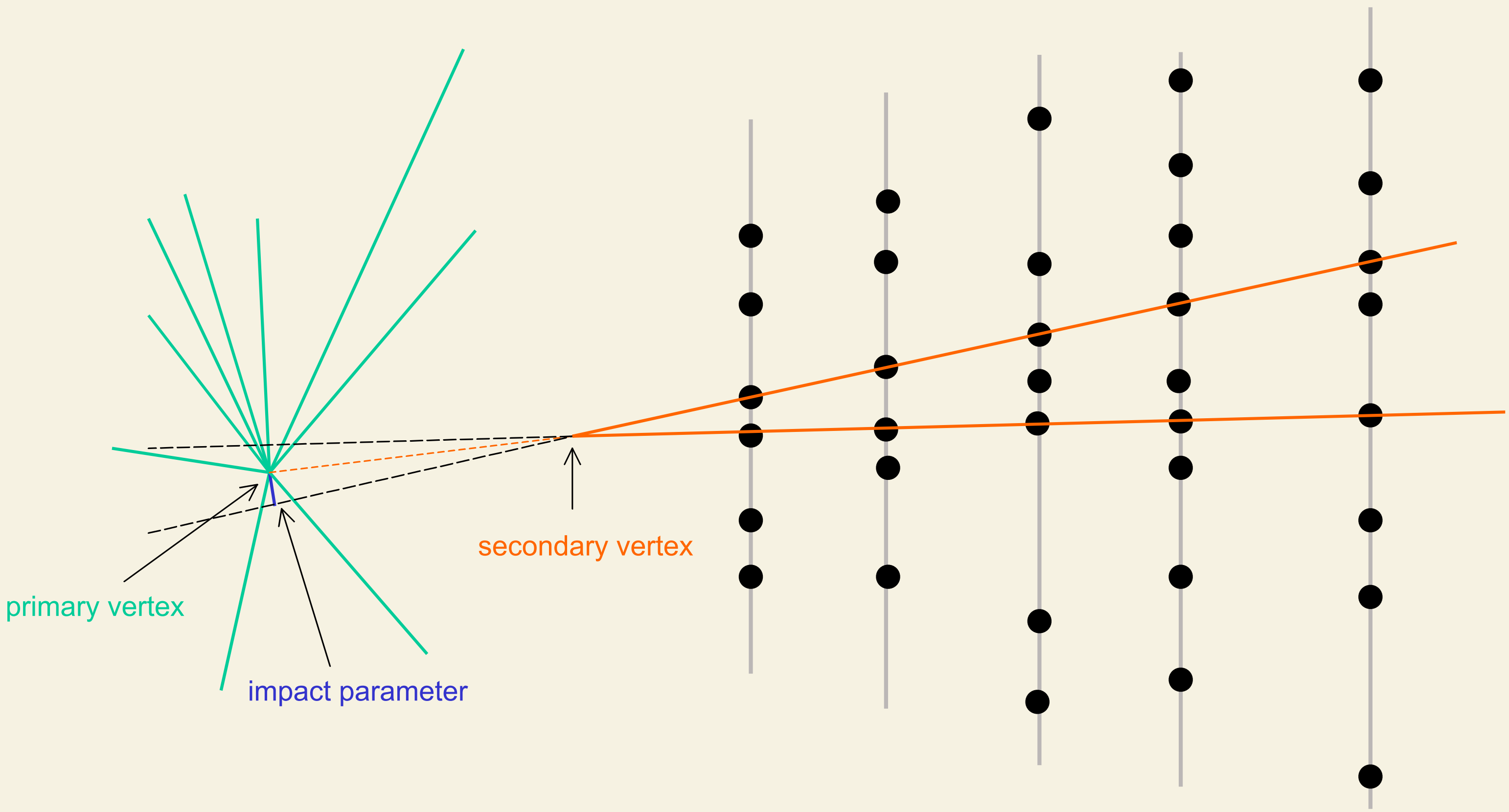
looking for the right combination of hits



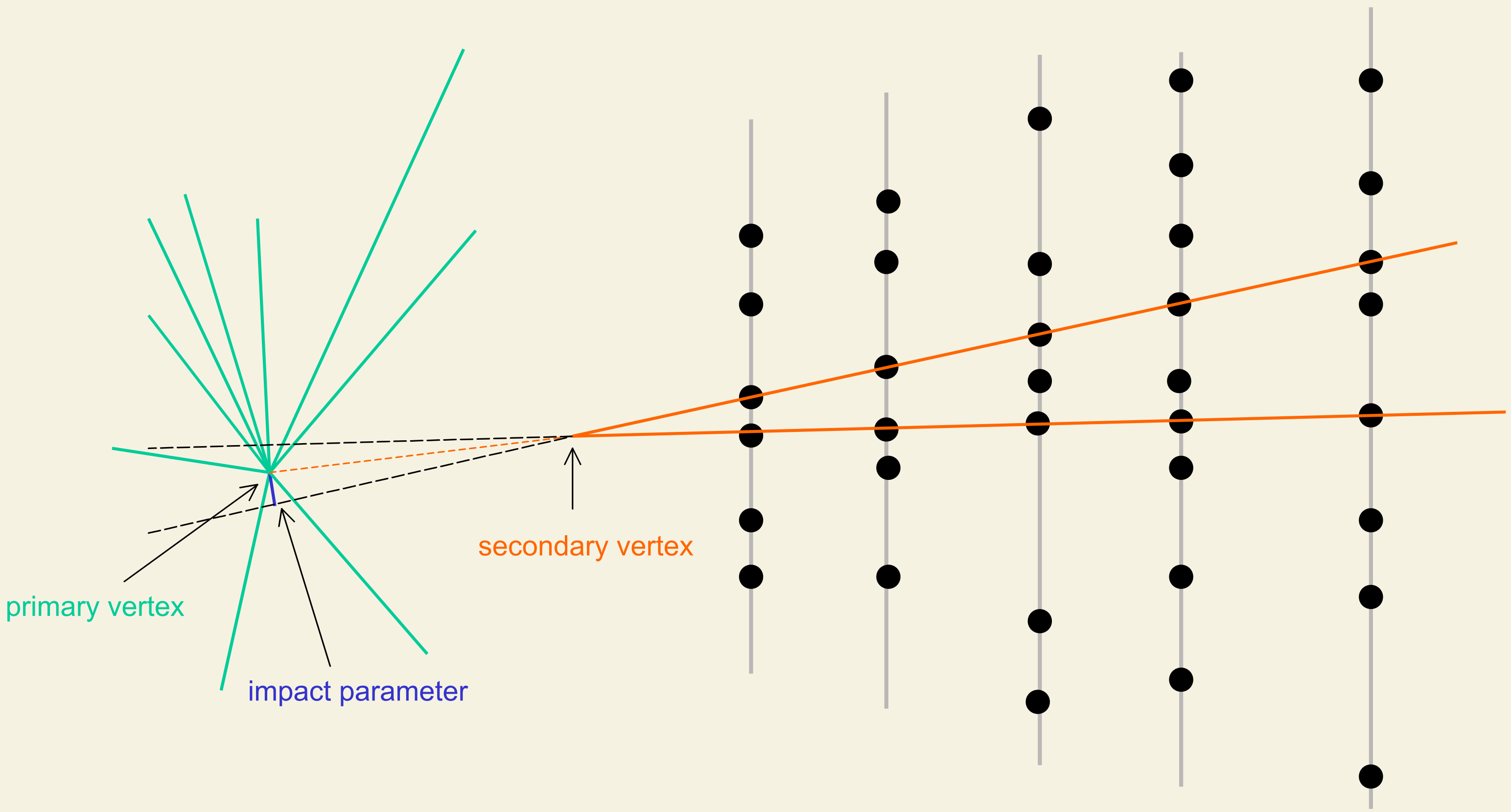
number of trials $\propto$ (number of hits)^3

straight line = 2 degrees of freedom

pattern recognition



pattern recognition



computationally intensive

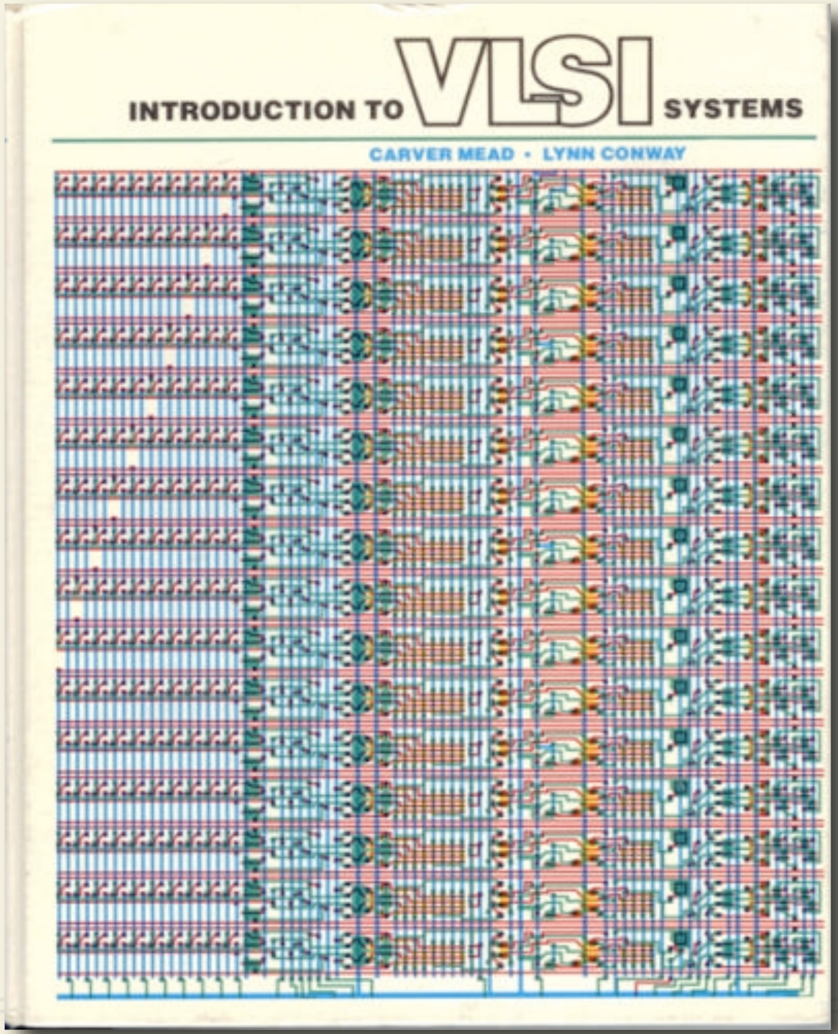
typically solved by trial and error

execution time typically goes as 3rd or 4th power of occupancy

It all started around 1986

Very Large Scale Integration
the revolution

Carver Mead & Lynn Conway

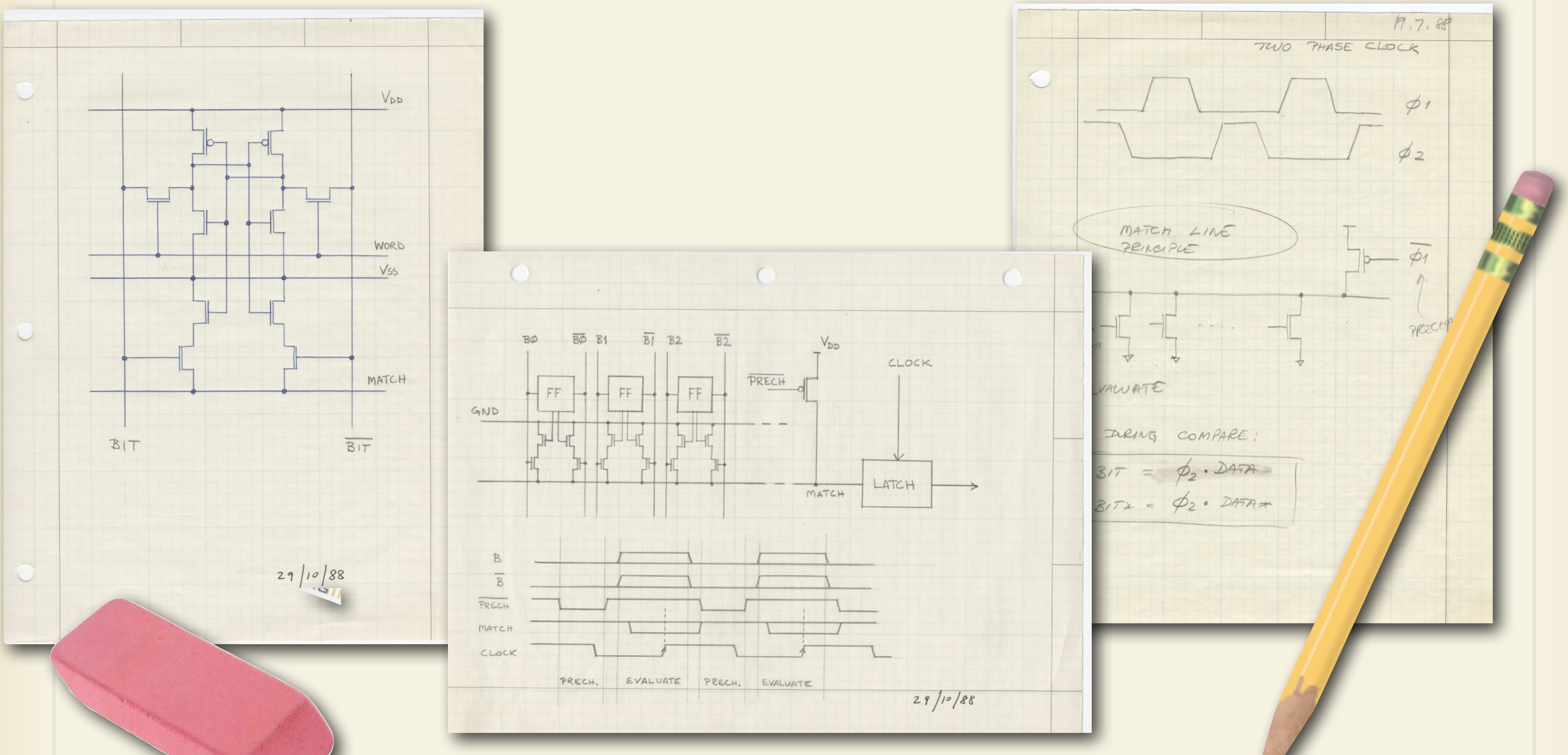


in the ‘80s the technology of VLSI design becomes available to the universities and to small research projects

need for speed

- Our main goal is speed
- Many hit combinations to be examined in sequence
  - → long execution time
- In principle **all combinations can be examined in parallel**
  - many processing unit each one checking a single combination (a very simple task)
  - not **all** combinations, only **valid** combinations
- For finite detector resolution the number of valid hit combinations is finite. Is it tractable?
  - processors are **very simple**
  - basically an AND circuit

it is at this point in time that we came up with the idea to use VLSI technology to solve the pattern recognition problem and reconstruct tracks in the detector in a very short time



October 1988: paper, pencil, eraser....

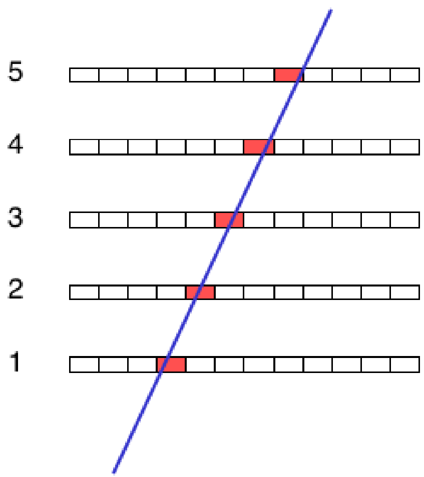
the Associative Memory

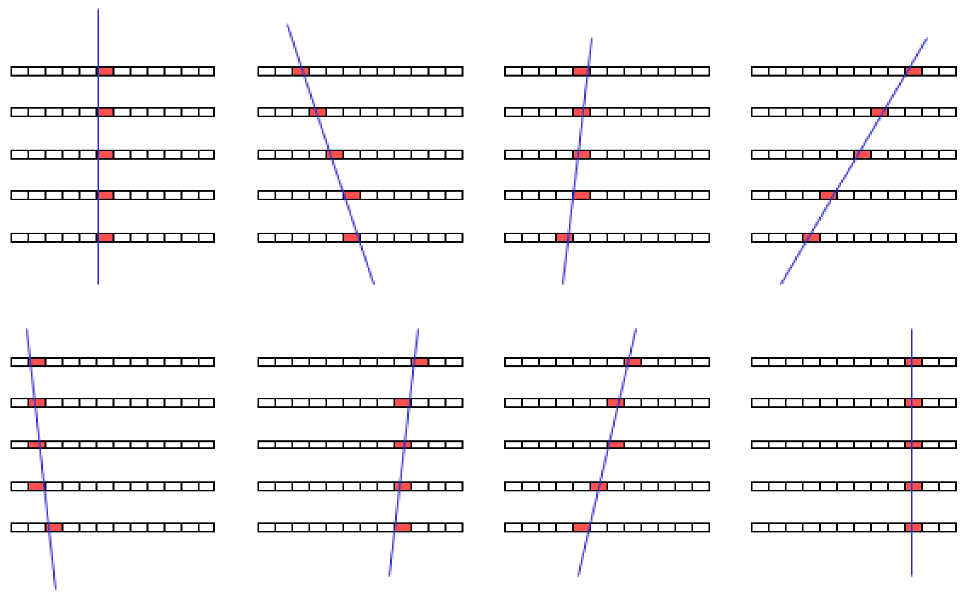
Building the “Pattern Bank”

SVT

Instead of looking for hit combinations such that *f(x1,x2,x3,…) = 0*

1. Build a database with all patterns corresponding to “good” tracks
2. Compare hits in each event with all patterns to find track candidates





In this example:
Straight lines, 5 layers, 12 bins/layer

⇨ Total number of patterns ~ $(12)^2 * (5-1) = 576$

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.

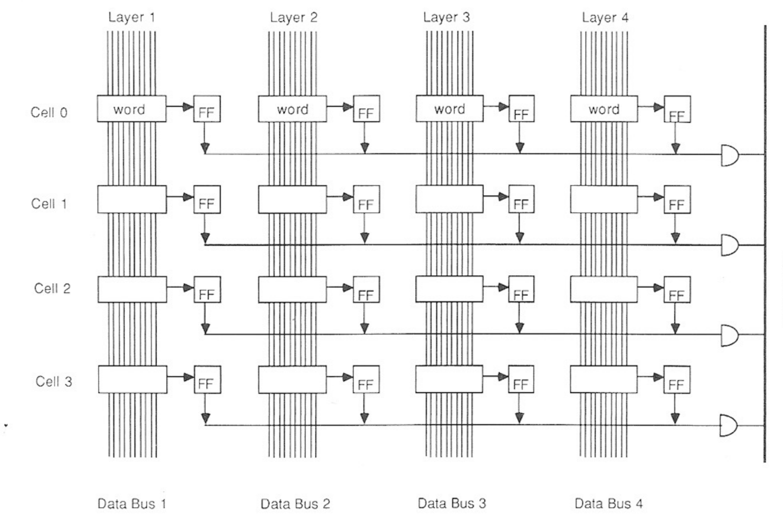


Fig. 3. Associative memory architecture.

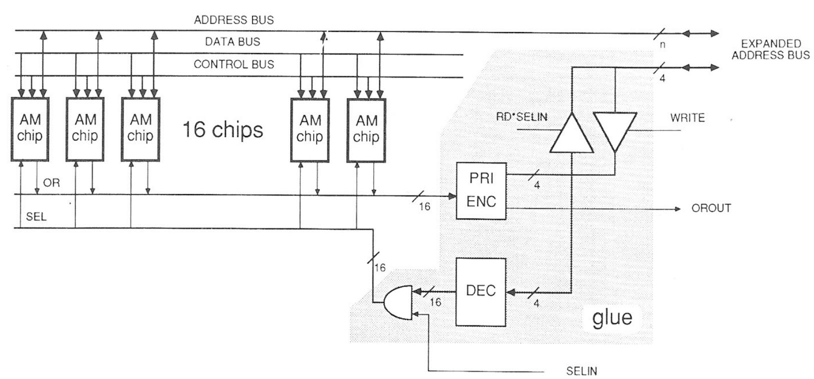
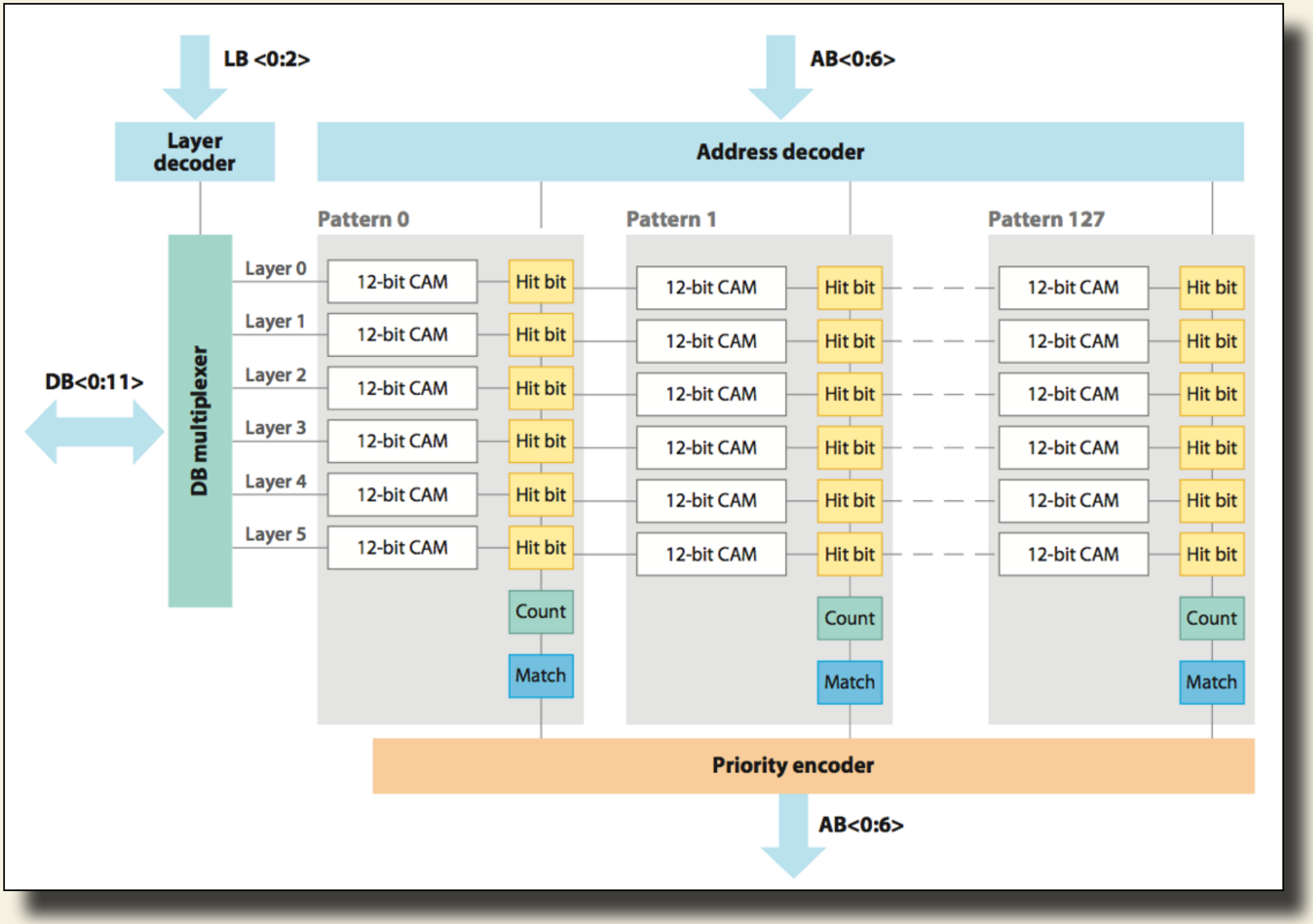


Fig. 5. 16 AM chips tied by the "glue".

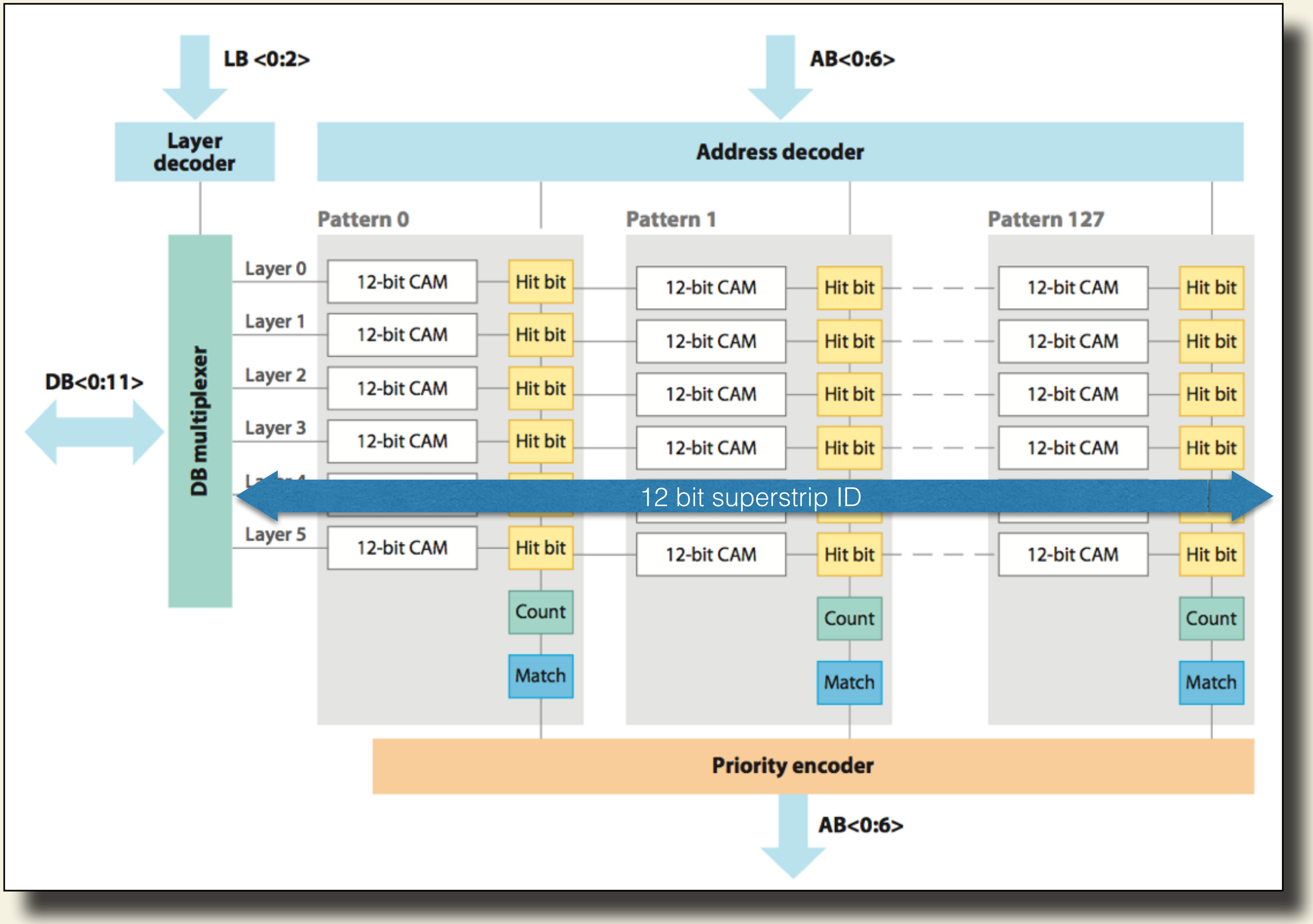
We discuss the architecture of a device based on the concept of *associative memory* designed to solve the track finding problem, typical of high energy physics experiments, in a time span of a few microseconds even for very high multiplicity events. This "machine" is implemented as a large array of 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.

How does it work?

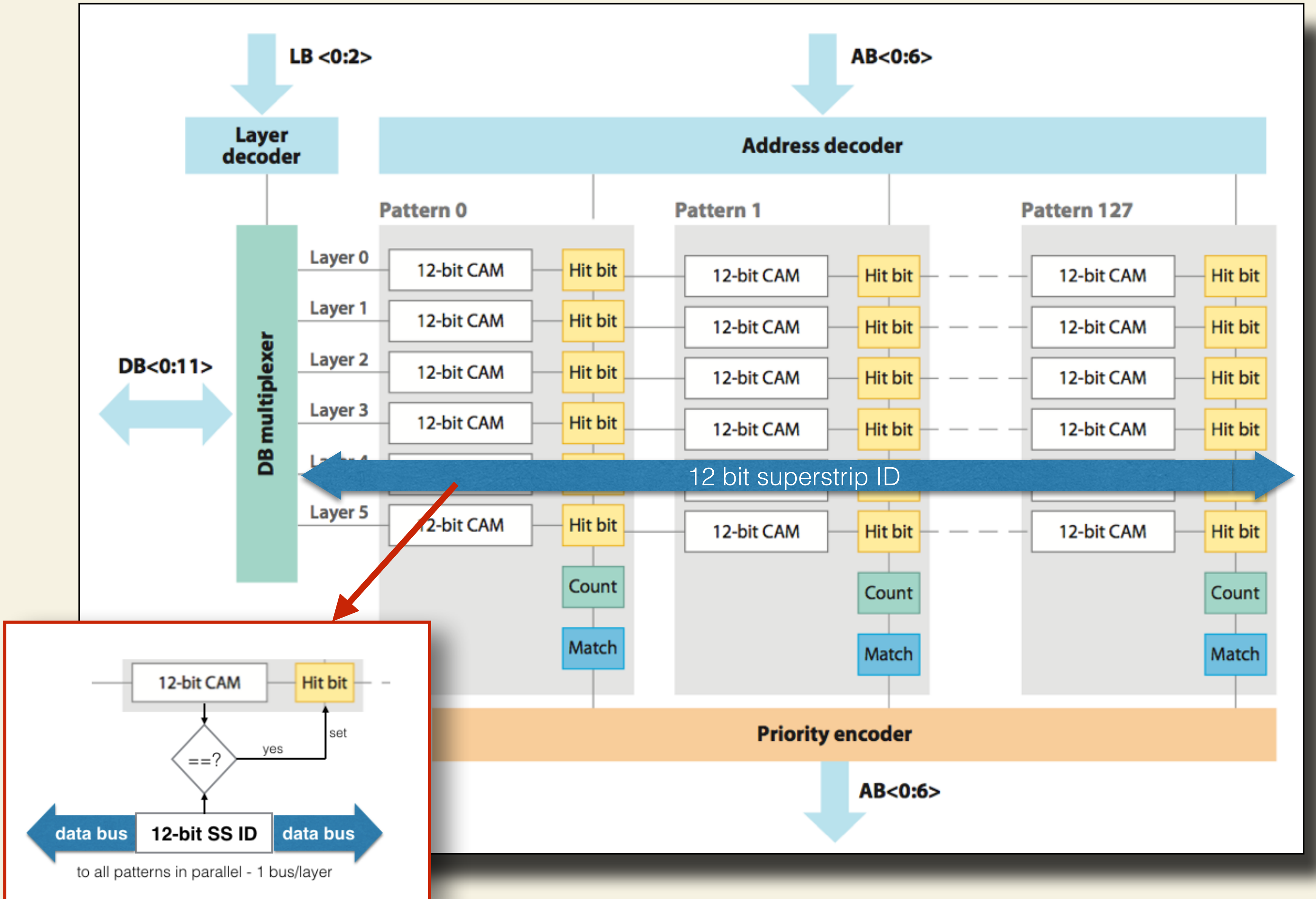
AM internal structure



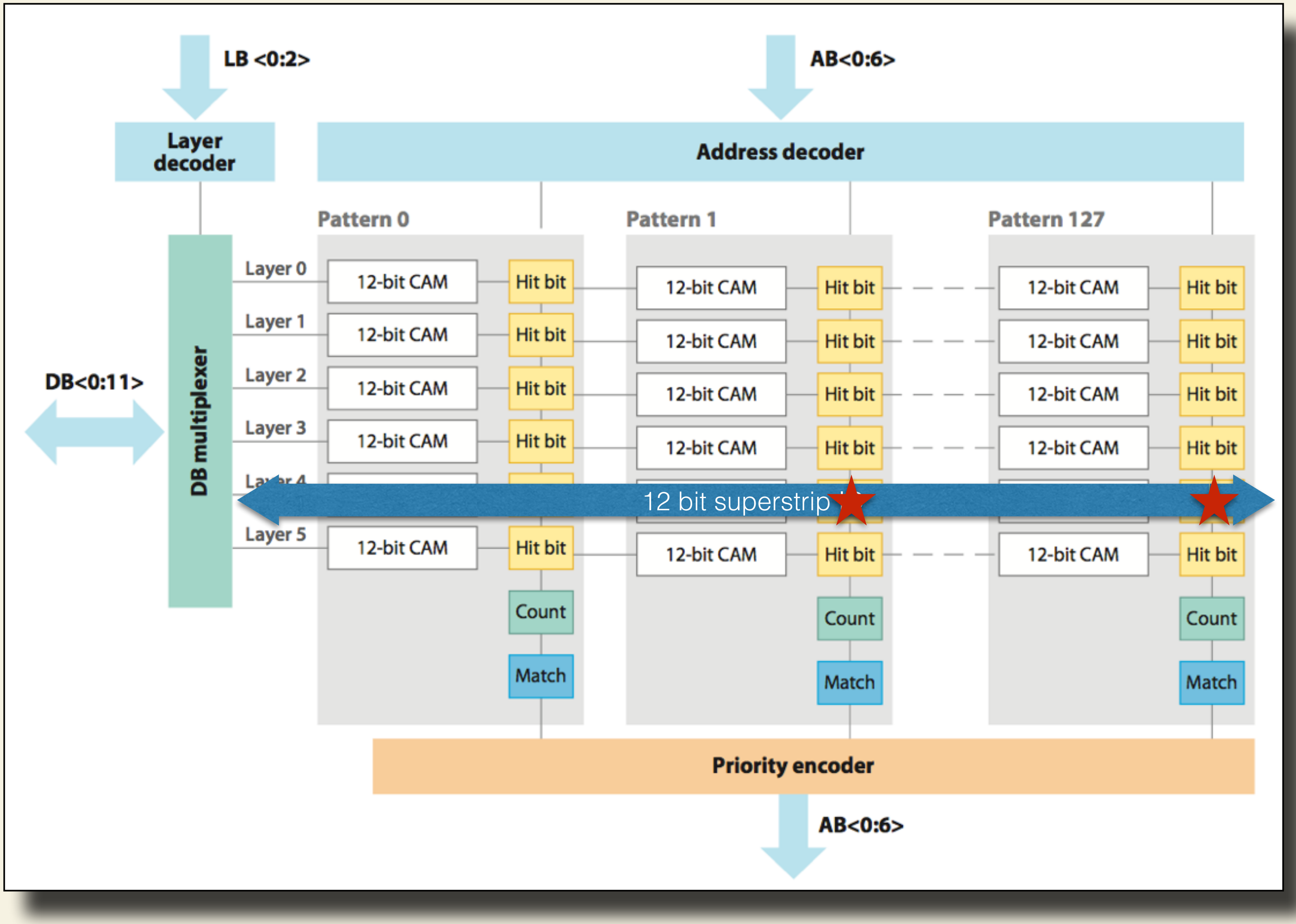
AM internal structure



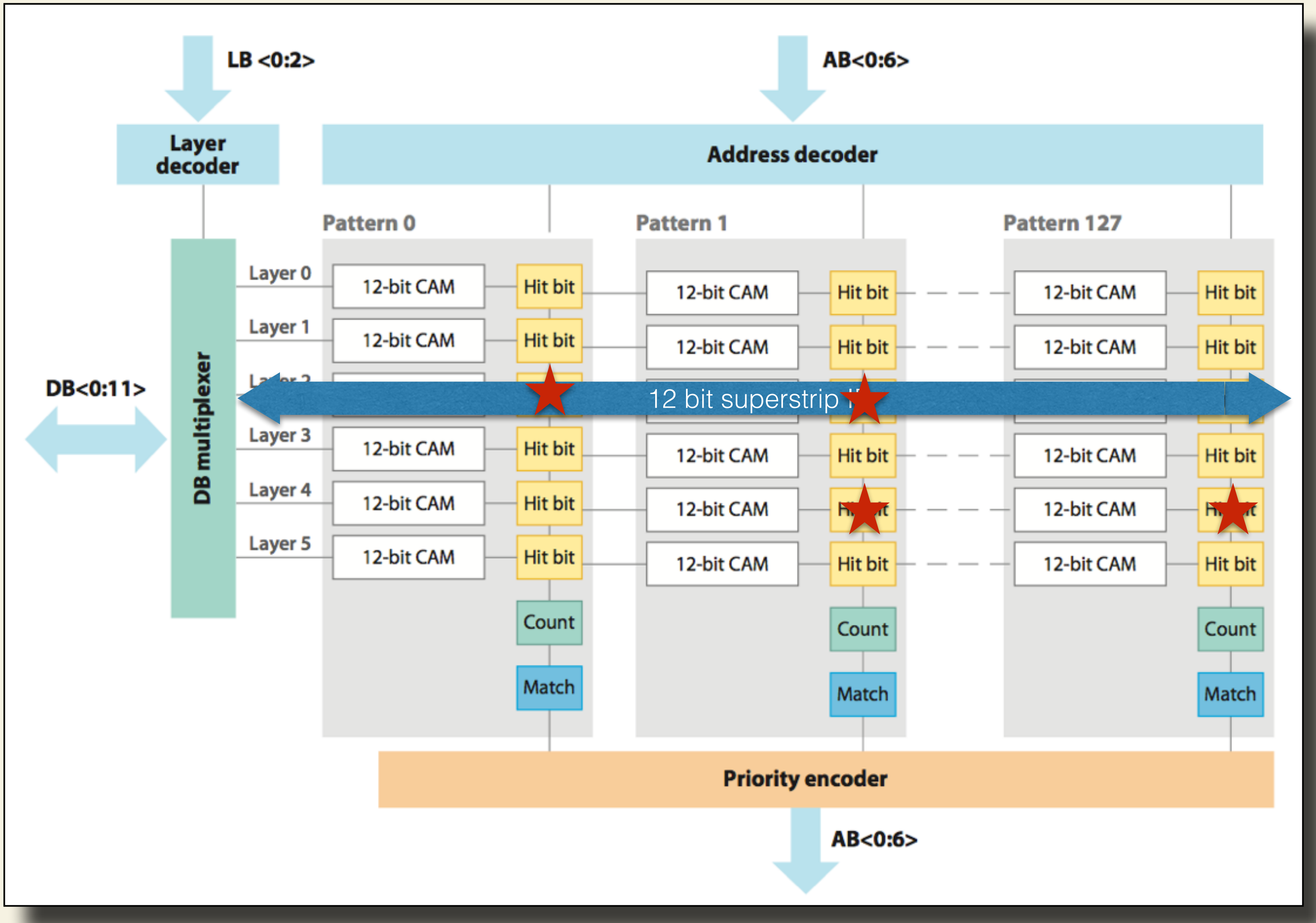
AM internal structure



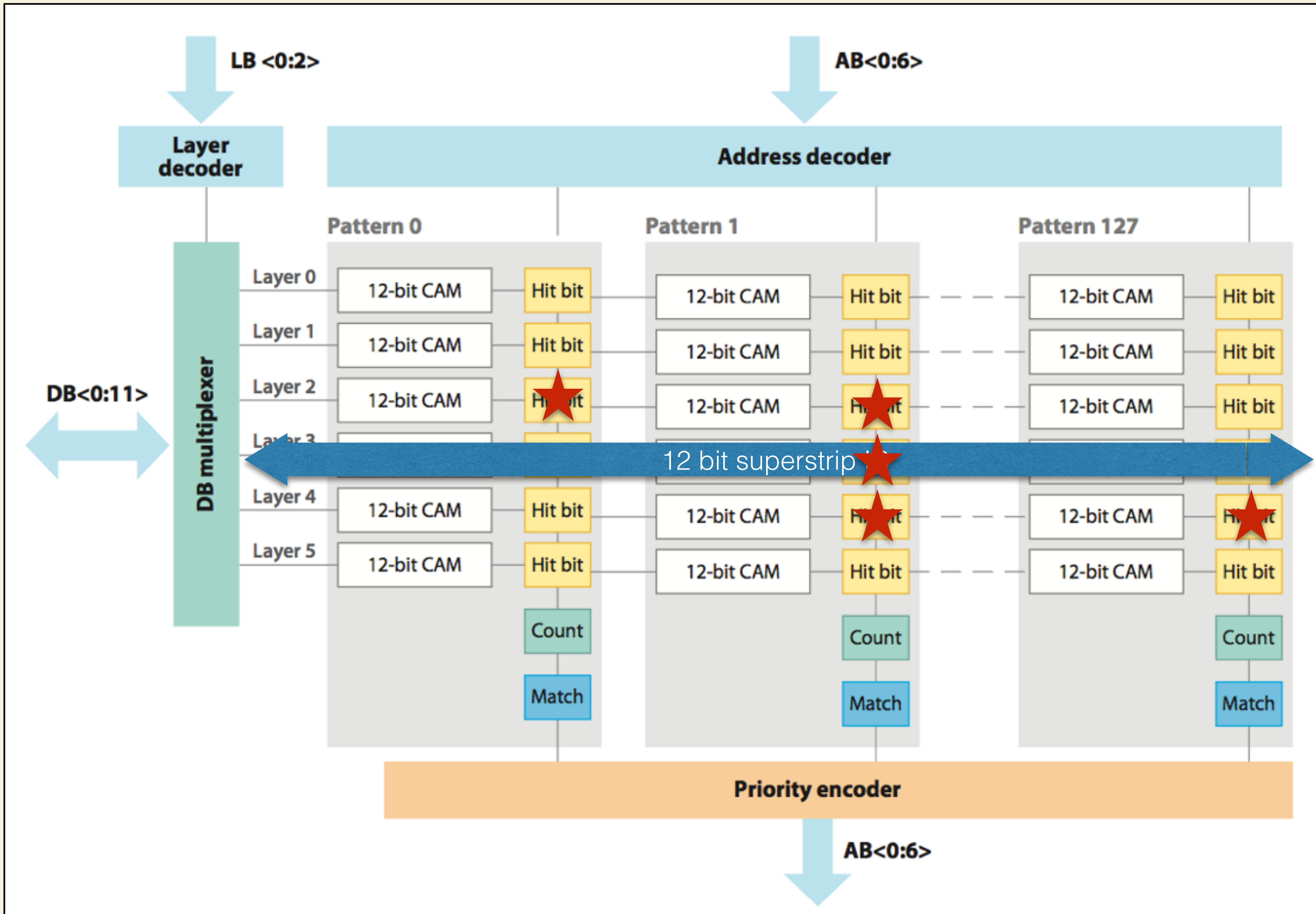
AM internal structure



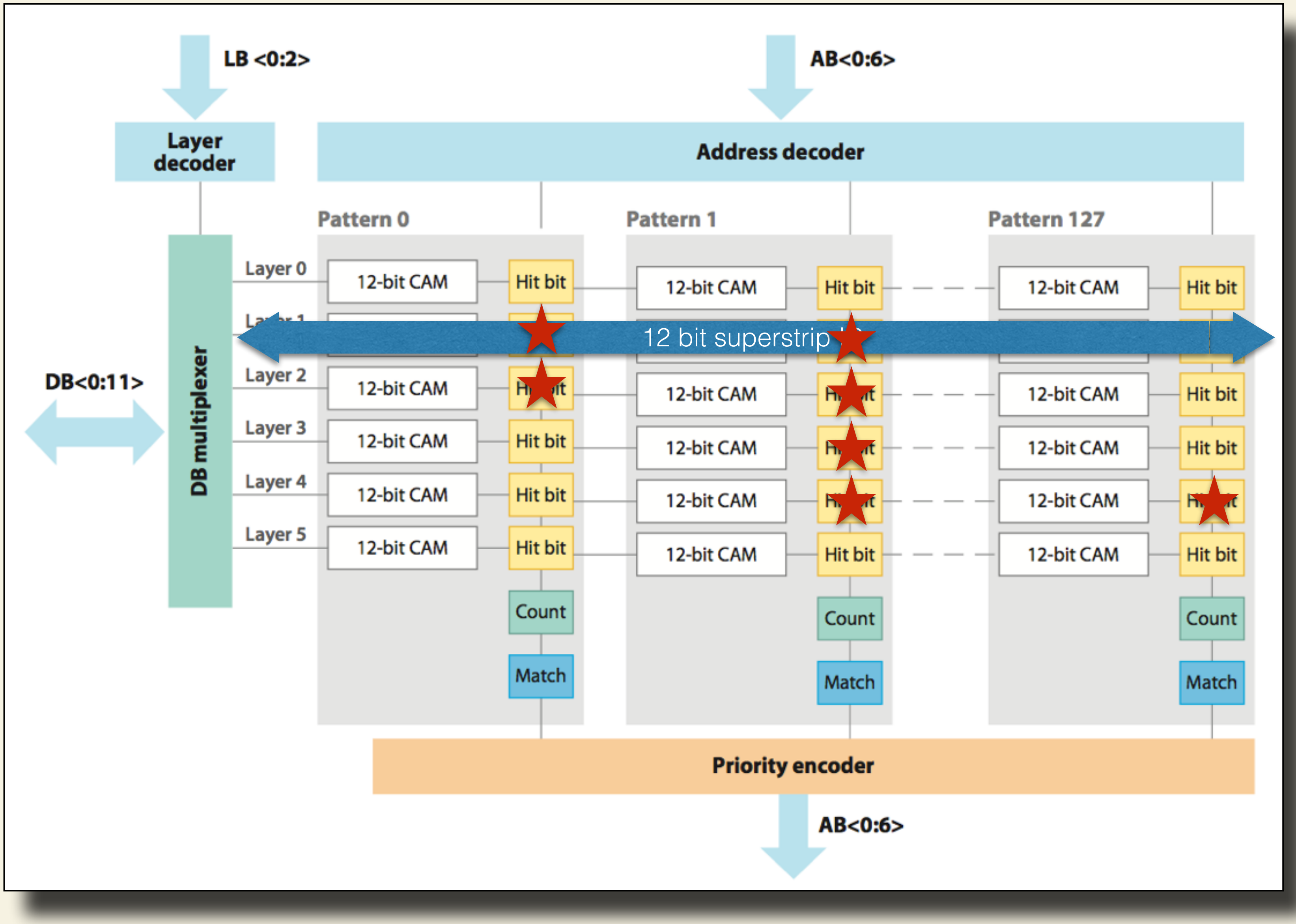
AM internal structure



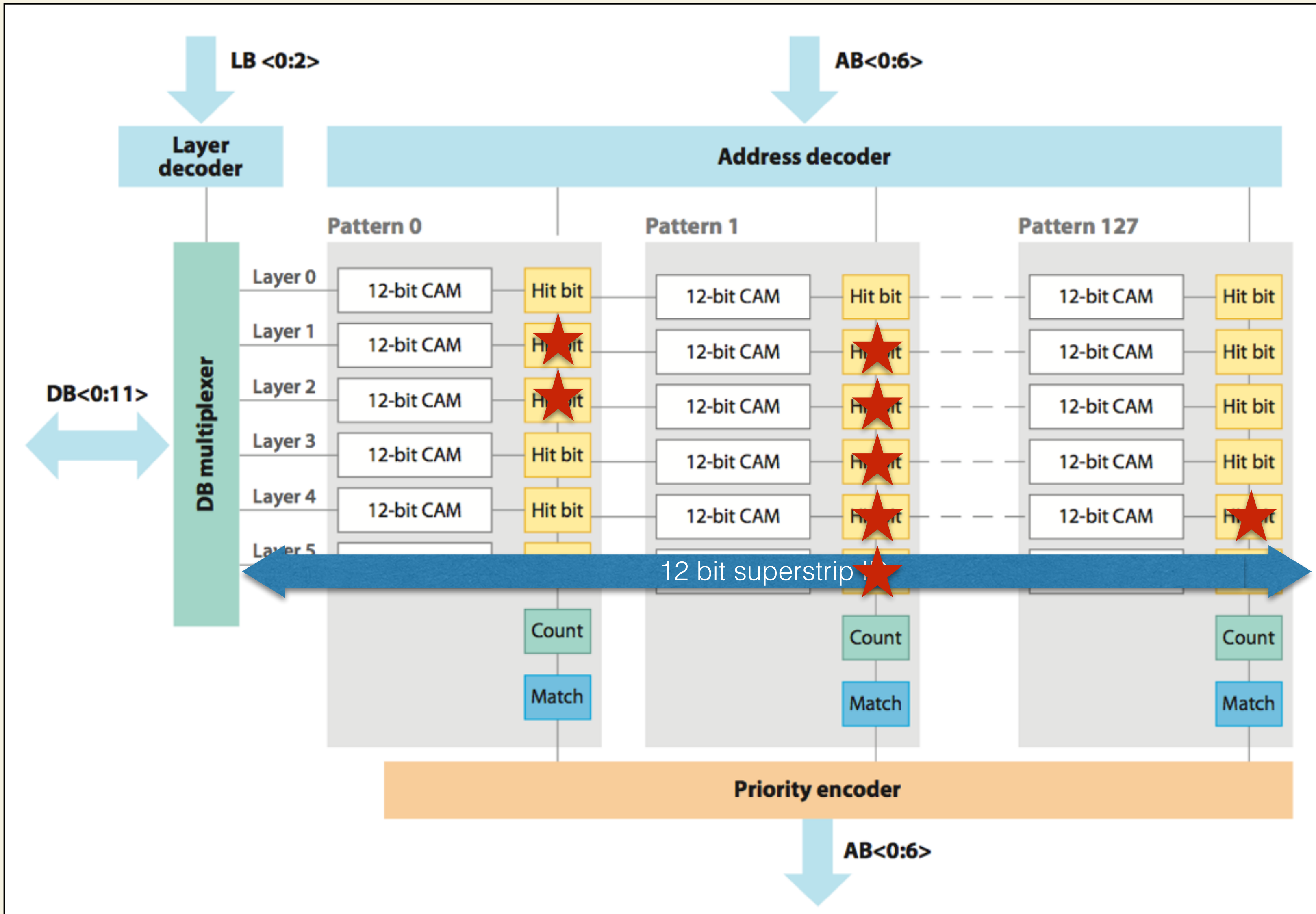
AM internal structure



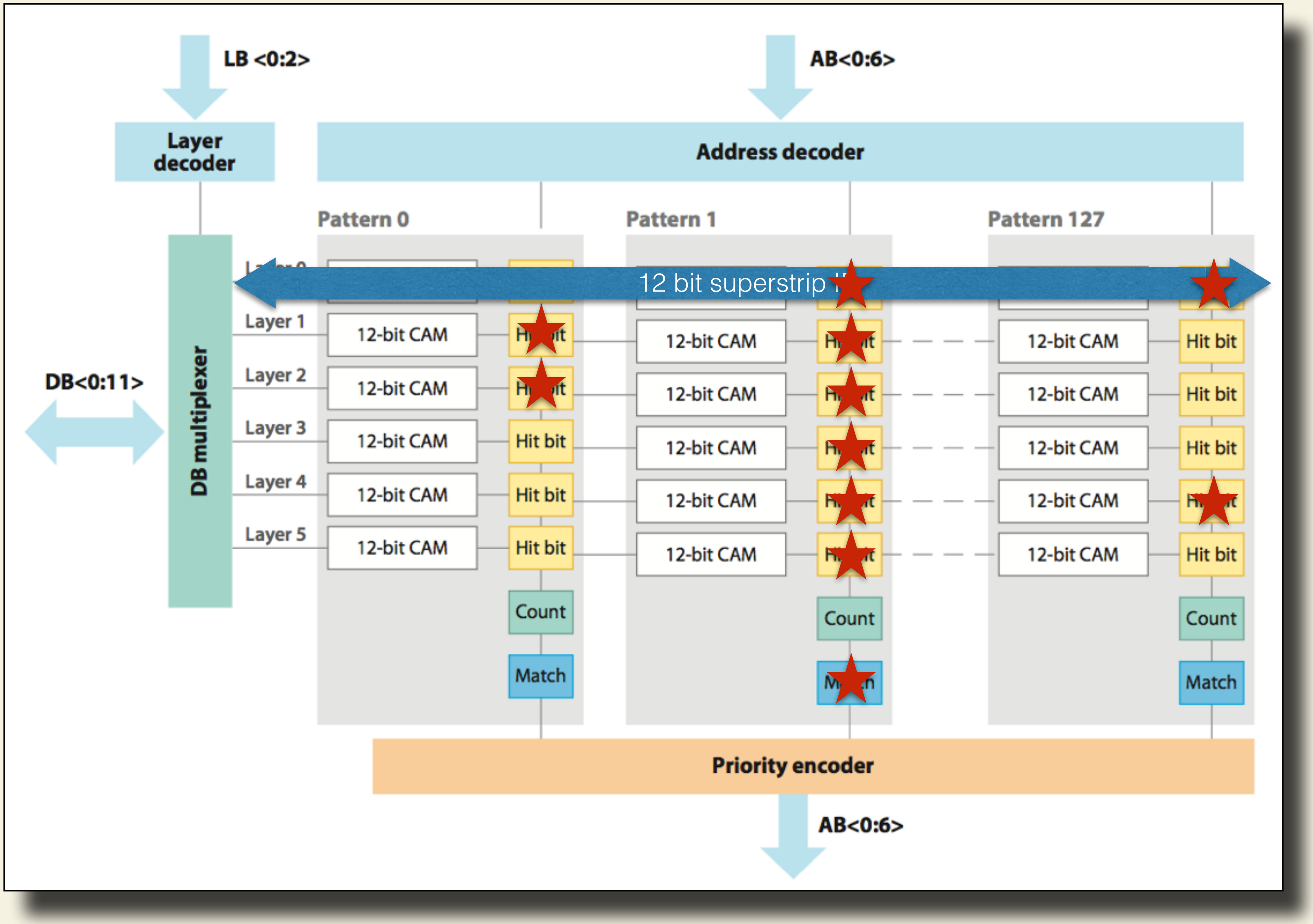
AM internal structure



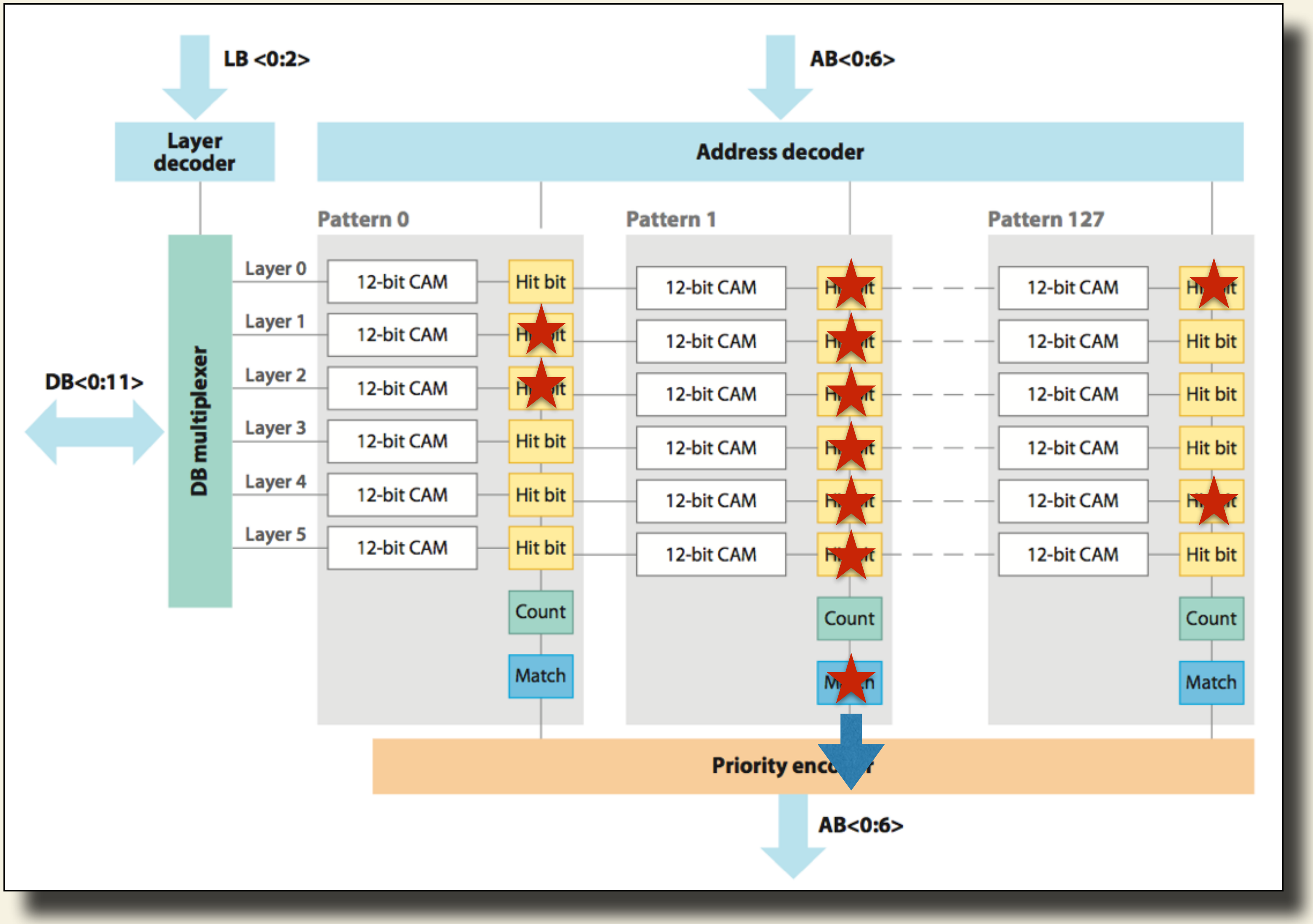
AM internal structure



AM internal structure



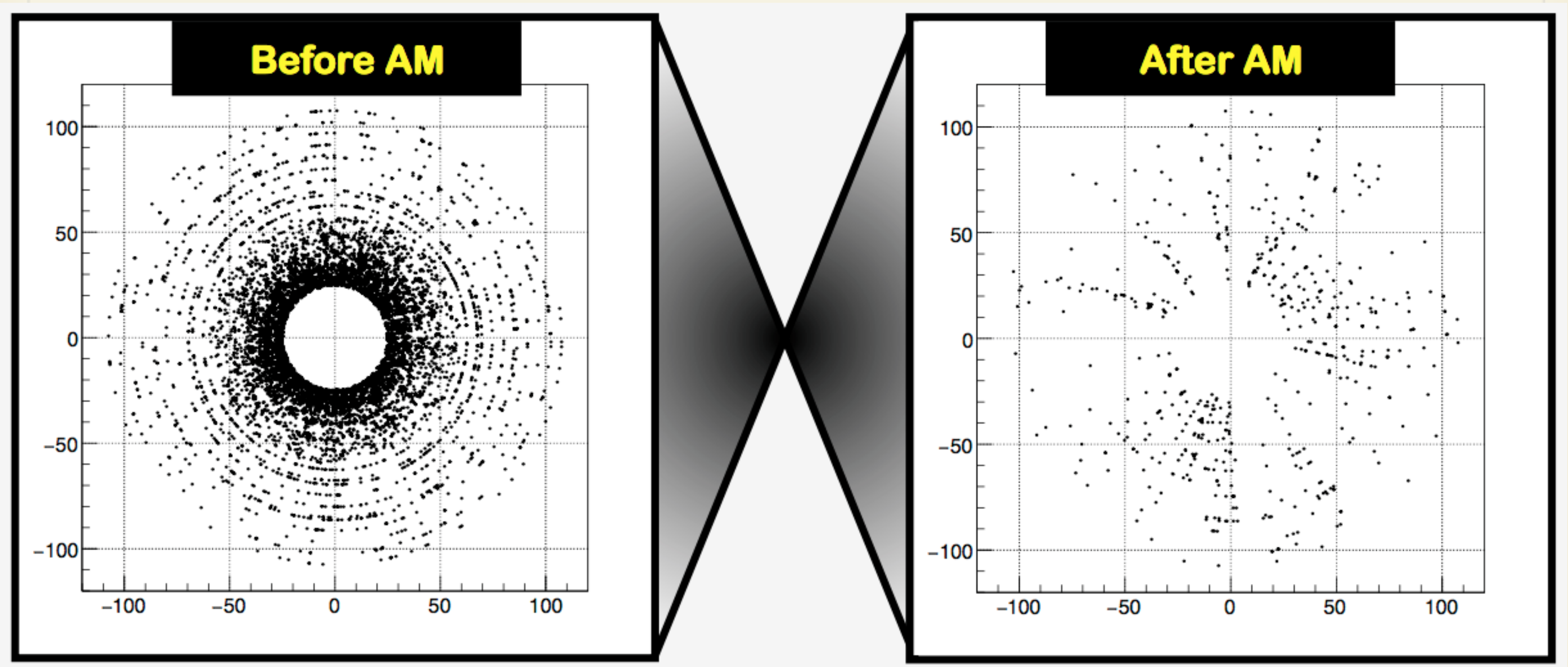
AM internal structure



as soon as all the hits have been read out from the detector into the AM, the first matched pattern is available immediately and all the others are coming out in a continuous stream, one per clock cycle

in this sense, pattern recognition, in the AM, takes very little time since it is mostly hidden behind the readout time

The Associative Memory does a lot in a very short time



Simulated beam crossing at the HL-LHC

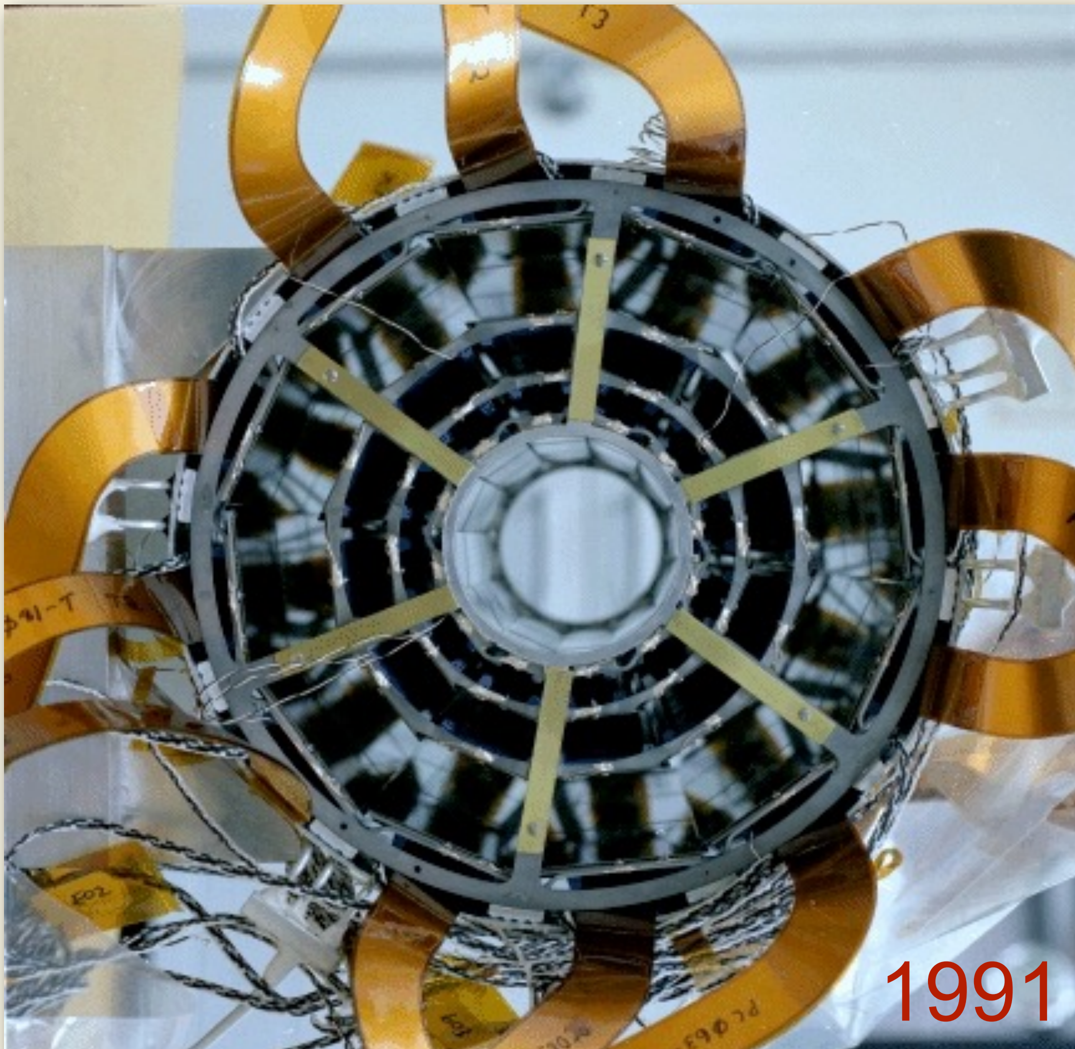
(drawing courtesy of Seb Viret)

but let’s go back again to the beginning…

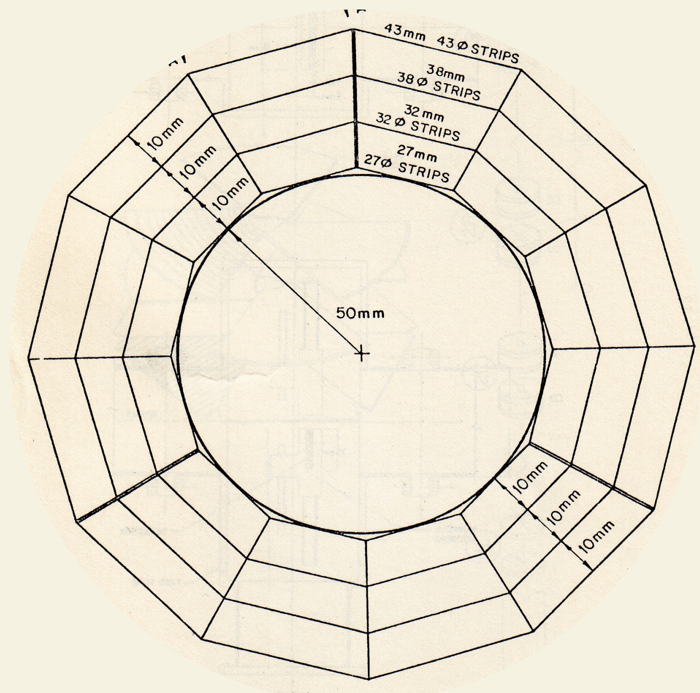
CDF and the SVX

SVX: the first CDF micro vertex detector

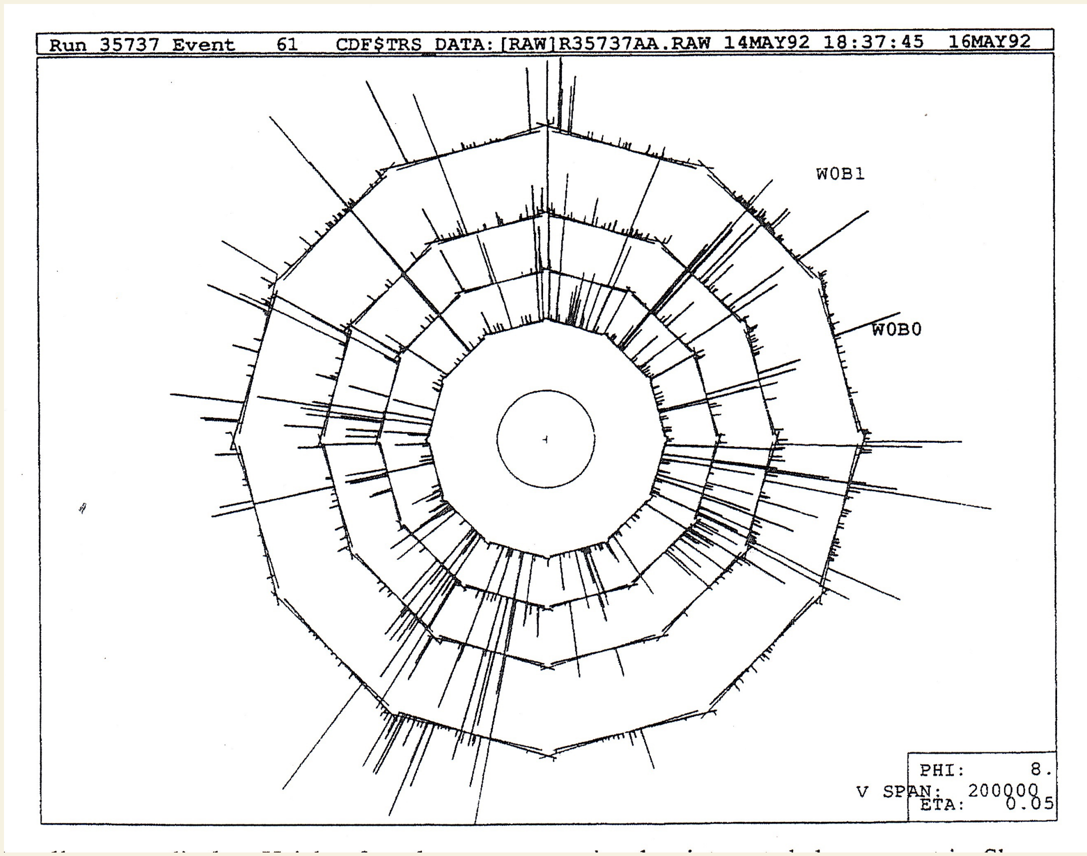
finished in 1991



1981



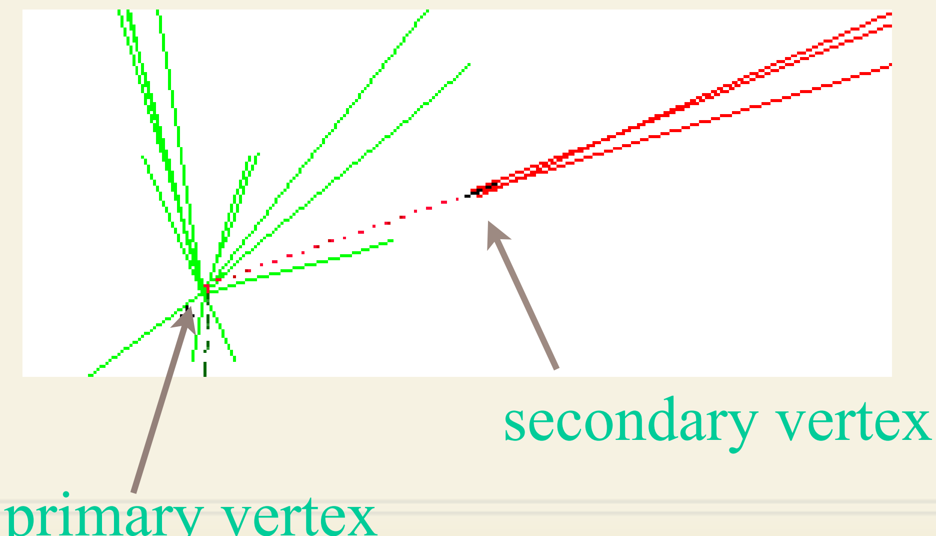
1992: particles from collisions are recorded by SVX



will be of crucial importance for the discovery of the Top quark in 1995

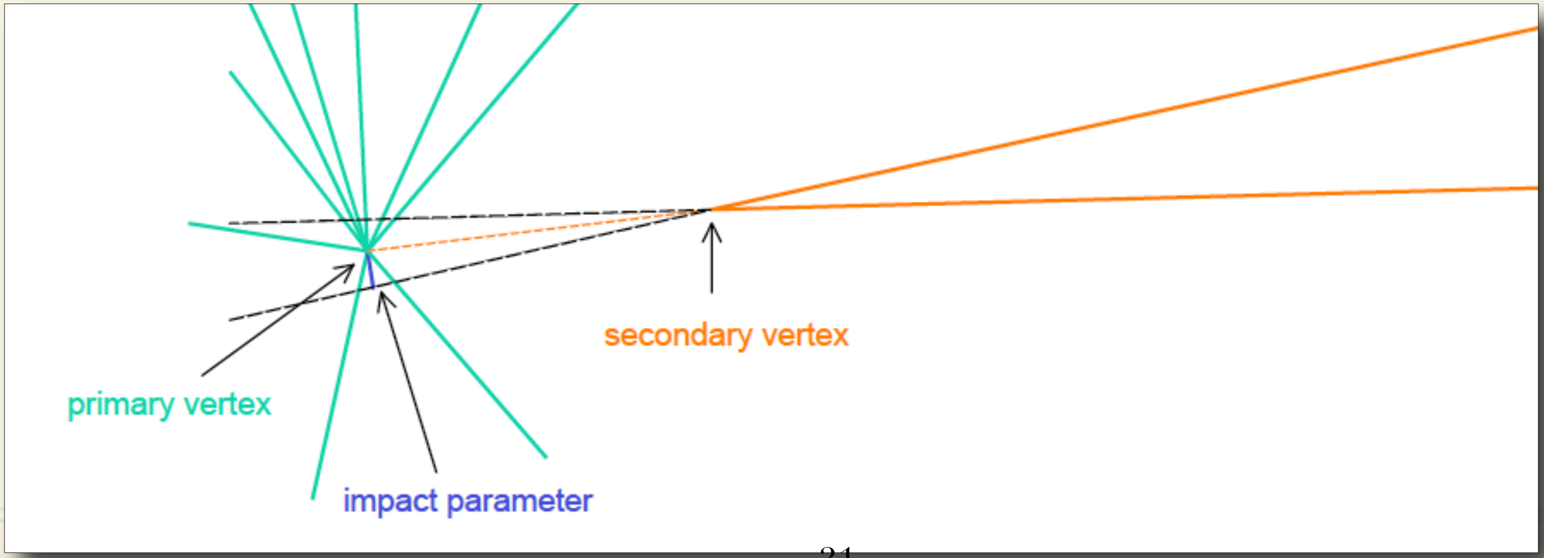
Heavy Flavor Physics

- at the start of CDF, at the end of the '80s, the first priority was the search for the *Top* quark but...
- quarks with *Charm* and *Beauty* are produced abundantly at the Tevatron and turn out to be extremely interesting too, but they are hard to identify being produced mostly at relatively low transverse momentum
- their main characteristics is a relatively long lifetime which creates *secondary vertices* at distances of the order of millimeters from the collision point



to trigger on secondary vertices you need to:

- perform pattern recognition and sort hits into tracks
- fit all tracks to extract relevant parameters (P_T, phi, d)
- do all this with
  - sufficient speed to be used at trigger level (~20µs)
  - sufficient impact parameter precision (~40µm)



May 1st 1991

CDF/DOC/TRIGGER/PUBLIC/1421

SVT
THE SILICON VERTEX TRACKER

Luciano Ristori

May 1, 1991

INTRODUCTION

This note describes the architecture of a device we believe we can build to reconstruct tracks in the Silicon Vertex Detector (SVX) with enough speed and accuracy to be used at trigger level 2 to select events containing secondary vertices originated by B decay. We name such a device *Silicon Vertex Tracker* (SVT).

The use of SVT as part of the CDF trigger would allow us to collect a large sample of B's (> 10^7 events) in a 100 pb^{-1} run.

B production at 2 TeV in the c.m. is abundant: Isajet predicts that, in the central region, 6.5% of two-jet events with Pt>20 GeV/c contain a B pair. Thus we need a trigger with a relatively modest rejection factor (10 ÷ 20) not necessarily requiring the presence of very high P_T tracks.

It turns out that the simple requirement of a single track with an impact parameter greater than a given threshold might do the job.

The possibility to use the output of SVT to actually reconstruct secondary vertices is left open and it's not discussed here.

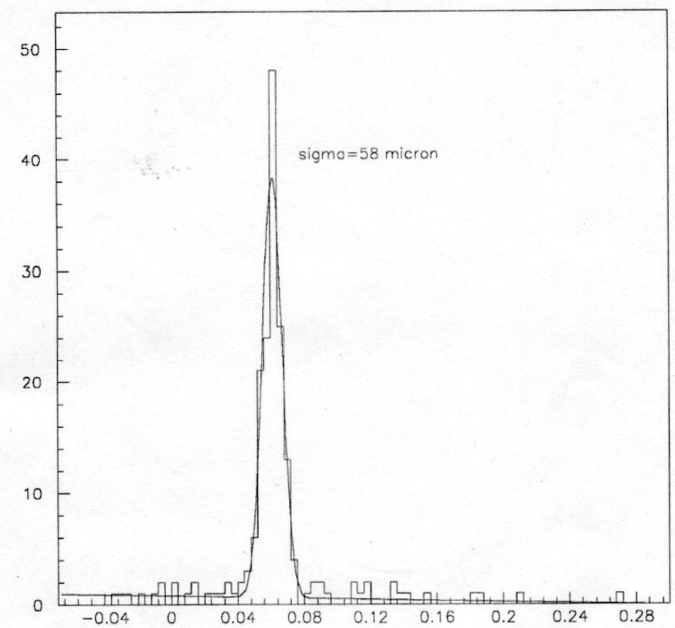
In Section 1 we report the results of some simple simulations we have done to show the efficacy of the impact parameter cut, in Section 2 we overview the overall architecture of SVT, in Section 3 we describe the different parts SVT is made of and how they relate to the different stages the track finding process goes through.

1. SIMULATION RESULTS

1.1 Impact Parameter Cut

The impact parameter s of each track is defined as the minimum

real data + SVT simulation



Transverse profile of the Tevatron luminous region obtained feeding real data from SVX into the simulation of SVT. The sigma of this curve (58 μm) is the result of the folding of spot size with impact parameter resolution.

Nov 17, 1992

May 1st 1991

CDF/DOC/TRIGGER/PUBLIC/1421

SVT
THE SILICON VERTEX TRACKER

Luciano Ristori

May 1, 1991

INTRODUCTION

This note describes the architecture of a device we believe we can build to reconstruct tracks in the Silicon Vertex Detector (SVX) with enough speed and accuracy to be used at trigger level 2 to select events containing secondary vertices originated by B decay. We name such a device *Silicon Vertex Tracker* (SVT).

the central region, 6.5% of two-jet events with Pt>20 GeV/c contain a B pair. Thus we need a trigger with a relatively modest rejection factor (10 ÷ 20) not necessarily requiring the presence of very high P_T tracks.

It turns out that the simple requirement of a single track with an impact parameter greater than a given threshold might do the job.

The possibility to use the output of SVT to actually reconstruct secondary vertices is left open and it's not discussed here.

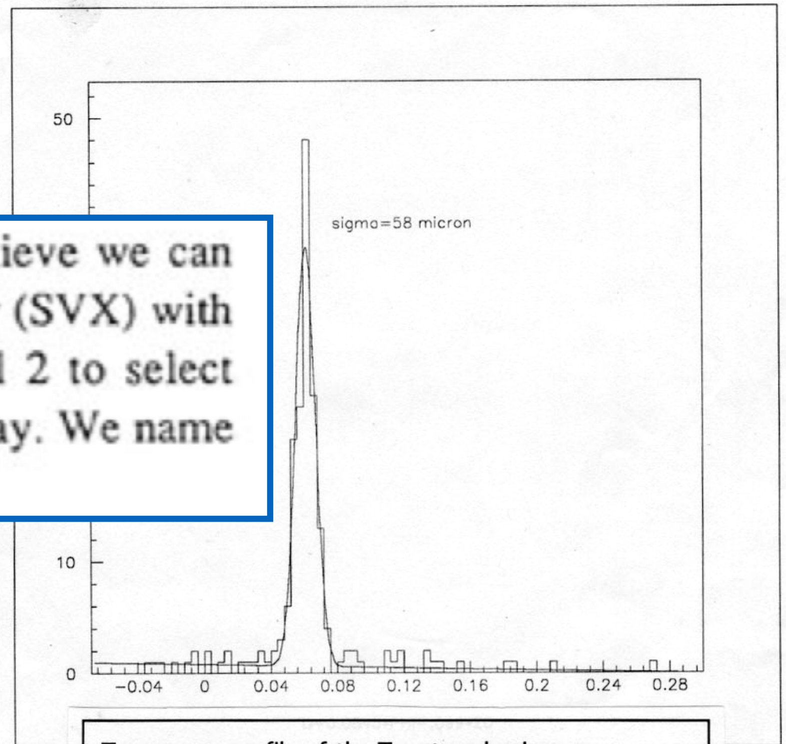
In Section 1 we report the results of some simple simulations we have done to show the efficacy of the impact parameter cut, in Section 2 we overview the overall architecture of SVT, in Section 3 we describe the different parts SVT is made of and how they relate to the different stages the track finding process goes through.

1. SIMULATION RESULTS

1.1 Impact Parameter Cut

The impact parameter s of each track is defined as the minimum

real data + SVT simulation



Transverse profile of the Tevatron luminous region obtained feeding real data from SVX into the simulation of SVT. The sigma of this curve (58 μm) is the result of the folding of spot size with impact parameter resolution.

Nov 17, 1992

May 1st 1991

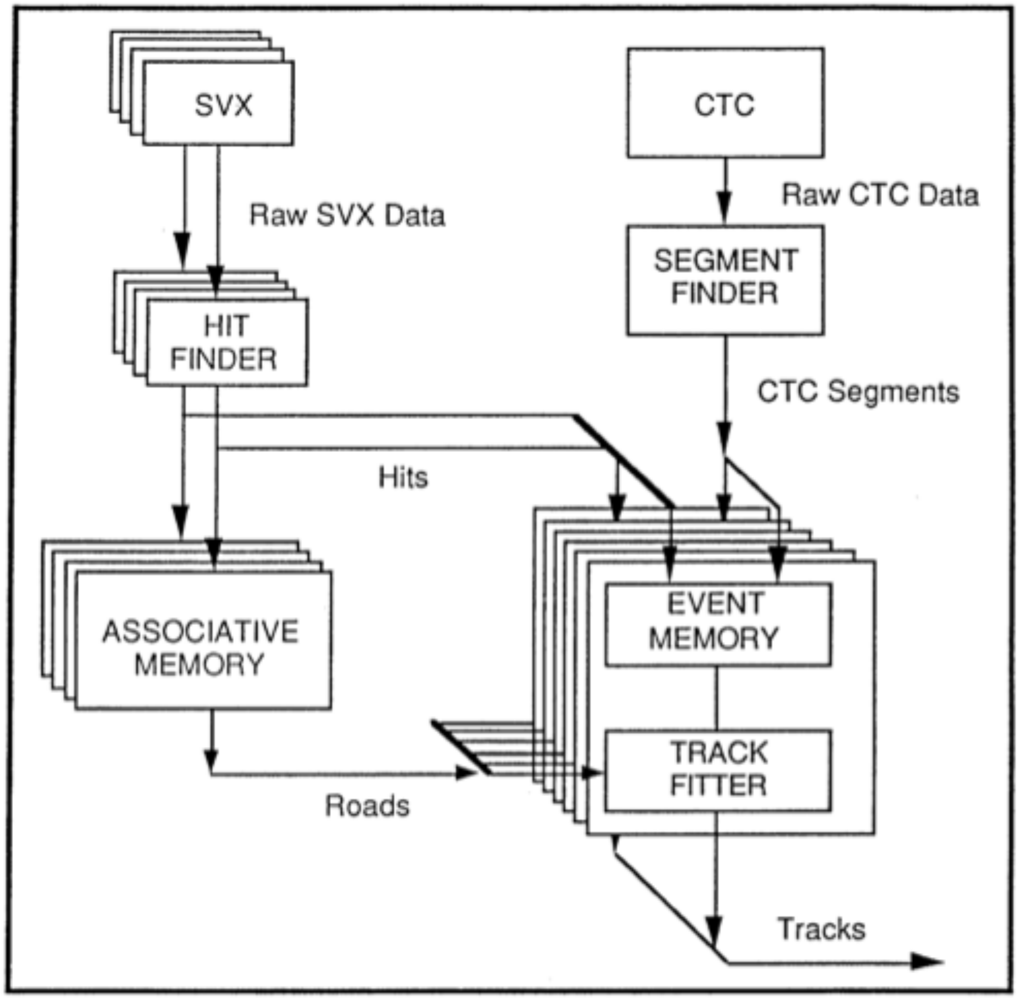


Fig. 5: SVT Architecture

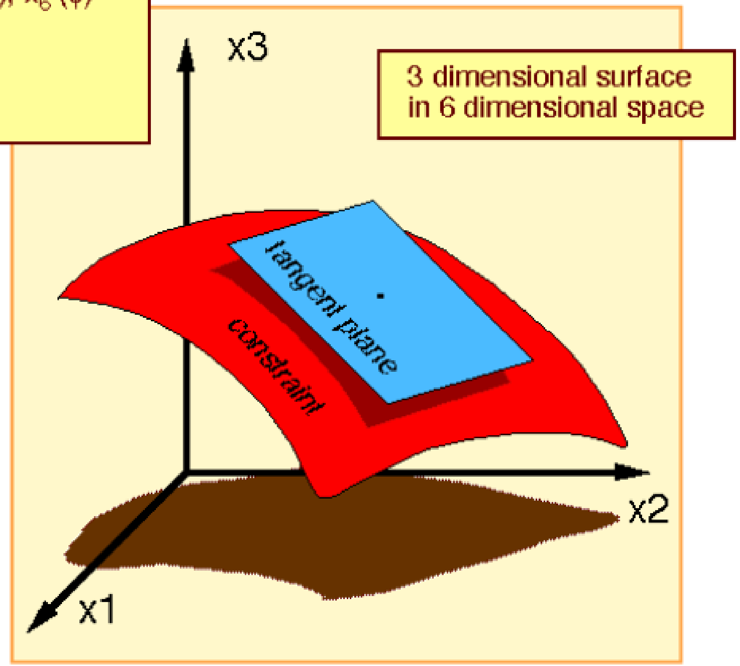
Linearized Fit

SVT

6 coordinates: x_1, x_2, x_3, x_4, x_5 (P_T), x_6 (ϕ)

3 parameters to fit: P_T , ϕ , d

3 constraints



tangent plane:

$$\sum_{1}^{6} a_i x_i = b$$

track parameters:

$$d \approx c_0 + \sum_{1}^{6} c_i x_i$$

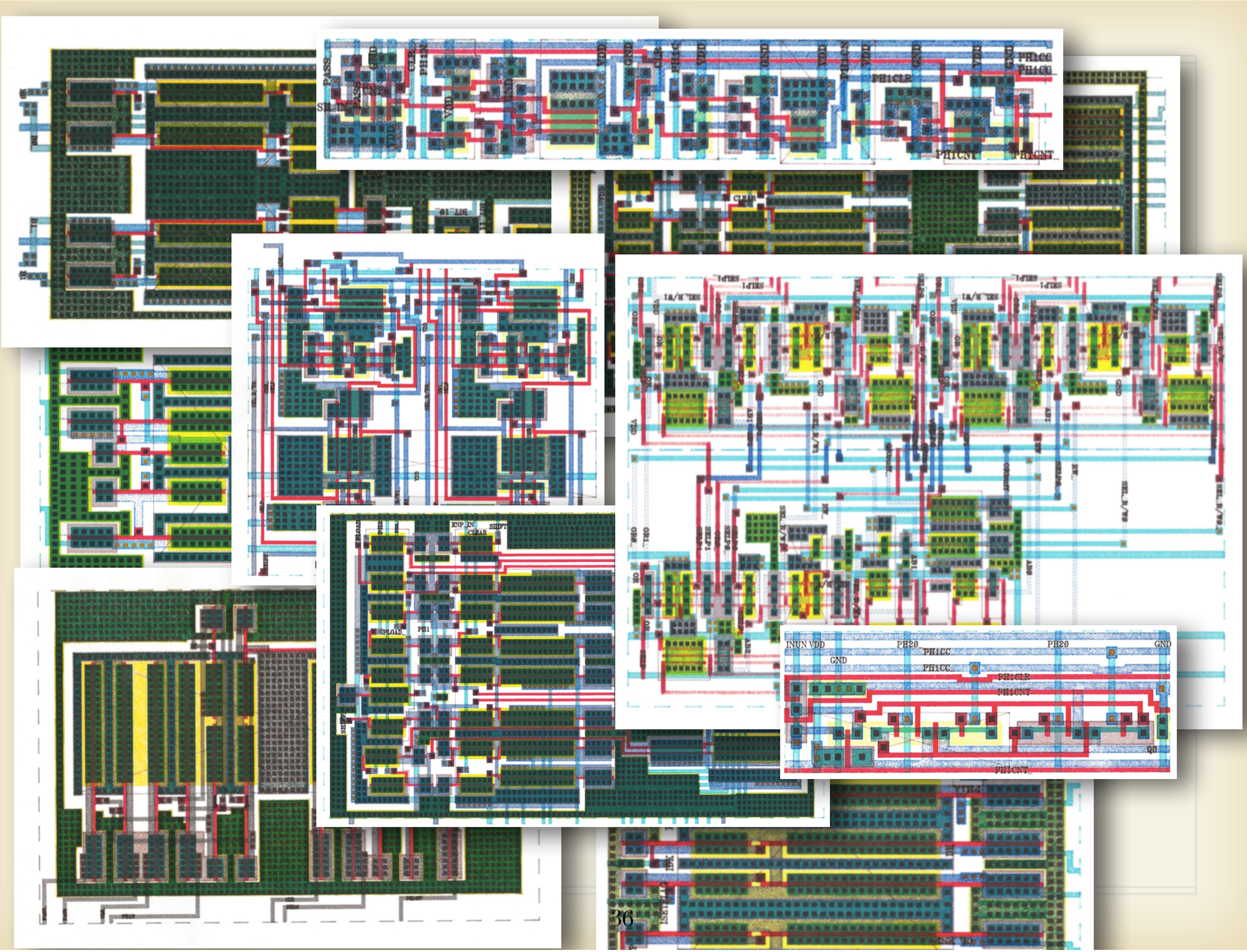
Linear approximation is so good that a single set of constants is sufficient for a whole detector wedge (30° in ϕ)

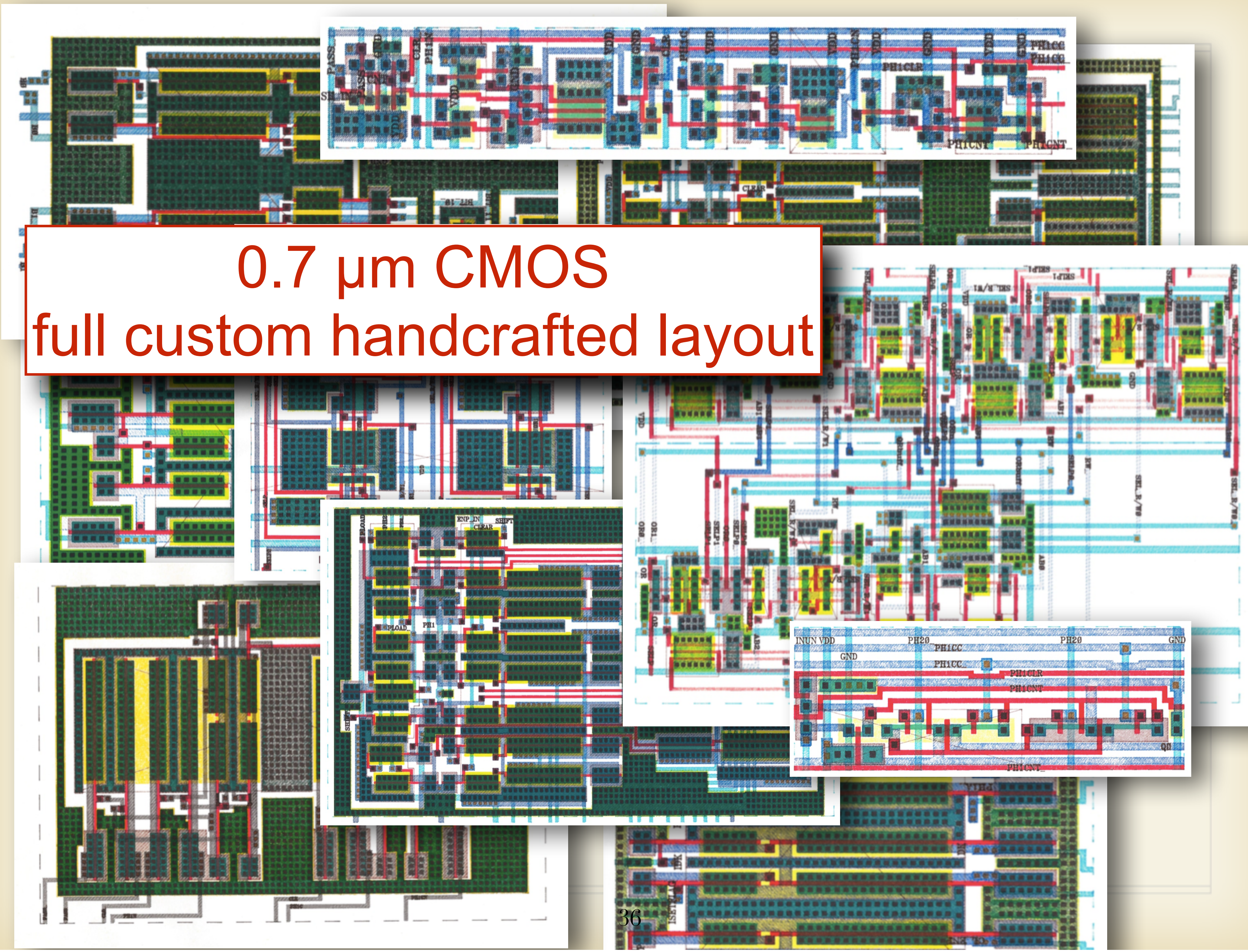
linear transformation

hit coordinates → track parameters

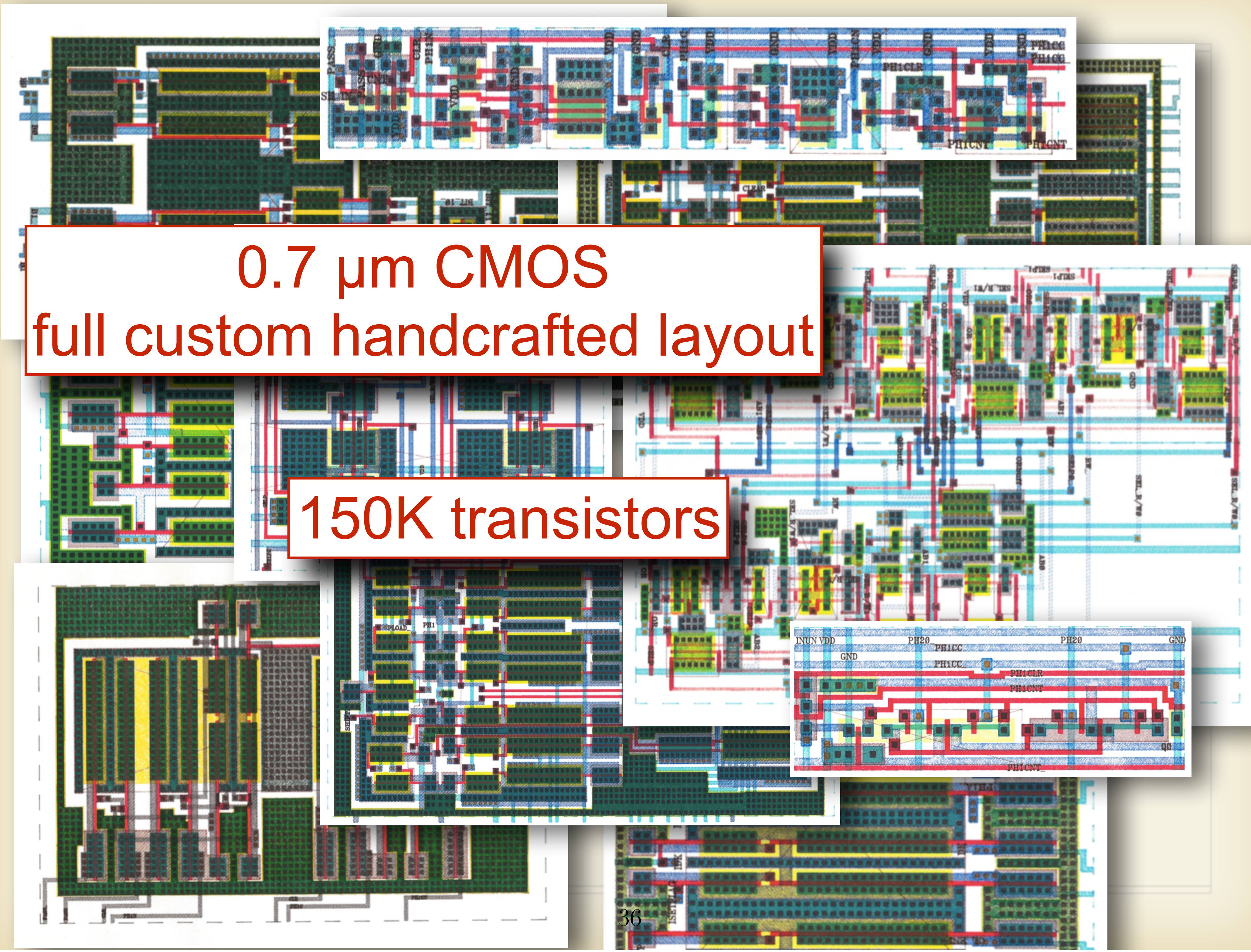
single AM pattern = small volume in phase space

The AM chip


PH1CC
PH1CLR
PH1CNT
INUN VDD
GND
PH2Ø
CLEAR
PLOAD
PH1
ENP_IN
SHIFT

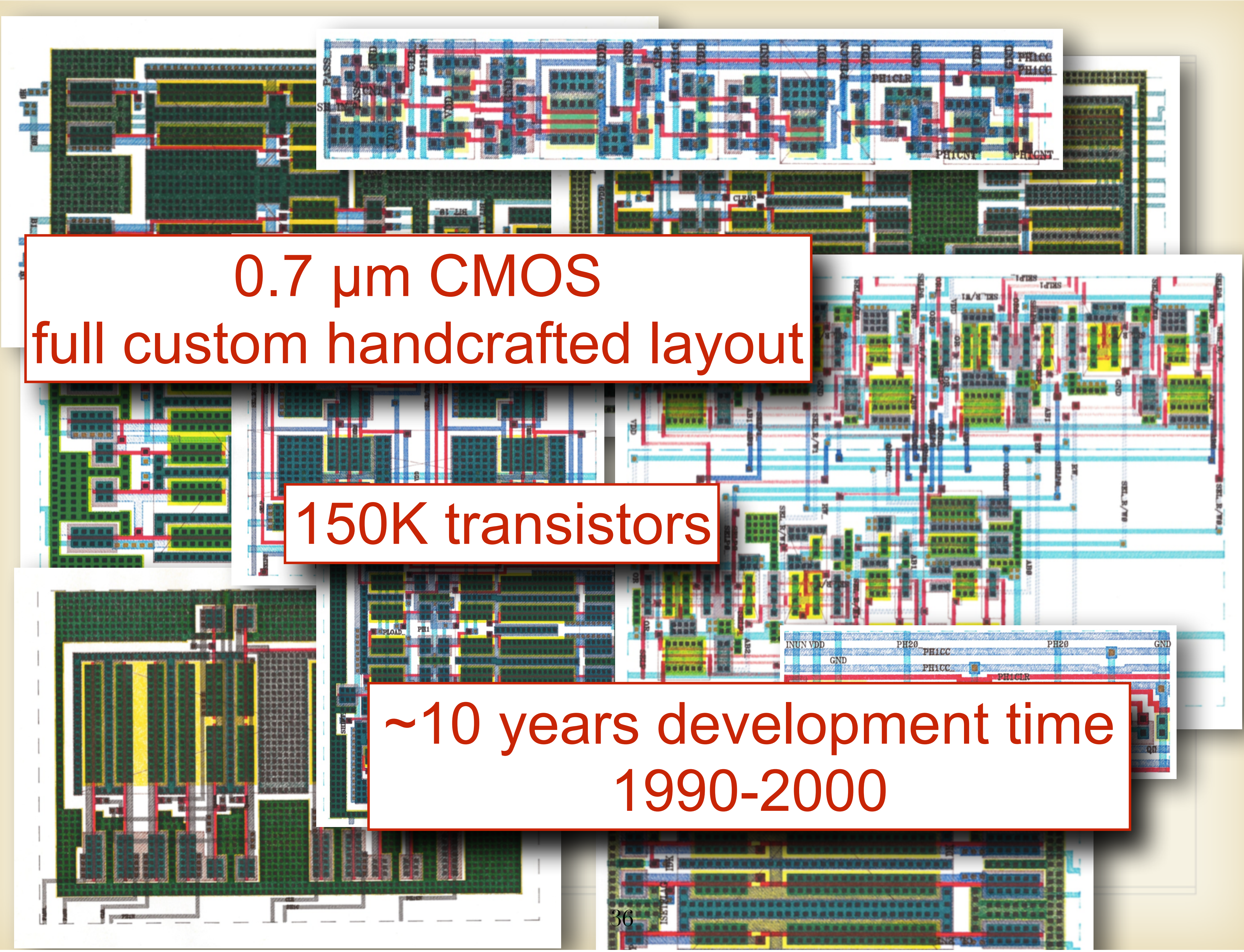


0.7 µm CMOS full custom handcrafted layout



0.7 µm CMOS
full custom handcrafted layout

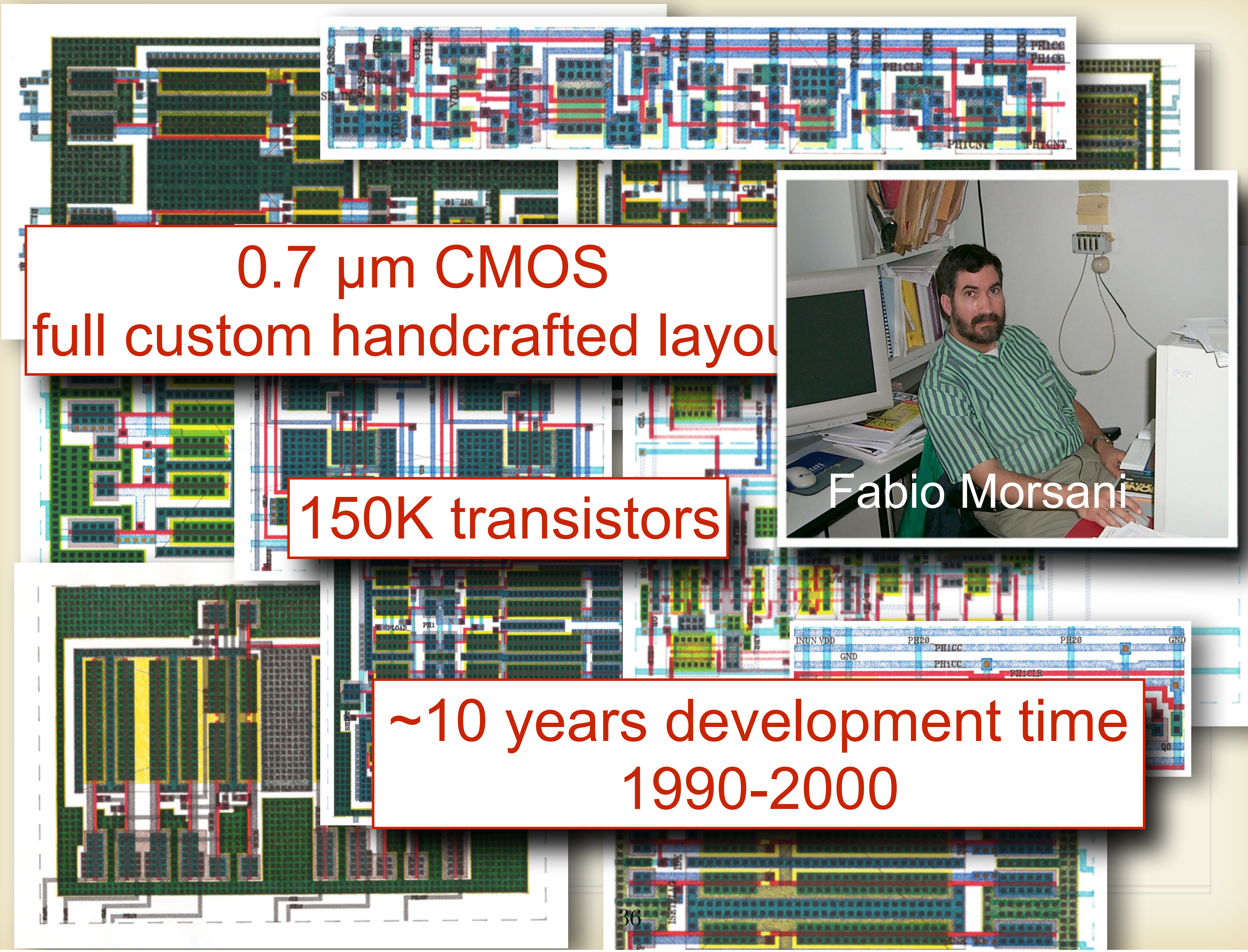
150K transistors



0.7 µm CMOS
full custom handcrafted layout

150K transistors

~10 years development time
1990-2000

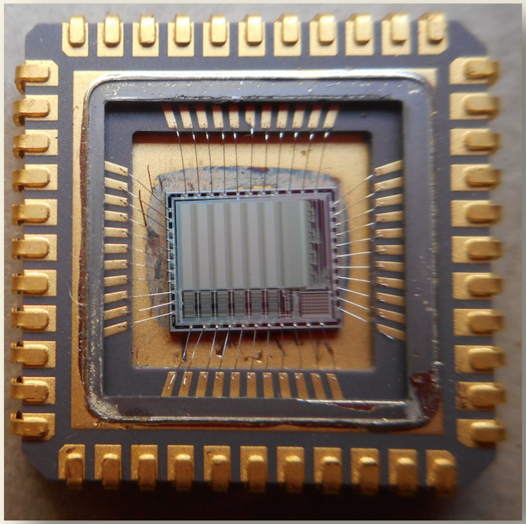


0.7 µm CMOS
full custom handcrafted layout

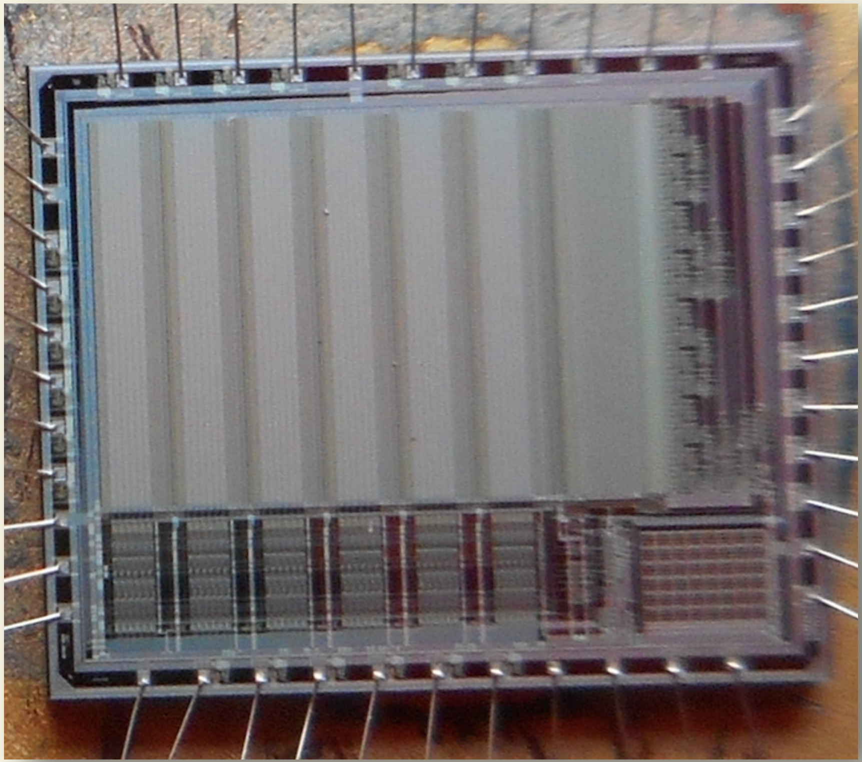
150K transistors

Fabio Morsani

~10 years development time
1990-2000



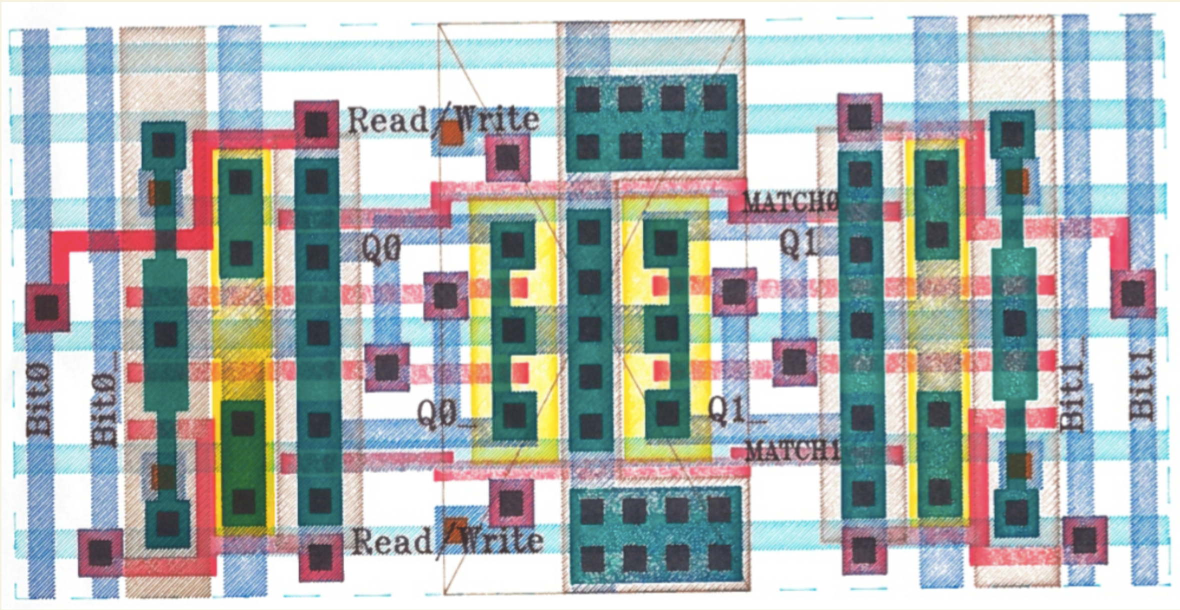
First production prototype end of 1998

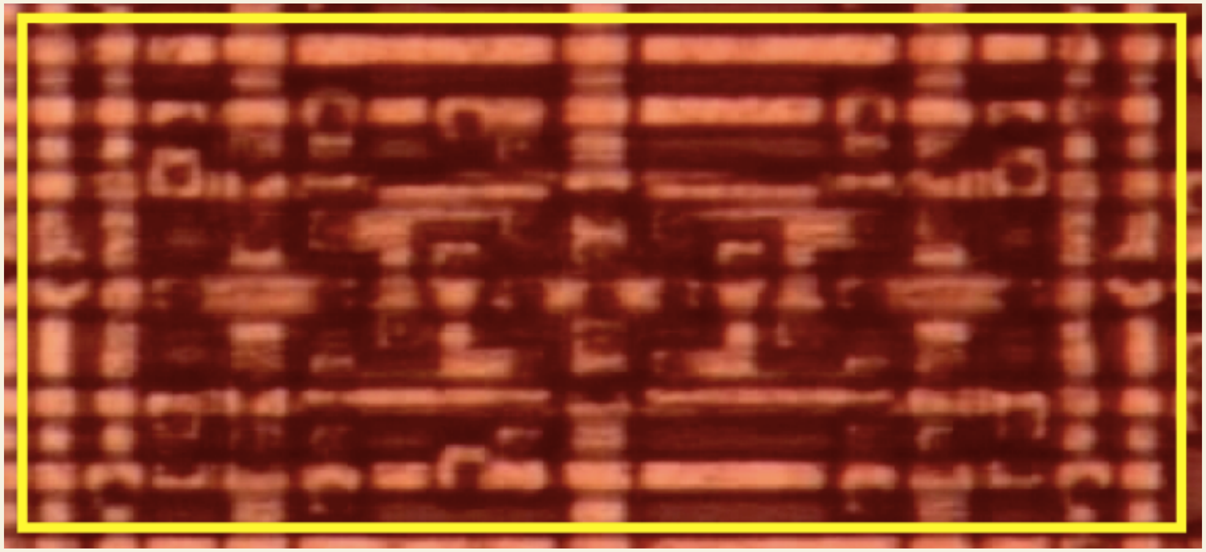


35 mm^2

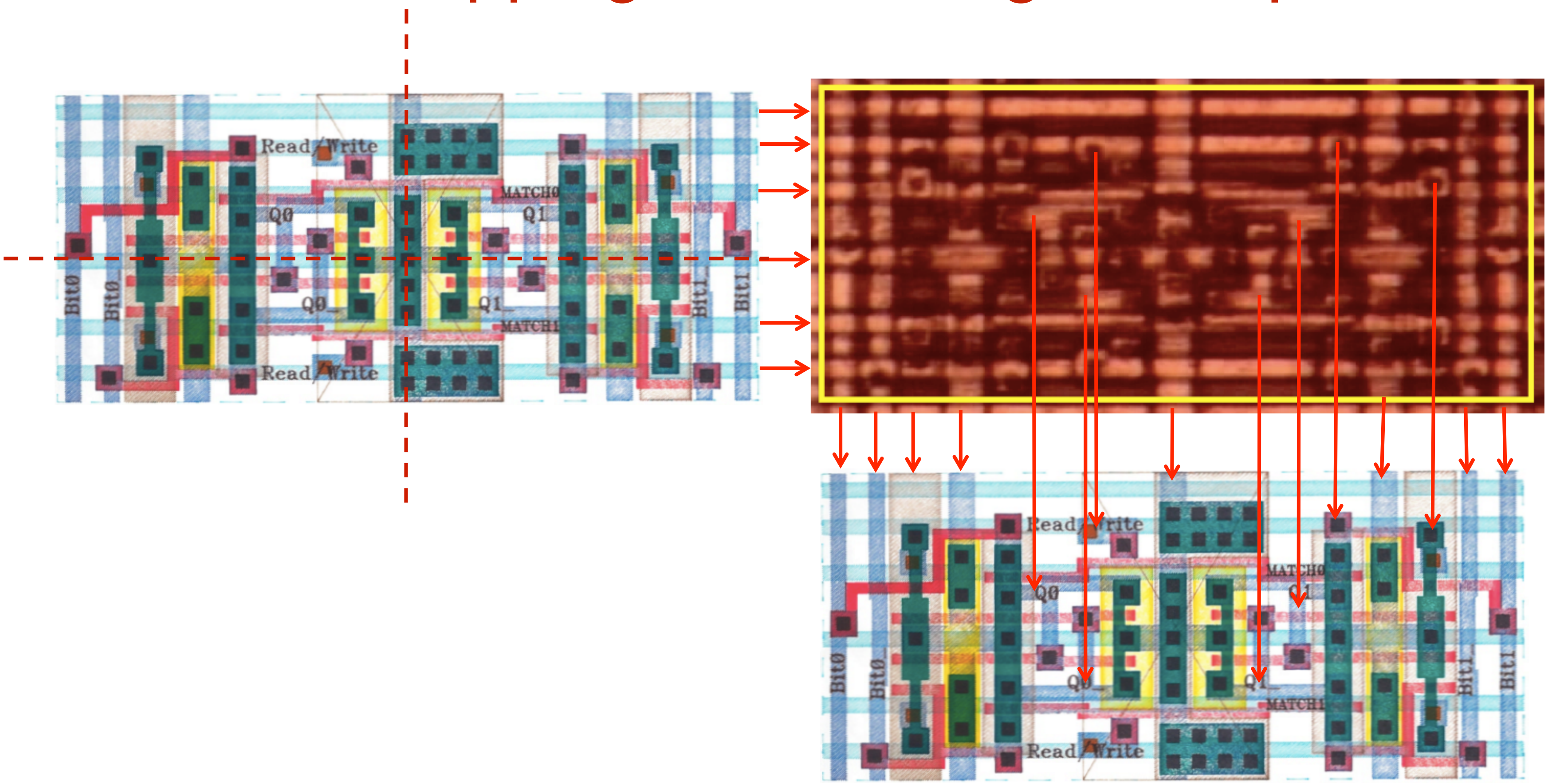
150K transistors

AM cell: pen plotter drawing and microscope photo





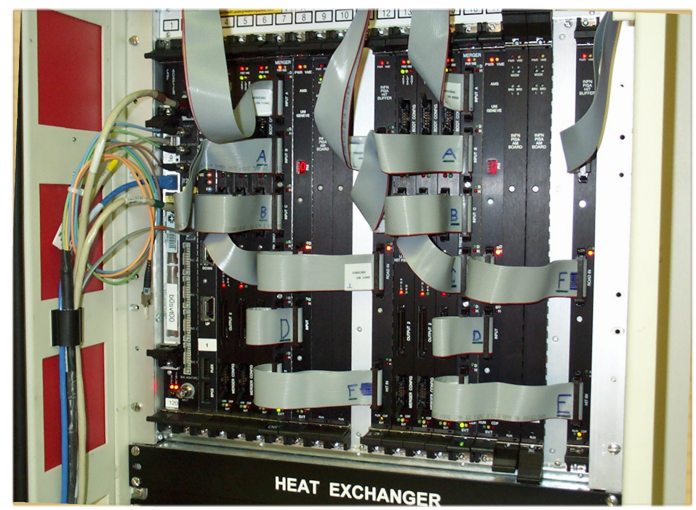
AM cell: mapping the drawing to the photo




Luciano
Giovanni
Alexei
Fabio
AM board
AM chips
May 2000

August 2000

All SVT boards installed in the CDF trigger room



Moore's Law

2000: CDF AMchip = 128 patterns/35 mm^2 = 3.7/mm^2

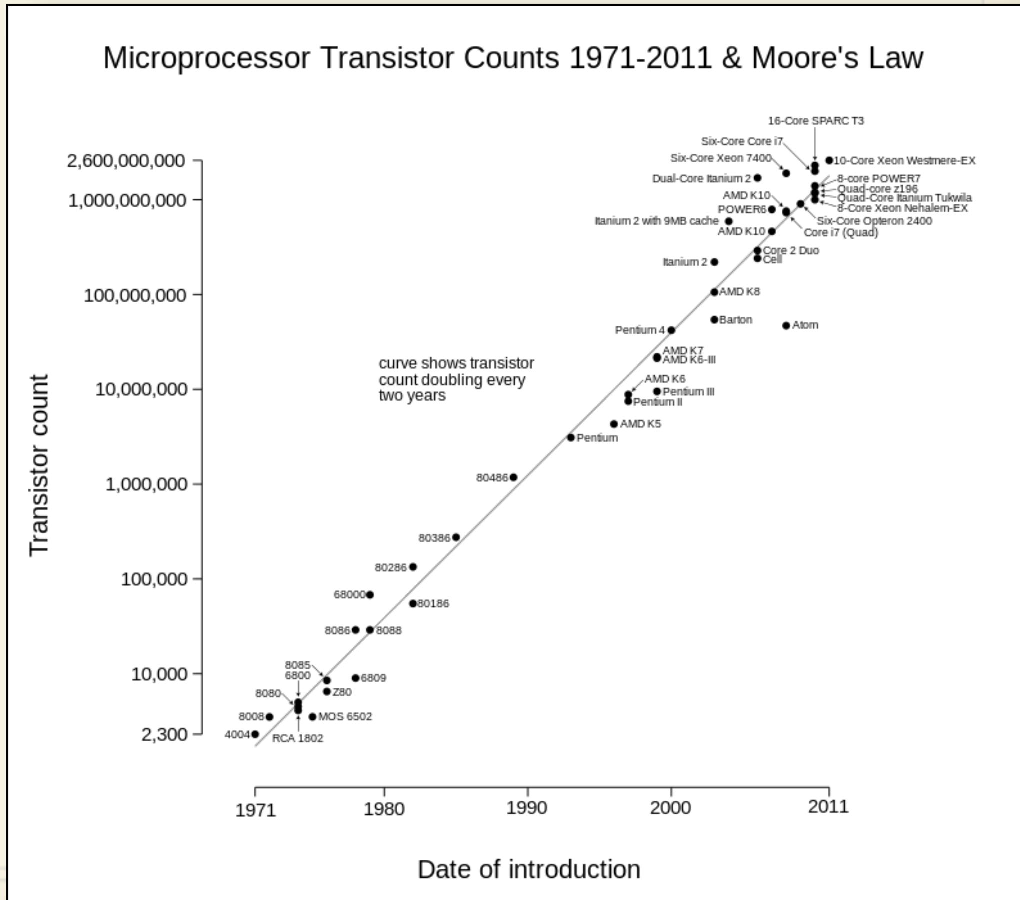
2016: ATLAS AM06 = 128K patterns/170 mm^2 = 753/mm^2

x 200 in 16 years

what's next?

3D integration?

but why do we need such a high density?



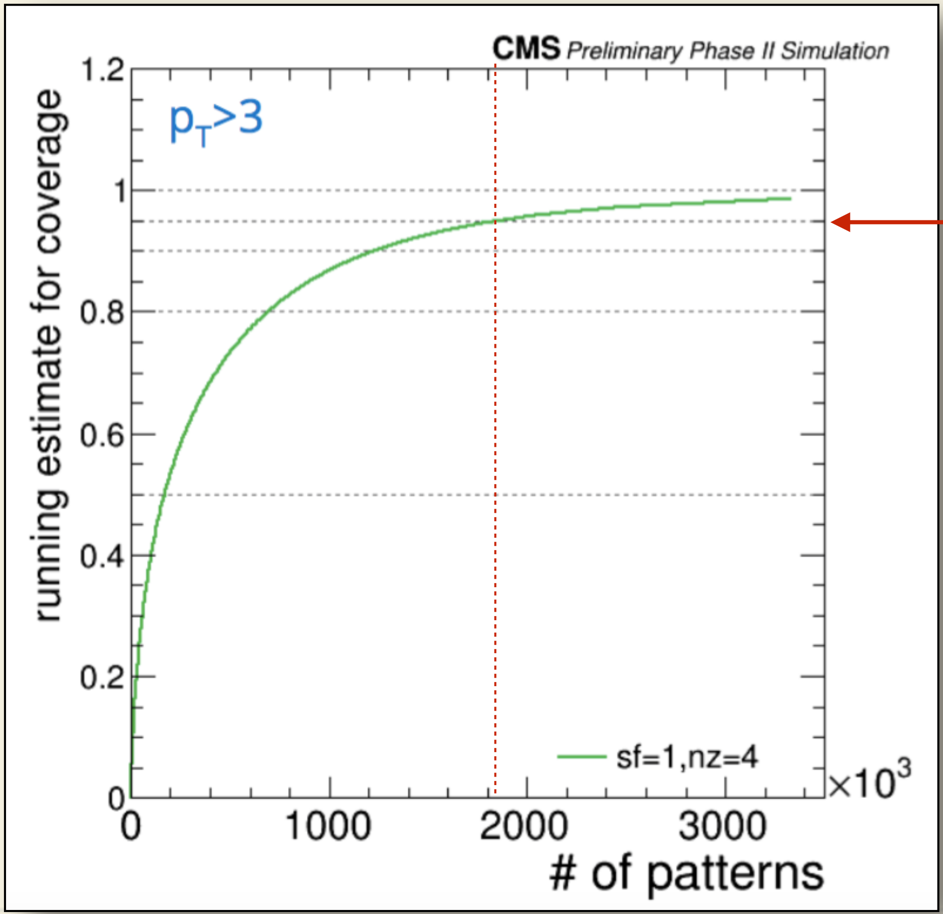
The Pattern Bank

what do we load in the AM ?

Building the pattern bank

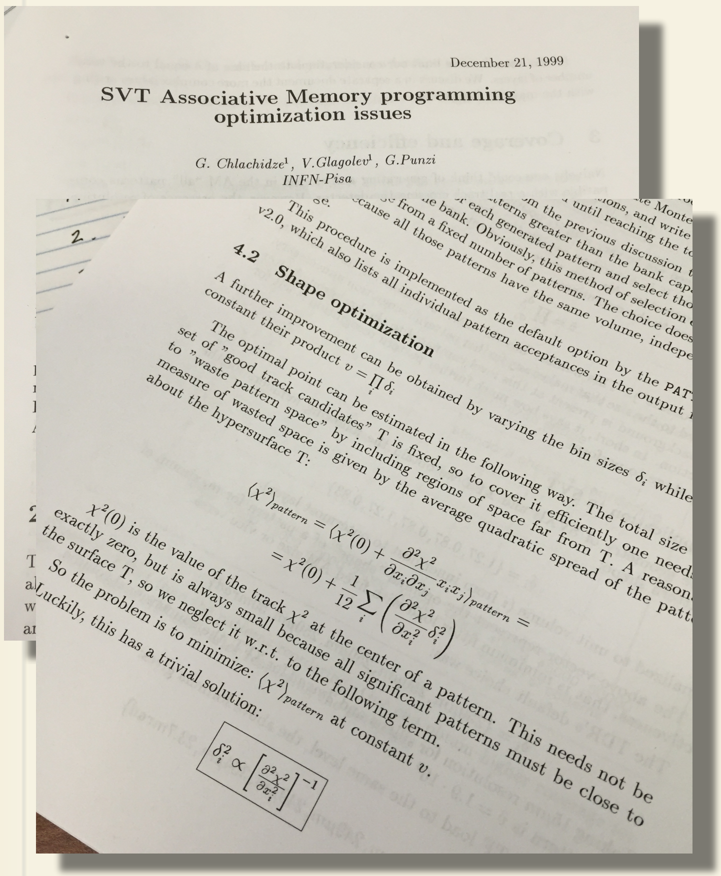
1. Define superstrips
2. Simulate single tracks from appropriate parameter region
3. Collect all patterns
4. Stop at target coverage

coverage:
fraction of tracks which fire a pattern already included in the AM



95%

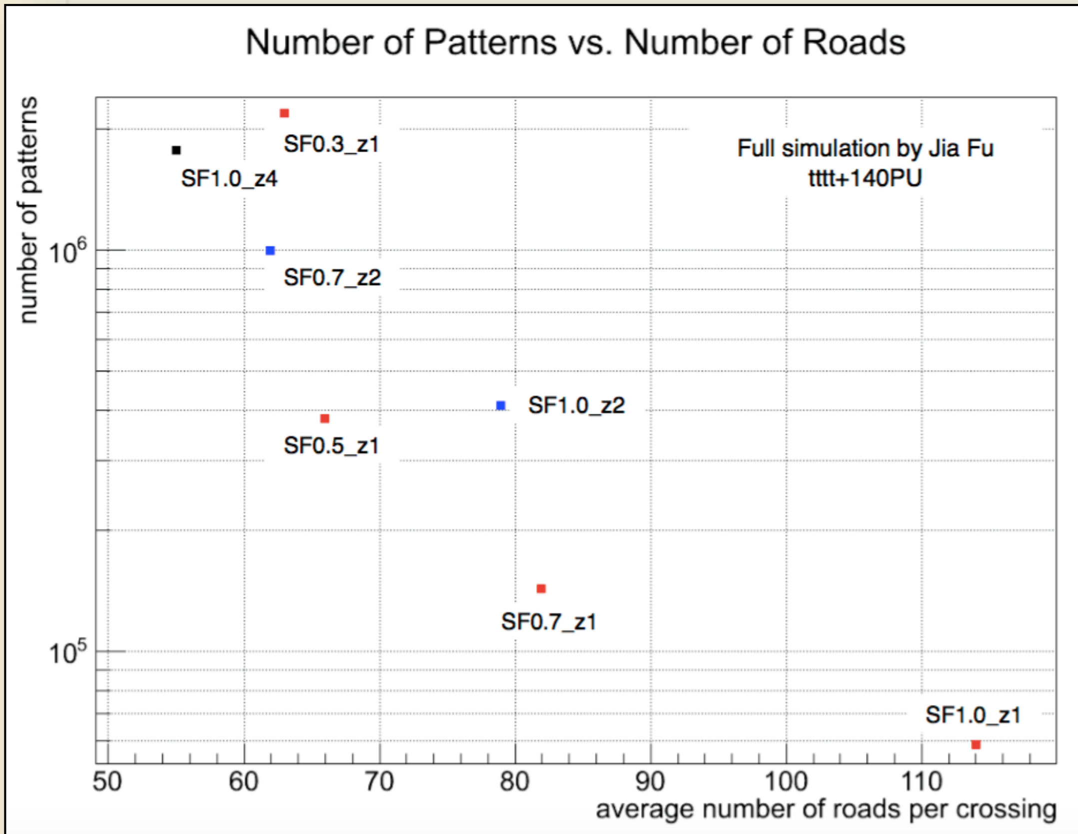
Superstrip size optimization

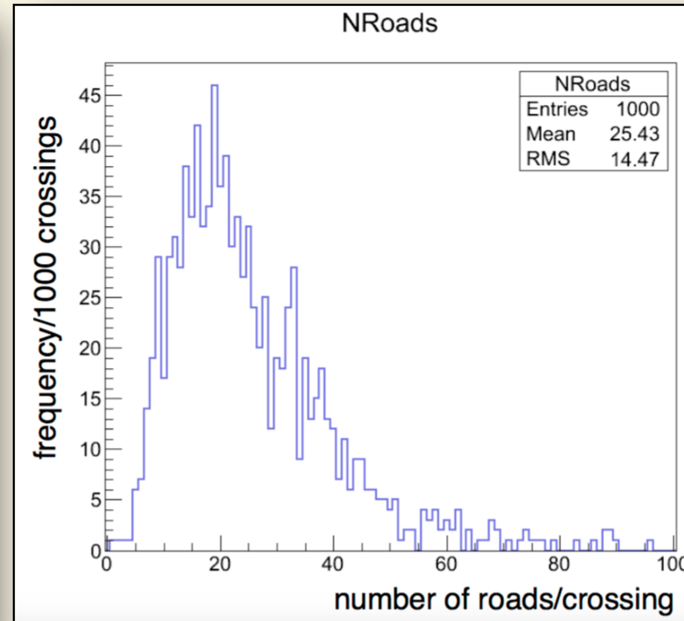


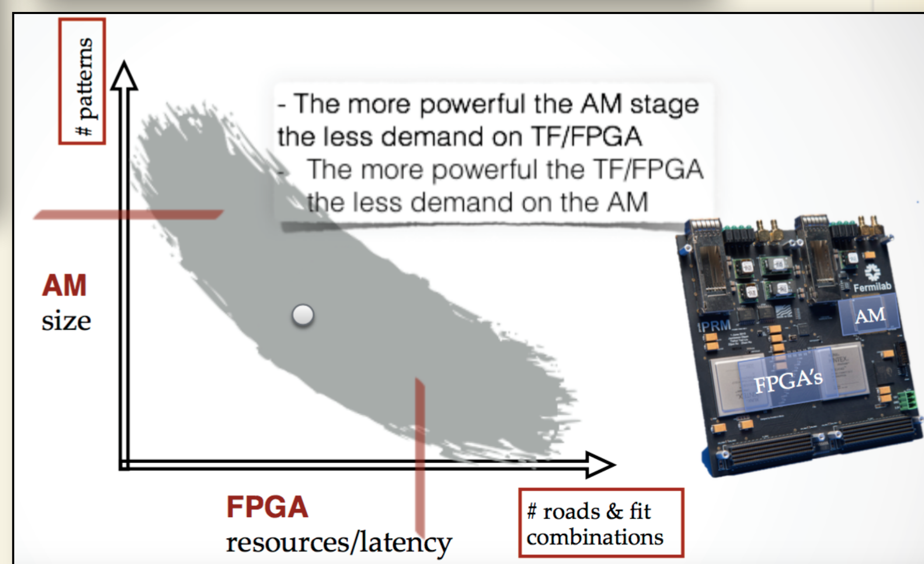
CDF_5201

- Goals
  - Maximize coverage
  - Minimize number of patterns
  - Minimize number of roads
- Rules of thumb
  - small occupancy
  - uniform occupancy
  - larger than resolution
- Optimal working point is searched by trial and error guided by intuition
- ← Interesting approximate theory exists but no exact solution

#patterns vs. #roads





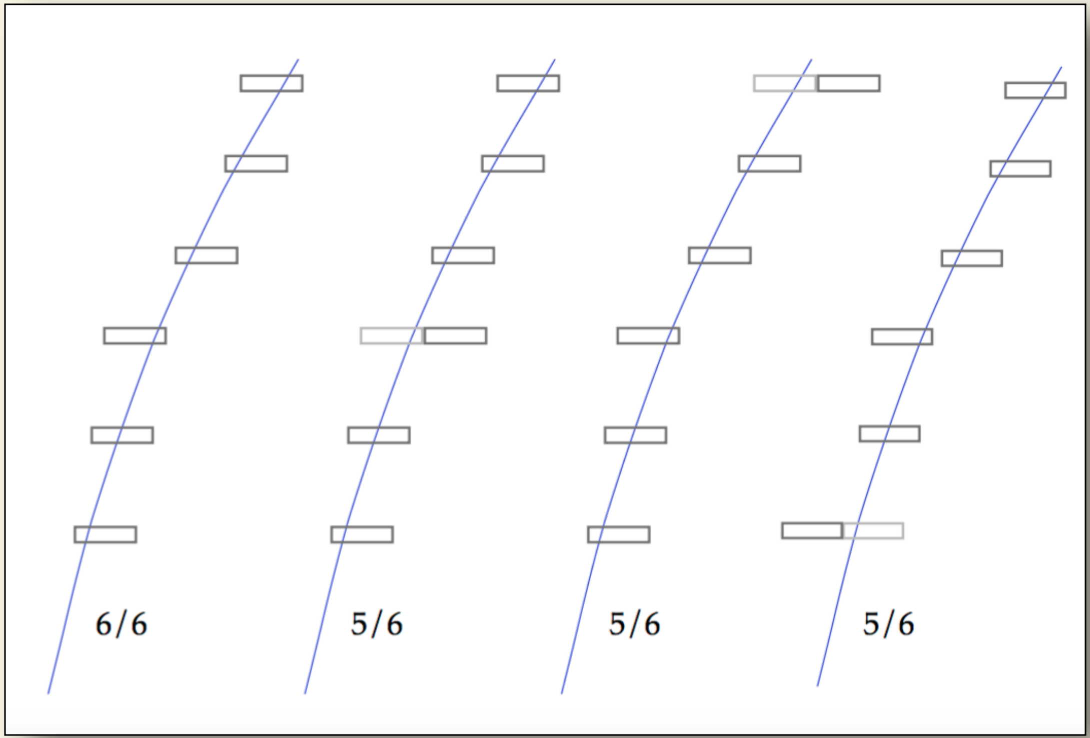


sharing the load

Majority Logic

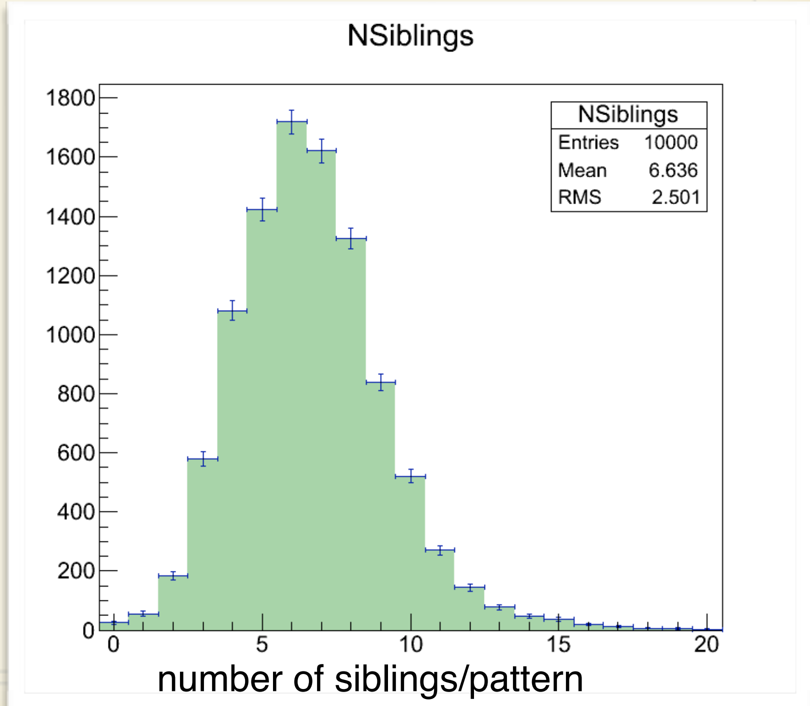
allow for missing hits:
e.g. fire patterns with 5 hits out of 6

Siblings and road proliferation



one good 6/6 track also fires several 5/6 patterns

"sibling" = each of two or more patterns differing only in one layer

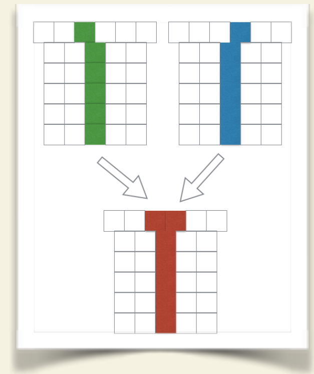


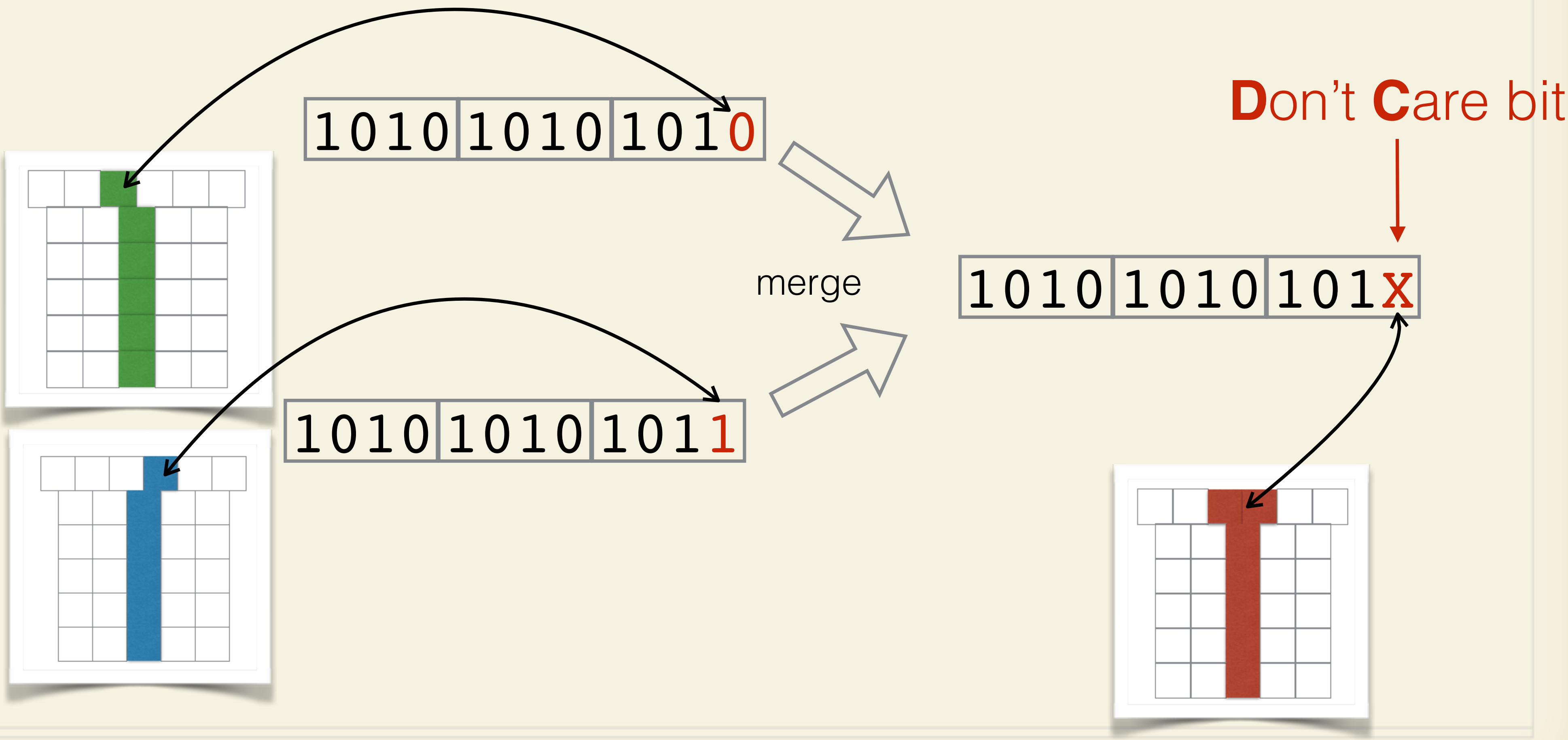
Major cause of road proliferation

Sibling Removal?

Merging two siblings

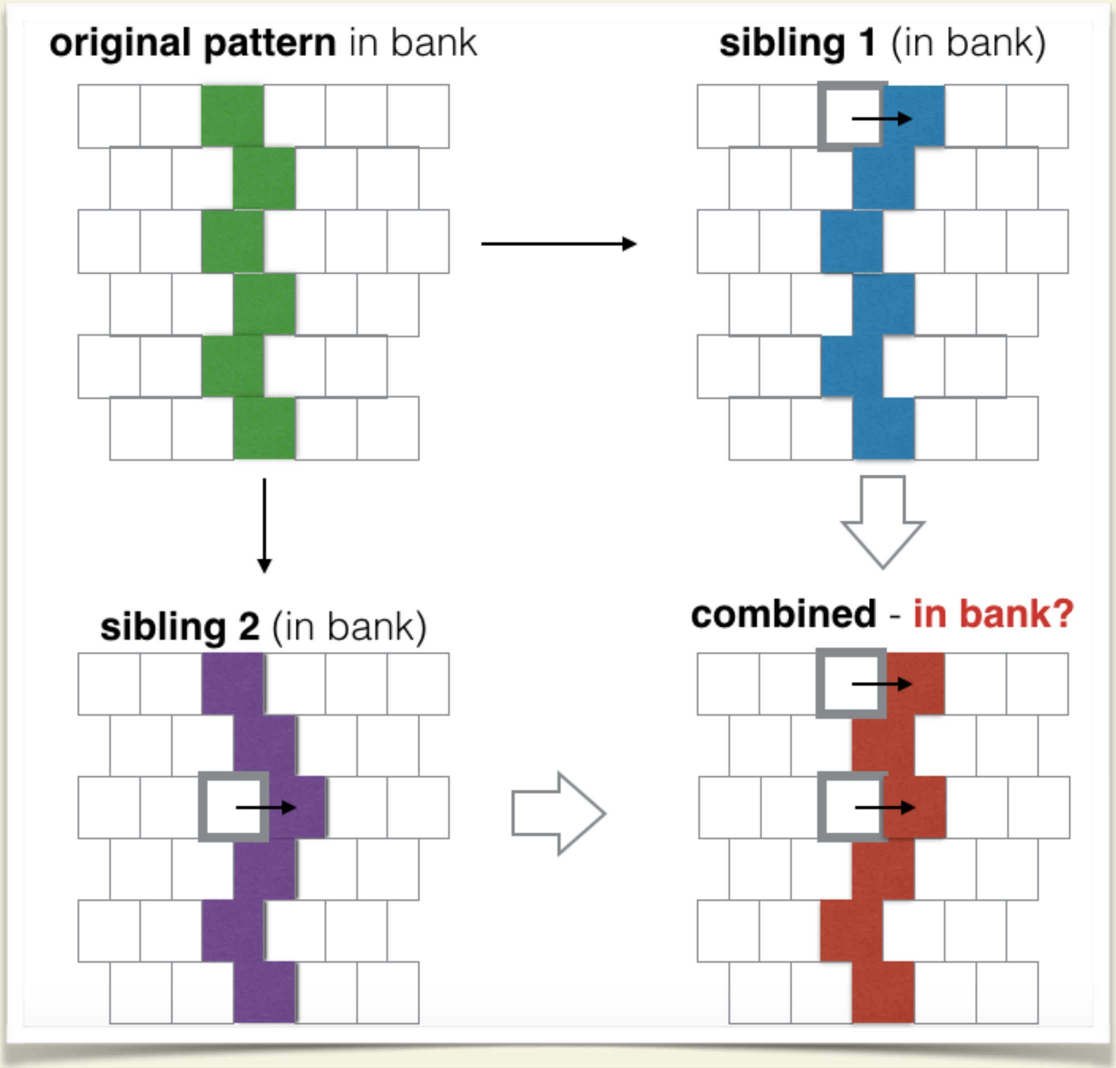
(DC bits introduced in the AM by Atlas-FTK)



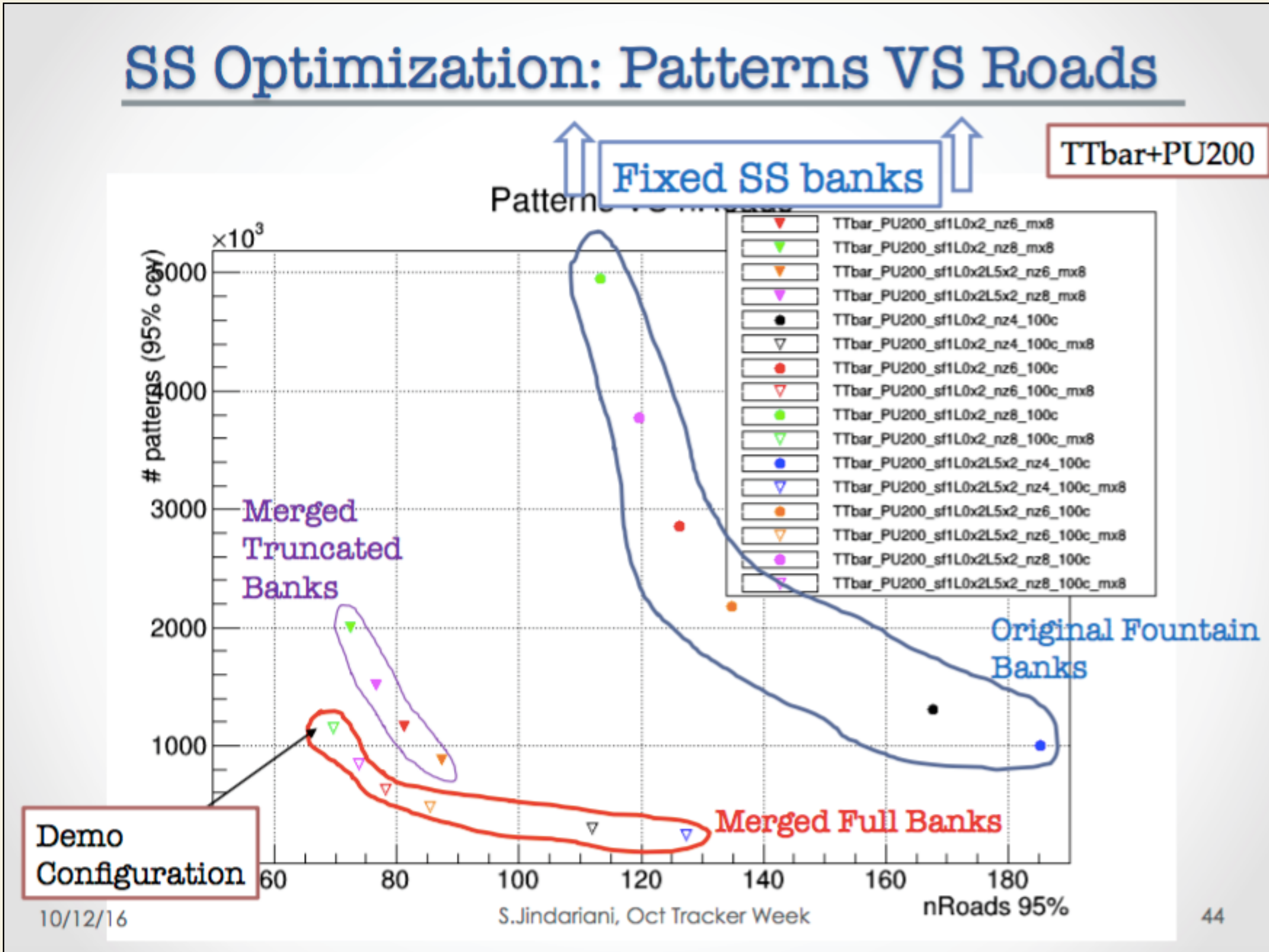


Merge x 4

two siblings in two different layers



not a sibling
of the original pattern

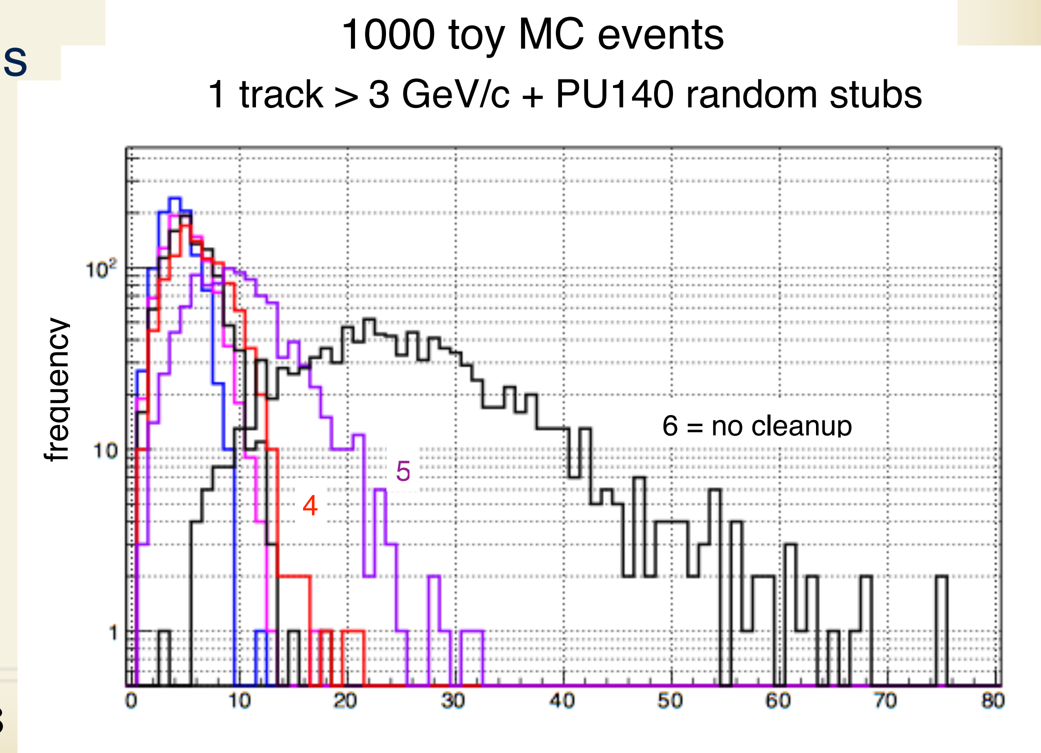


SS Optimization: Patterns VS Roads
Fixed SS banks
TTbar+PU200
×10^3
patterns (95% cov)
5000
4000
3000
2000
1000
60
80
100
120
140
160
180
nRoads 95%
TTbar_PU200_sf1L0x2_nz6_mx8
TTbar_PU200_sf1L0x2_nz8_mx8
TTbar_PU200_sf1L0x2L5x2_nz6_mx8
TTbar_PU200_sf1L0x2L5x2_nz8_mx8
TTbar_PU200_sf1L0x2_nz4_100c
TTbar_PU200_sf1L0x2_nz4_100c_mx8
TTbar_PU200_sf1L0x2_nz6_100c
TTbar_PU200_sf1L0x2_nz6_100c_mx8
TTbar_PU200_sf1L0x2_nz8_100c
TTbar_PU200_sf1L0x2_nz8_100c_mx8
TTbar_PU200_sf1L0x2L5x2_nz4_100c
TTbar_PU200_sf1L0x2L5x2_nz4_100c_mx8
TTbar_PU200_sf1L0x2L5x2_nz6_100c
TTbar_PU200_sf1L0x2L5x2_nz6_100c_mx8
TTbar_PU200_sf1L0x2L5x2_nz8_100c
TTbar_PU200_sf1L0x2L5x2_nz8_100c_mx8
Merged Truncated Banks
Original Fountain Banks
Merged Full Banks
Demo Configuration
10/12/16
S.Jindariani, Oct Tracker Week
44

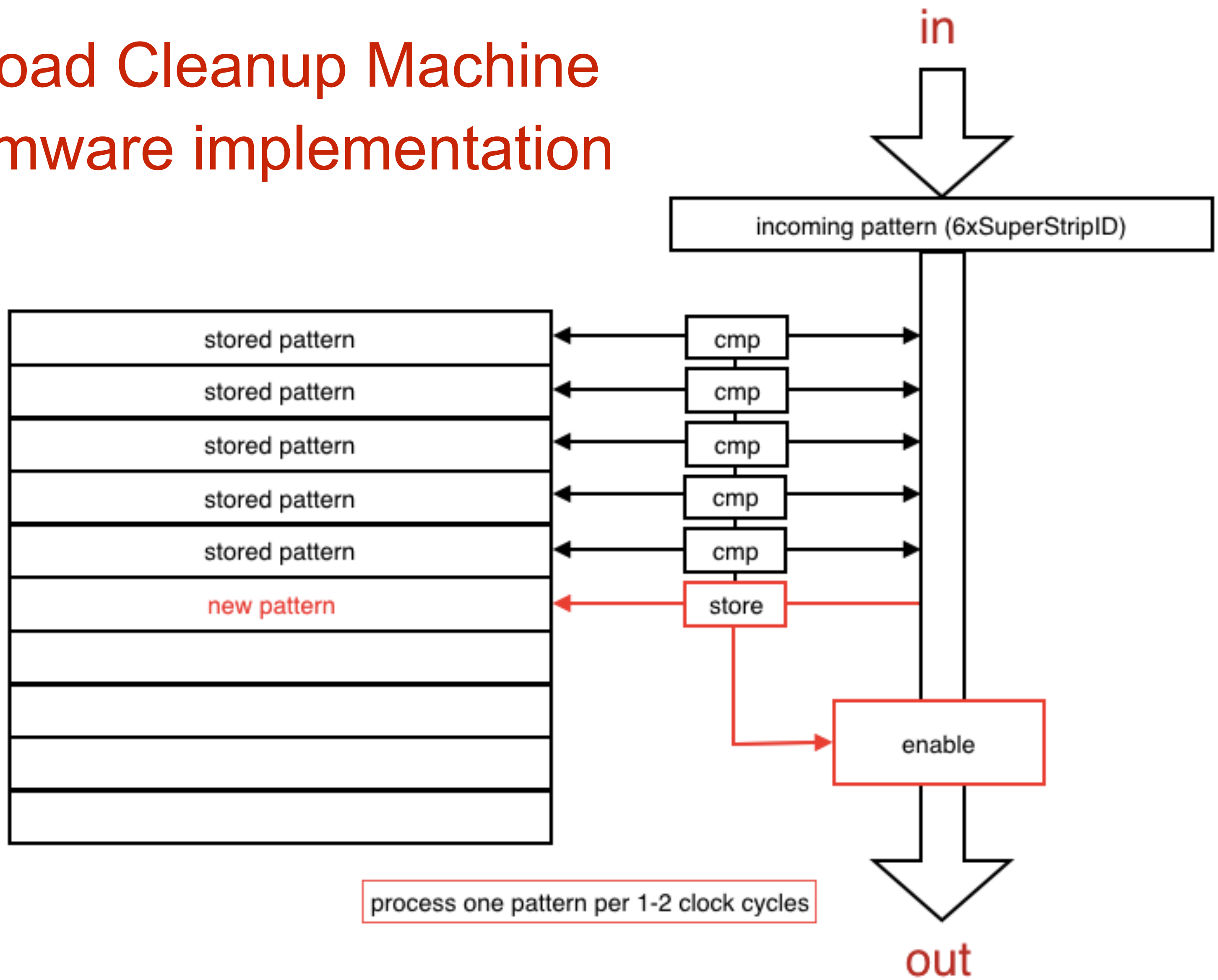
Road Cleanup Algorithm

1. Sort roads 6/6 first
2. Output first road and store it in the list of good roads
3. Examine each incoming road sequentially one at a time, compare it to all stored good roads to find if any one has *n* or more superstrips in common with this road
4. if yes, just ignore this road
5. if no, output this road and append it to the good road list
6. at end-of-event clear the list of good roads

optimal *n* found through simulation



Road Cleanup Machine firmware implementation

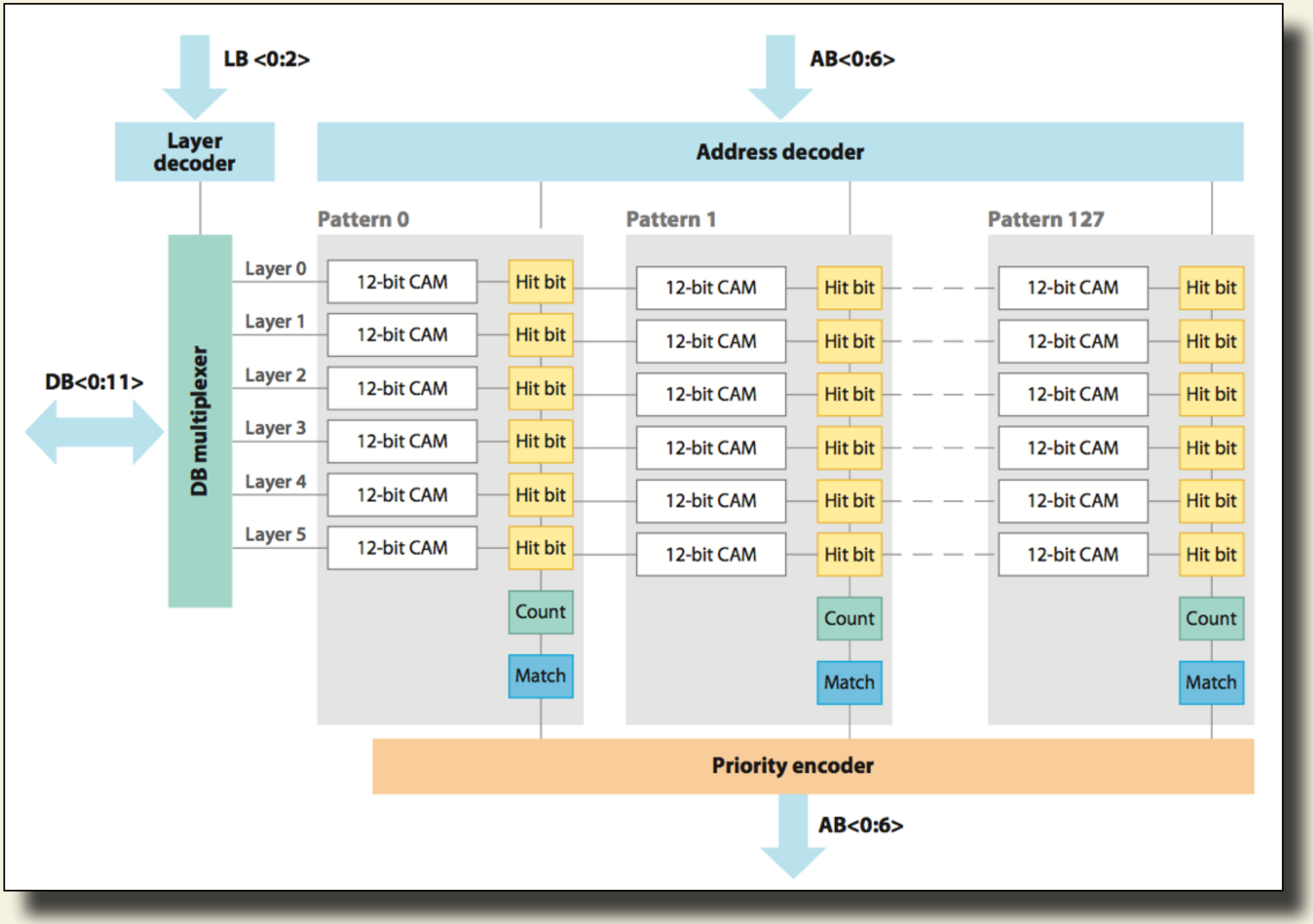


still there might be a better way…..

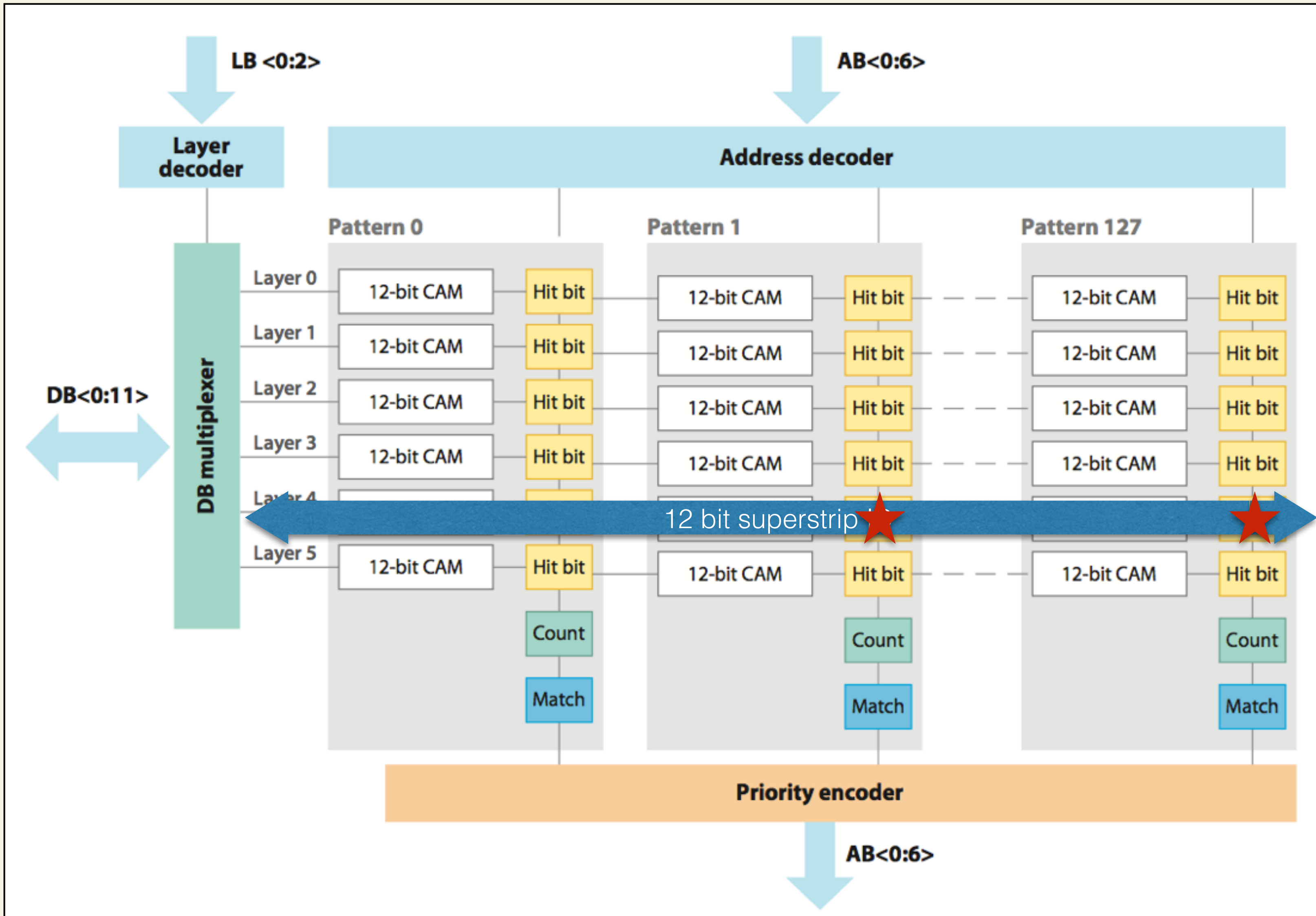
AM+

a possible solution to
the road proliferation problem

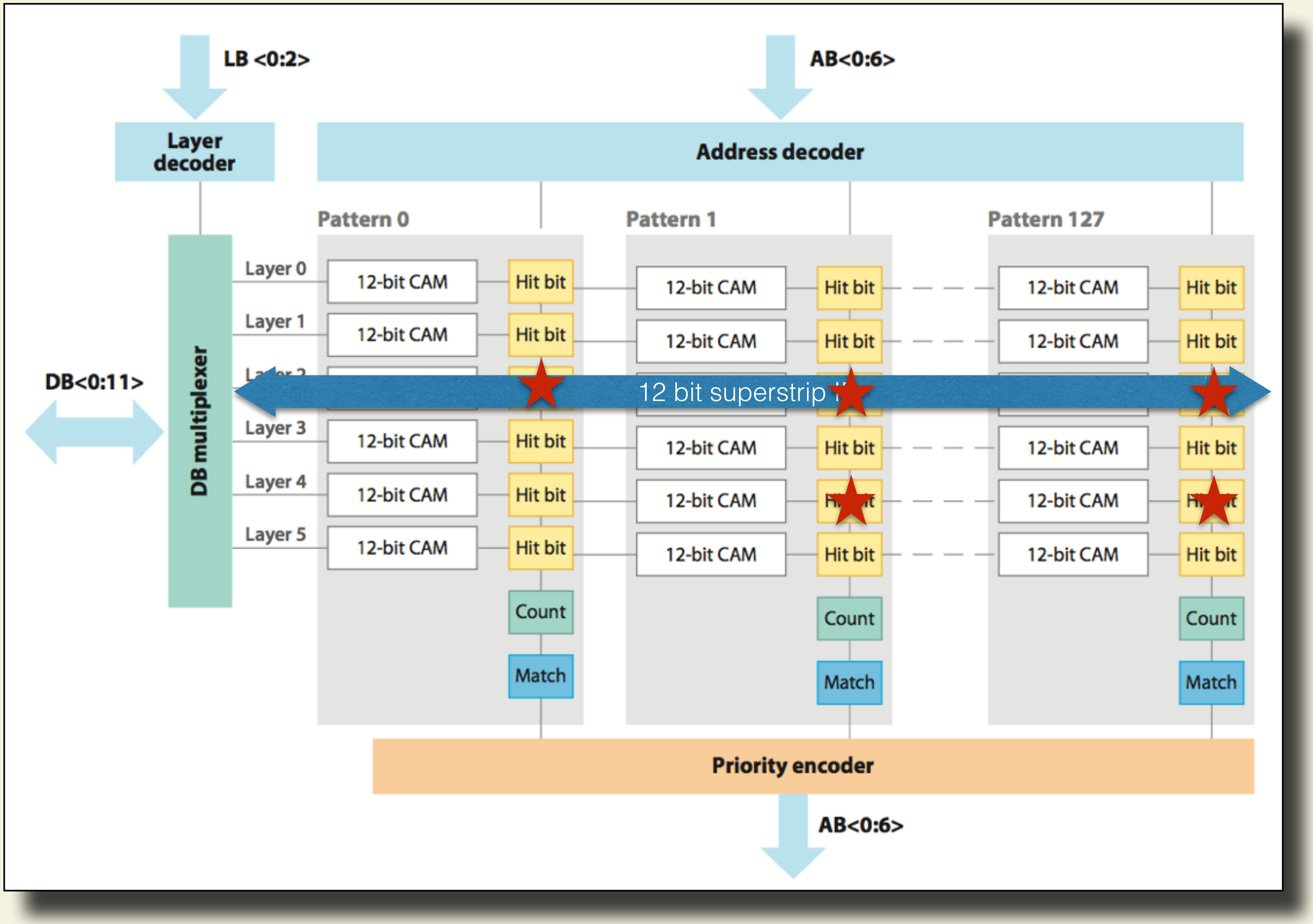
AM+ working principle



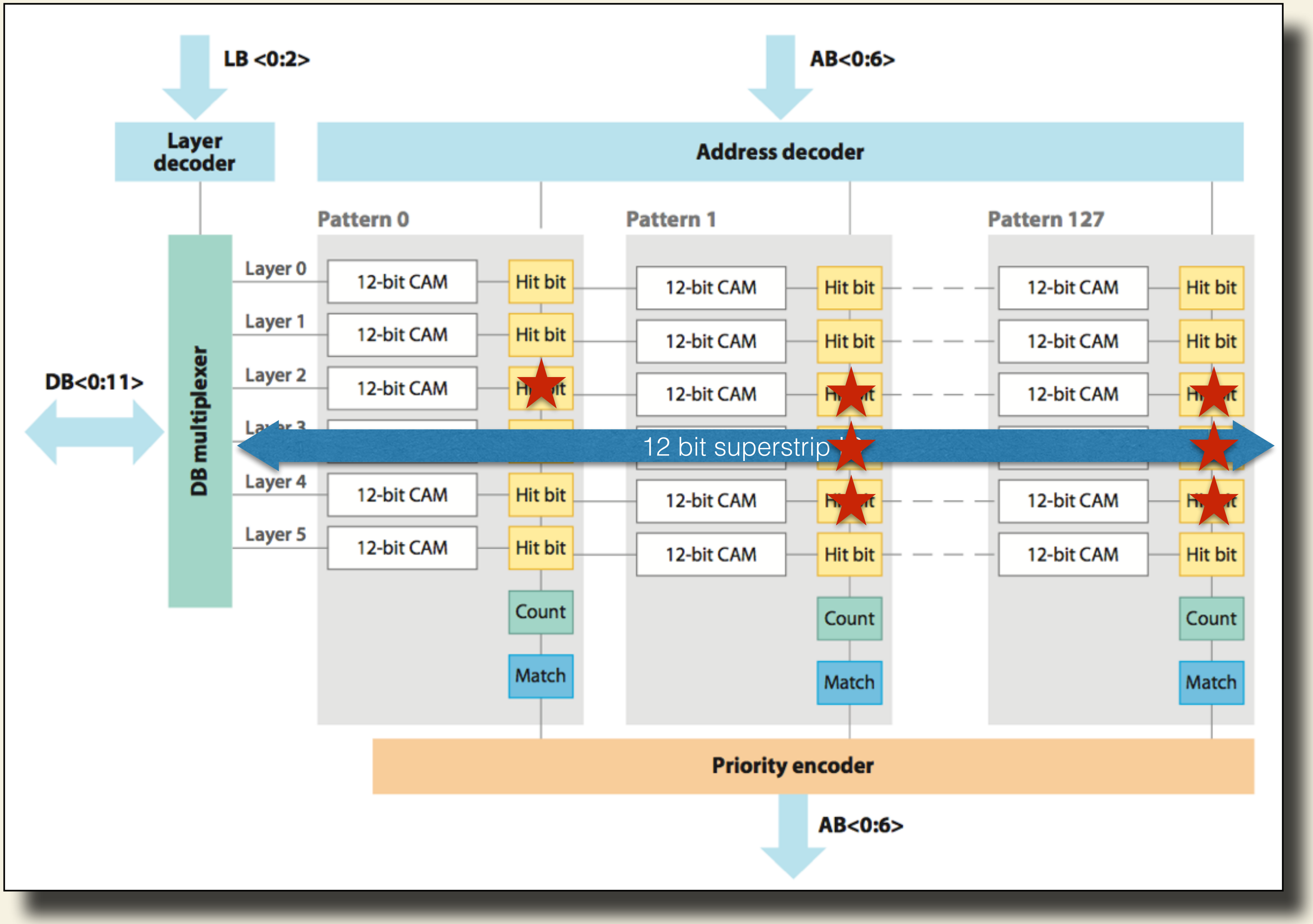
AM+ working principle



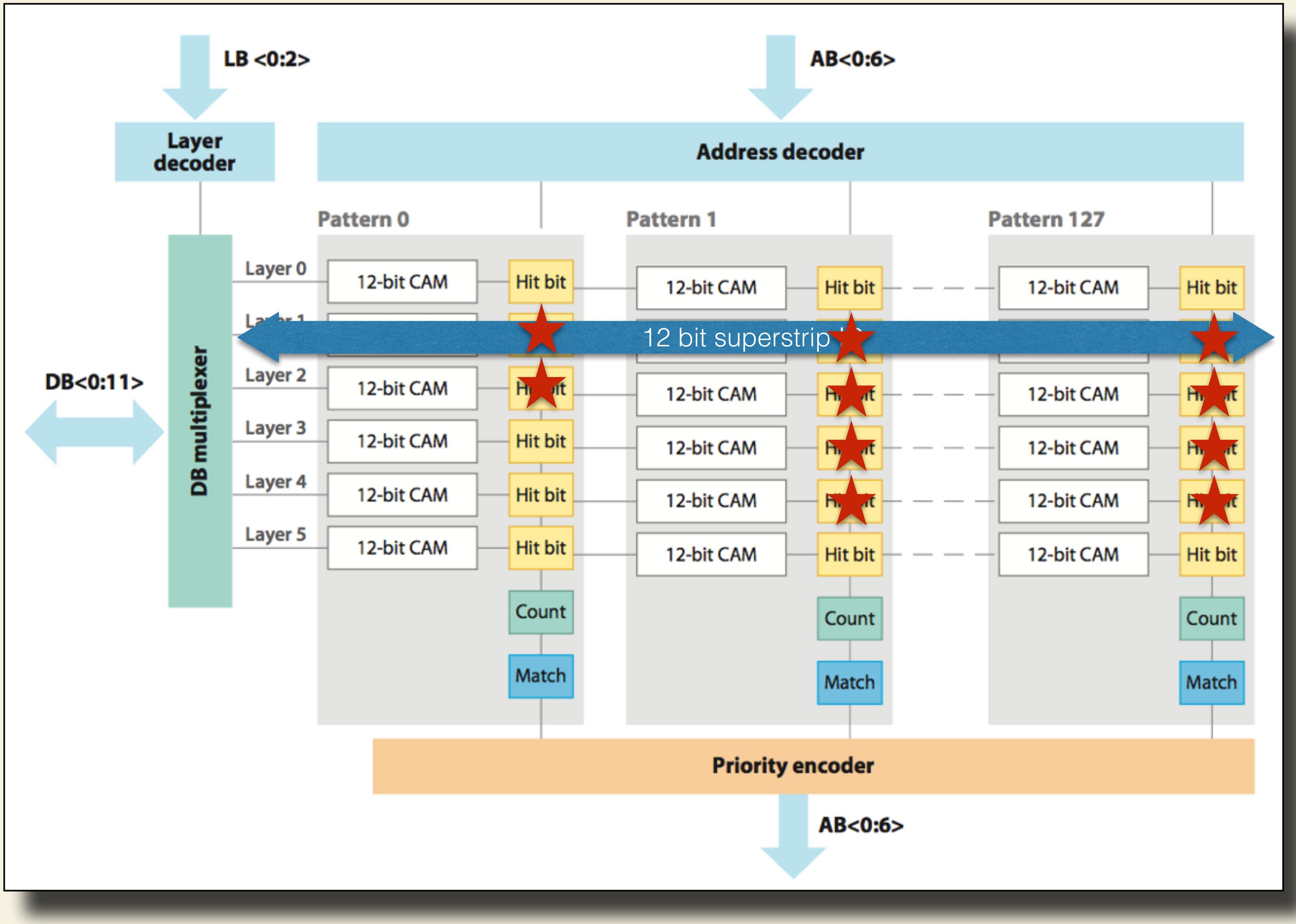
AM+ working principle



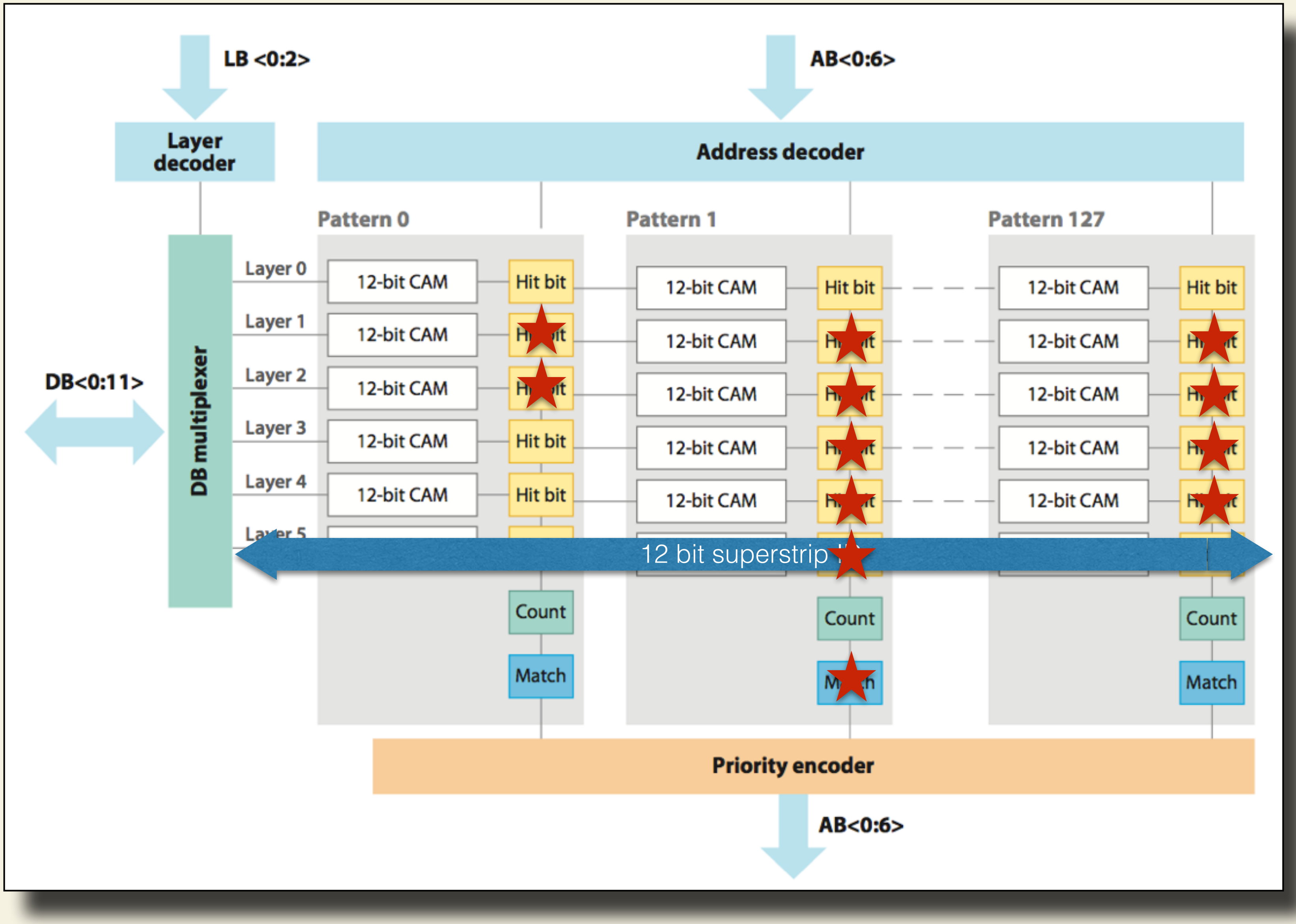
AM+ working principle



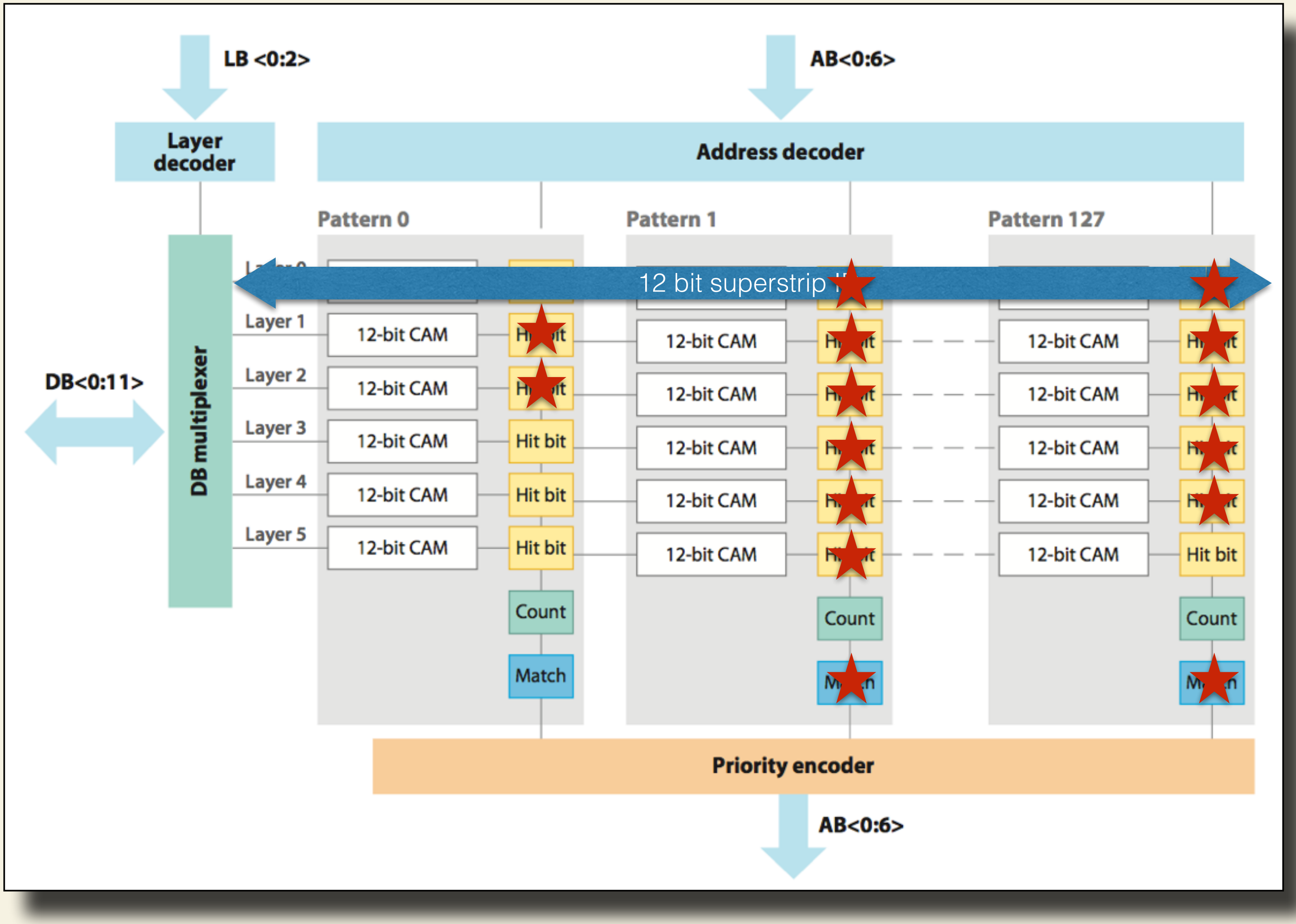
AM+ working principle



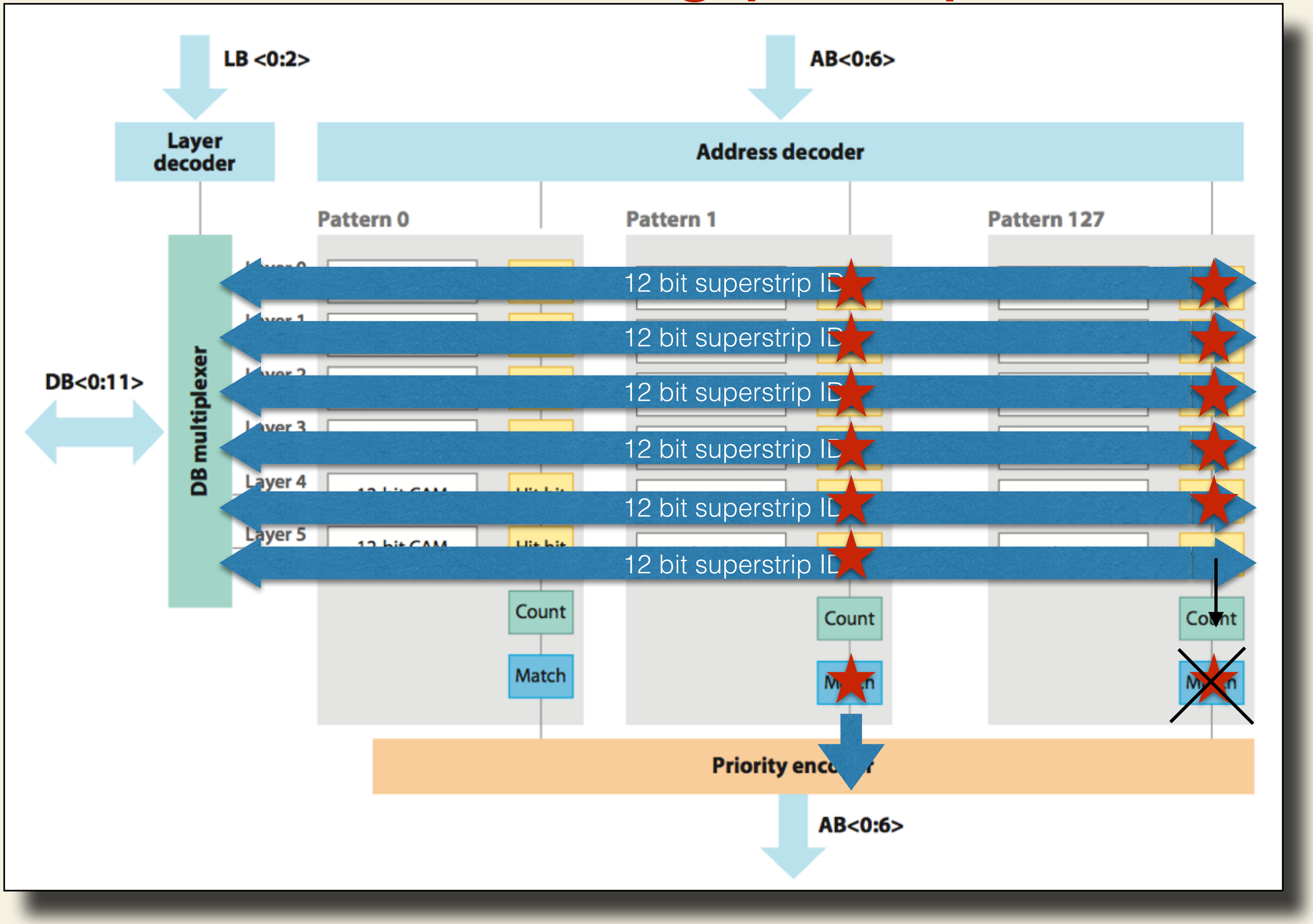
AM+ working principle



AM+ working principle



AM+ working principle



AM+

- The AM+ can potentially stop the problem of road proliferation due to siblings before it even starts
- 5/6 siblings of a 6/6 track are not even output by the AM saving many precious clock cycles
- Most of the logic needed is already in the current version of the AM
- Simulations indicate that loss in efficiency should be small
- Seems like an interesting possible future development

That is all

Thanks for listening