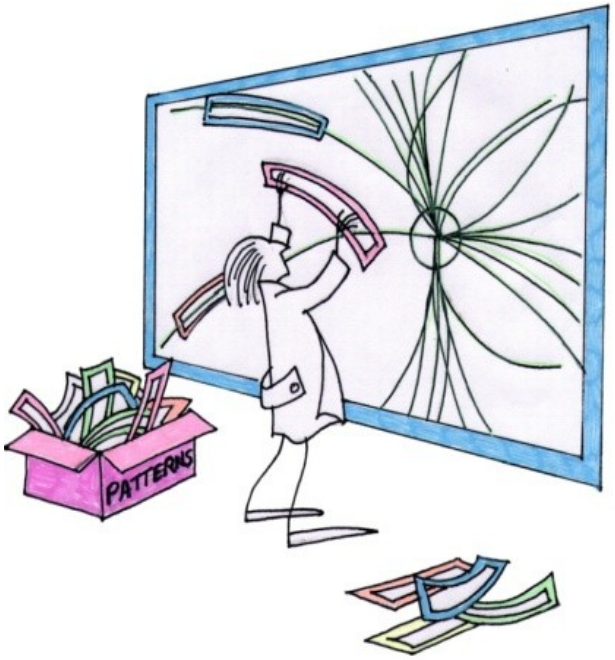
AMchip04: a new generation associative memory chip for HEP applications



F. Crescioli

Outline

- Online tracking at Hadron Colliders
- SVT/FTK algorithm
- Fast pattern recognition with dedicated hw
- AMchip history
- AMchip04: the new FTK variable resolution associative memory chip
- Beyond AMchip04
- Conclusions

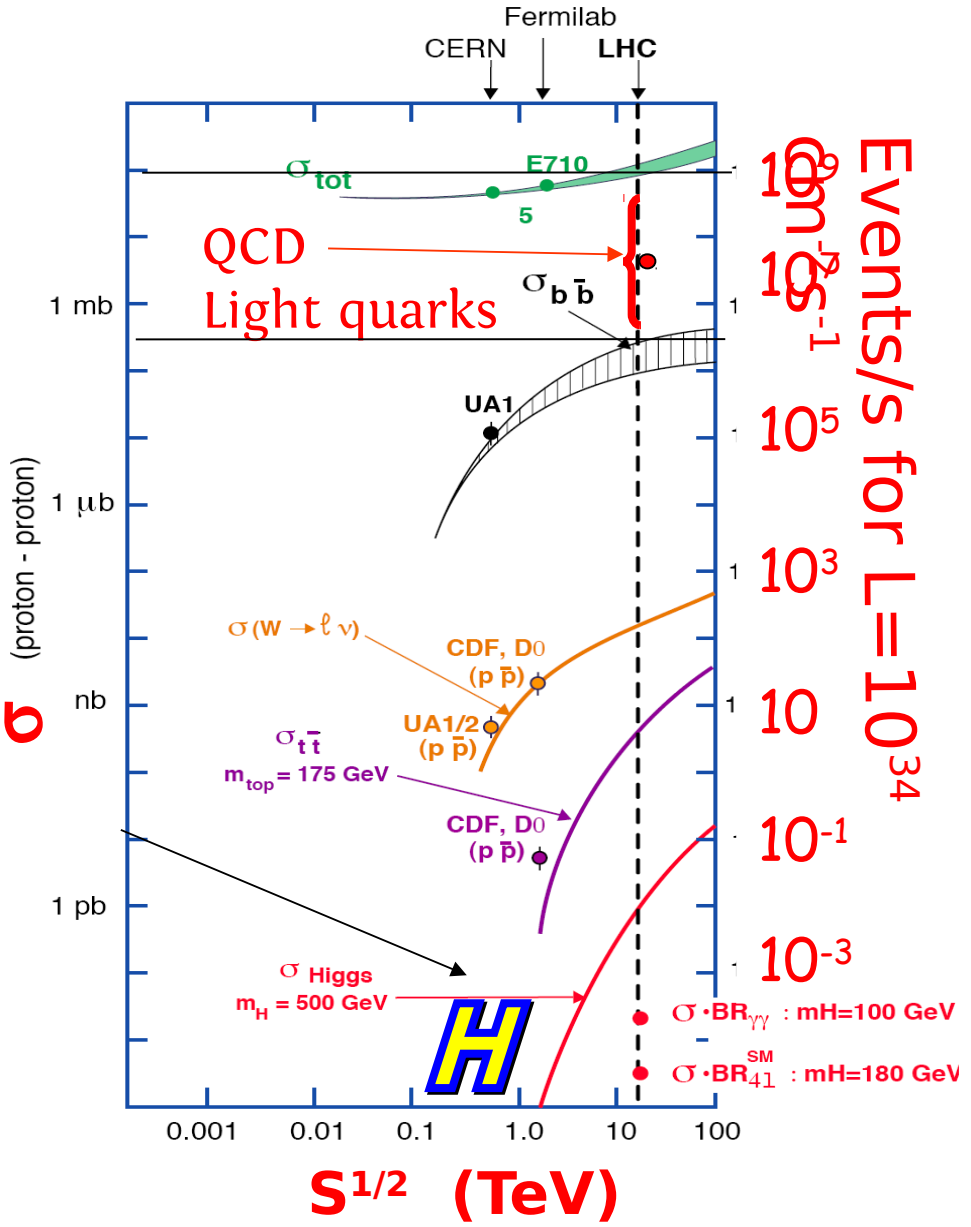
ONLINE TRACKING @ HADRON COLLIDERS

Search for rare SM or predicted BSM processes push the collider's intensity to new frontiers

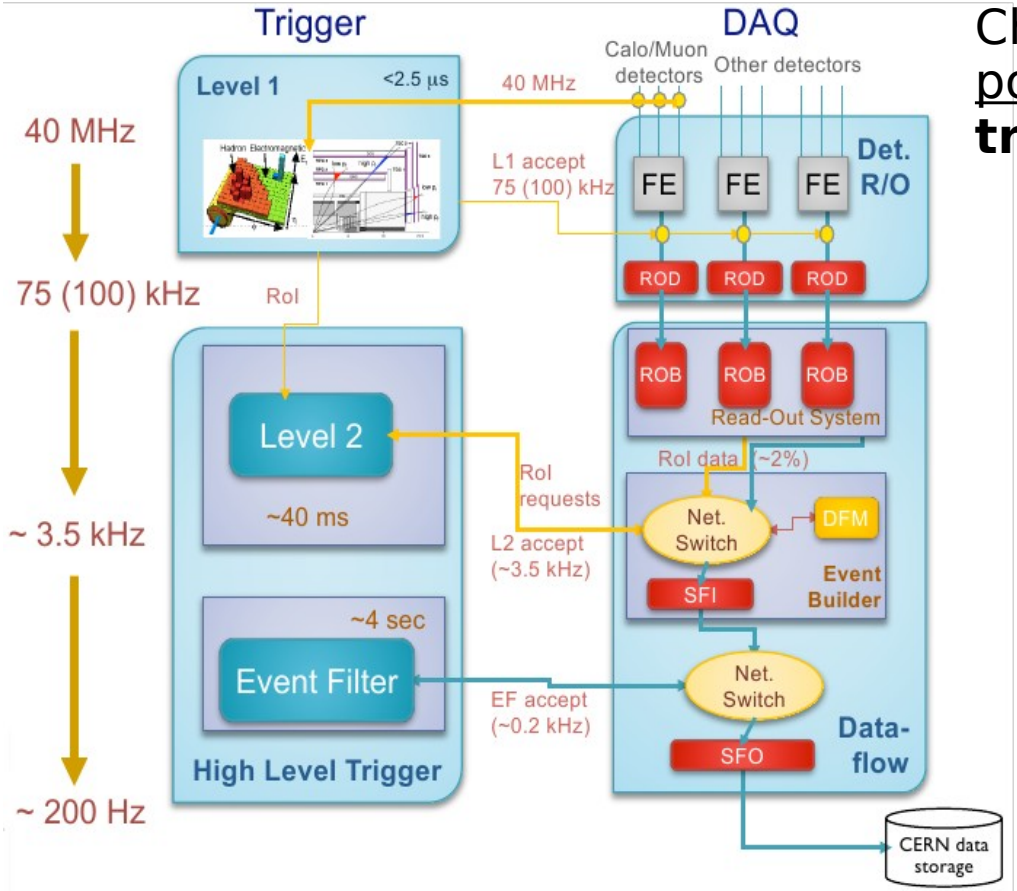
Rare processes are overwhelmed by well-known processes

- 8-9 orders of magnitude between Higgs and total cross-section
- Need to prioritize the physics output and leave flexibility for the unexpected

Need **sophisticated trigger techniques** to maximize the bandwidth for interesting events



ONLINE TRACKING @ HADRON COLLIDERS

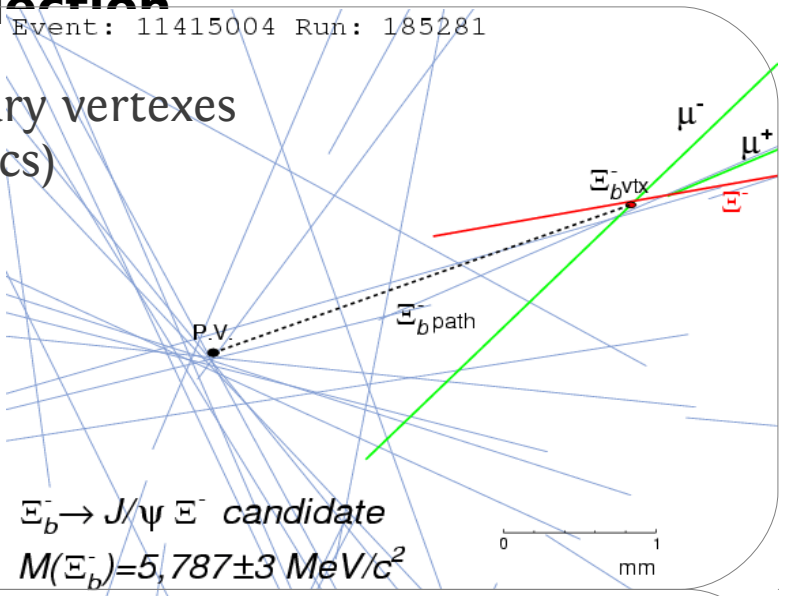


Track reconstruction is a complex and time-consuming computing task to perform in a short time budget.

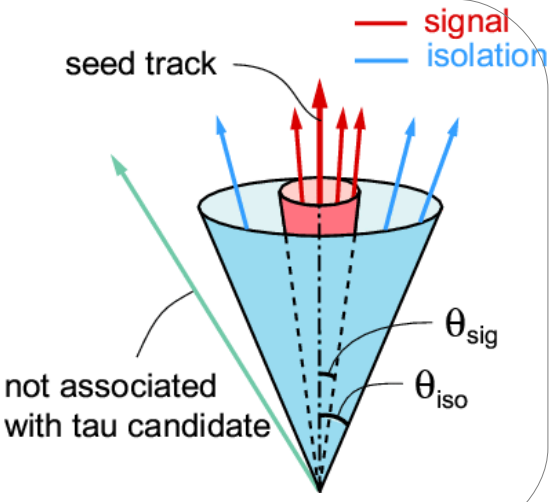
Hardware processors have been conceived for this task.
Examples: SVT (CDF), FTK (ATLAS)

Charged particle trajectories provide very powerful information for **sophisticated trigger selection**

Secondary vertexes
(b physics)

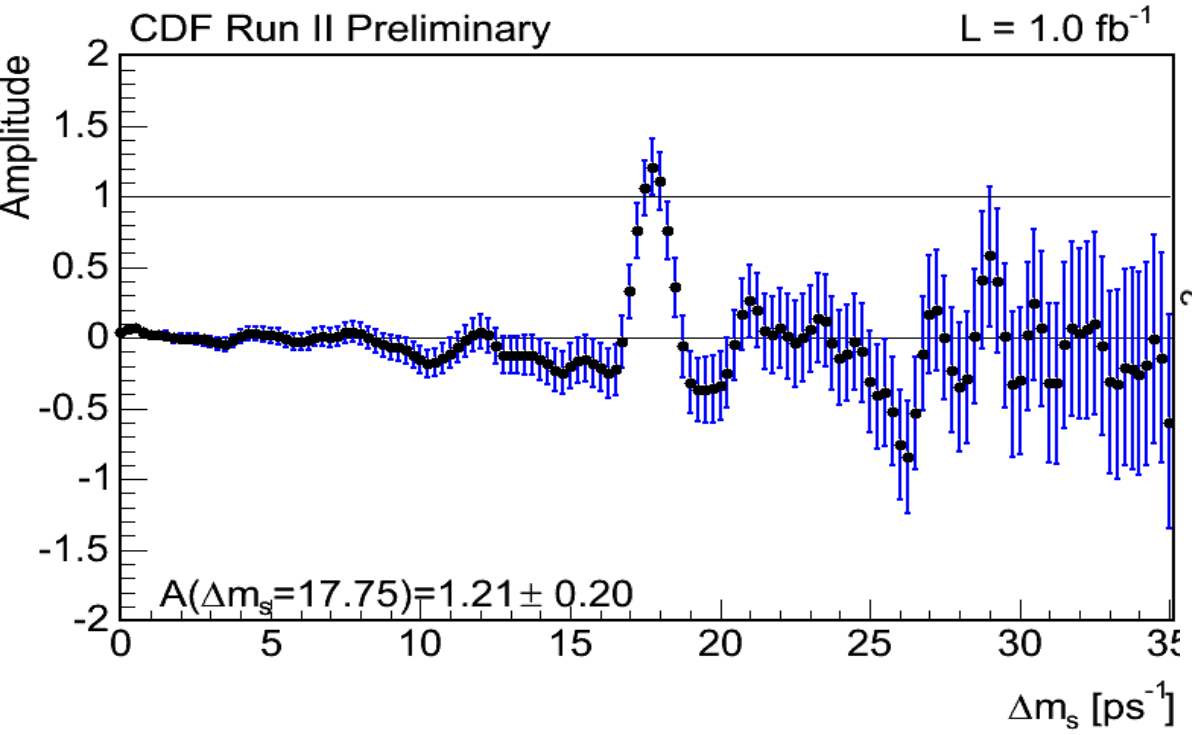


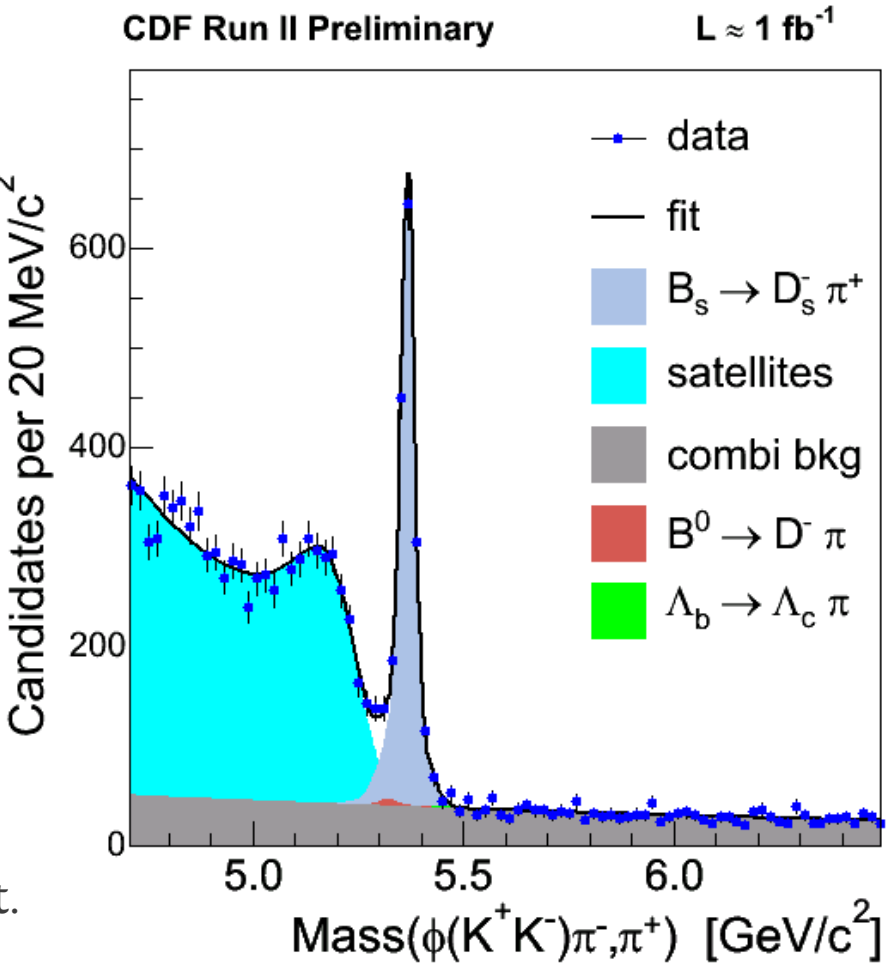
Isolation-based
Algorithms
(tau, muons)



Online tracking @ hadron colliders

SVT @ CDF





Online tracking (XFT @ L1 and SVT @ L2) has been a key element of many CDF analysies.

Example: the Bs oscillation frequency measurement. Many hadronic Bs collected thanks to SVT-based two-track trigger.

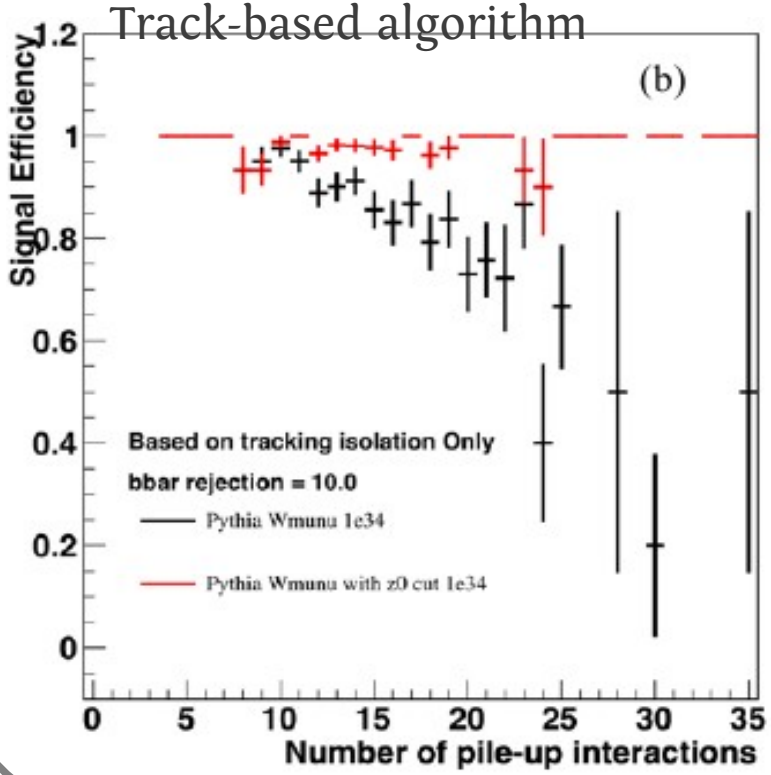
Online tracking @ hadron colliders

FastTracKer (FTK) has been recently approved by the ATLAS collaboration as an upgrade for its Level-2 trigger.

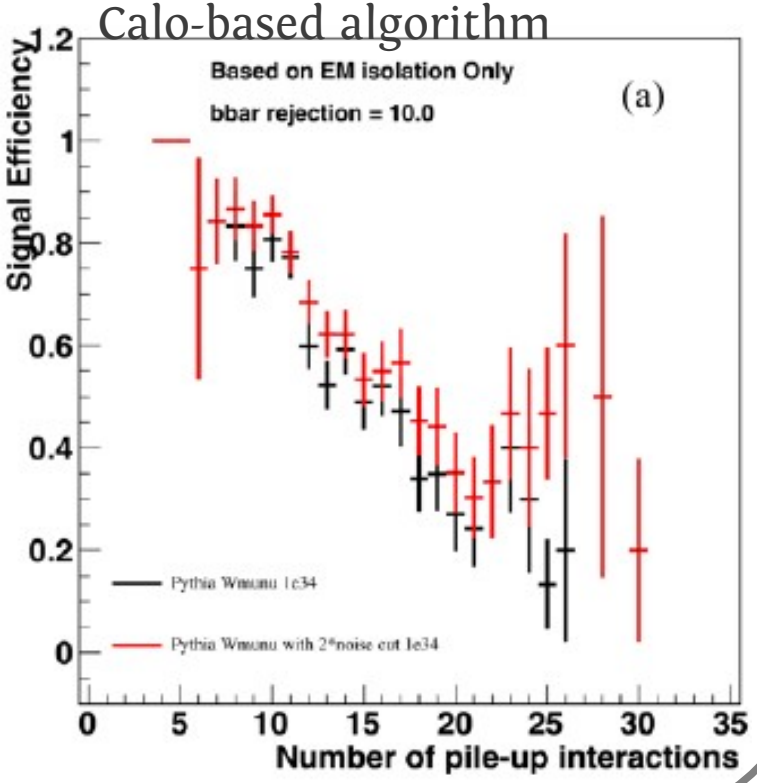
Online tracking is very important for high-luminosity, FTK is designed to handle track reconstruction up to 3×10^{34} $cm^{-2}s^{-1}$

Isolated muon trigger efficiency wrt pile-up

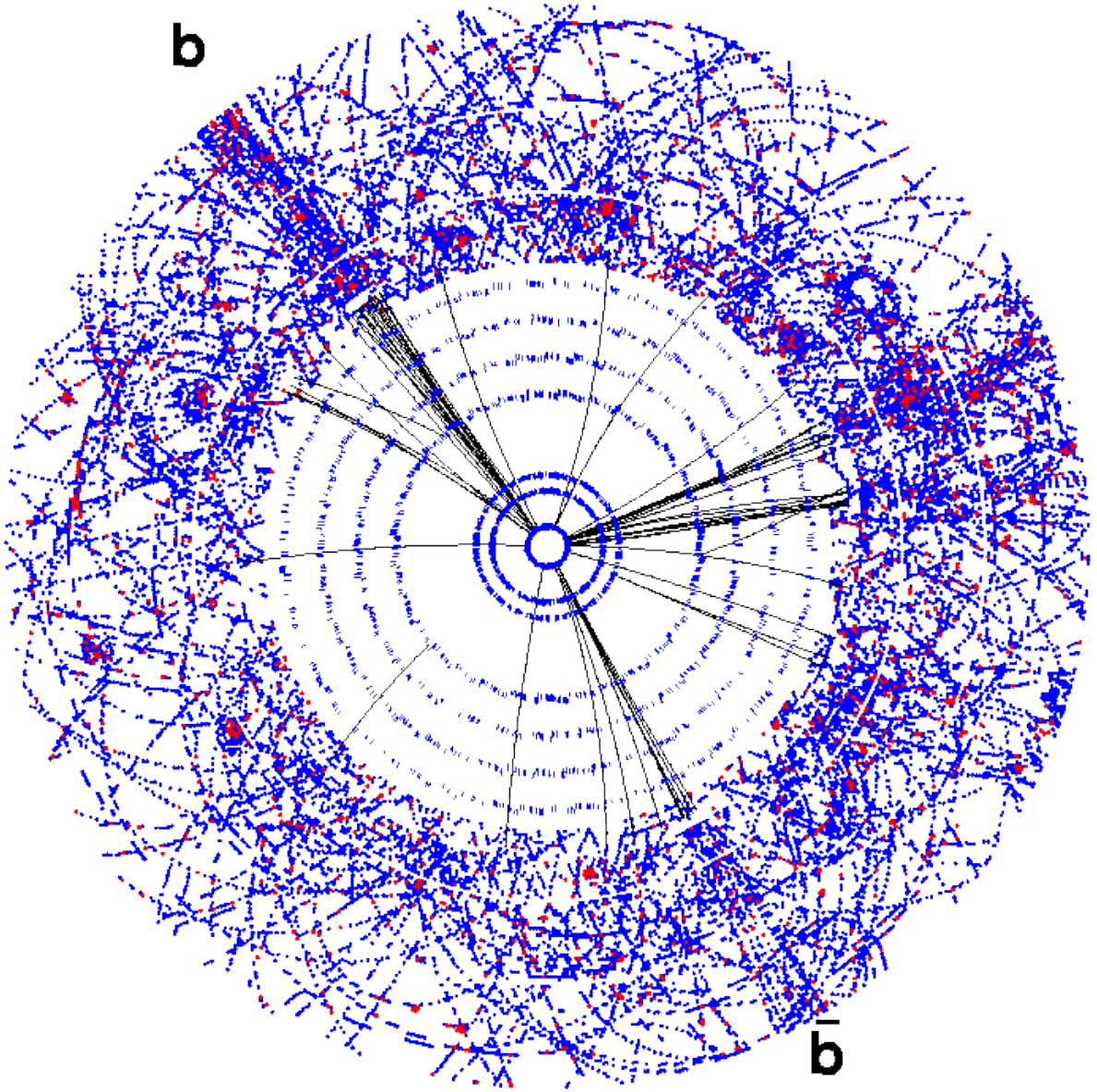
Track-based algorithm



Calo-based algorithm



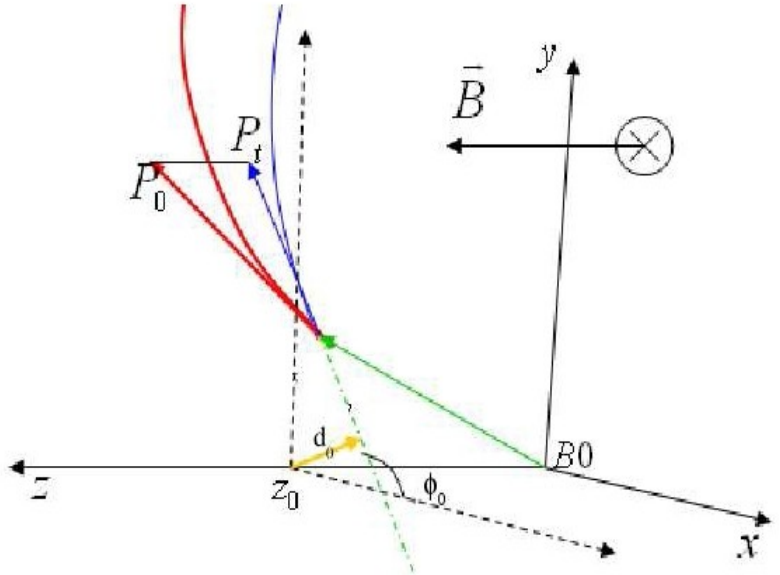
Online tracking @ hadron colliders



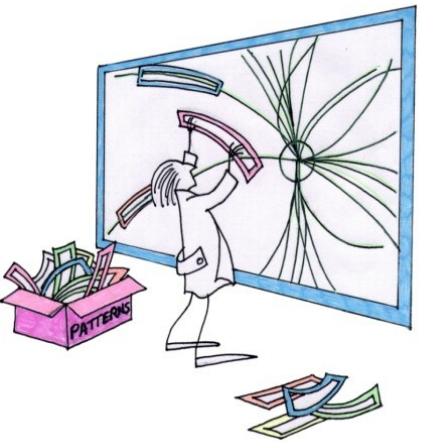
A Tracking processor must:

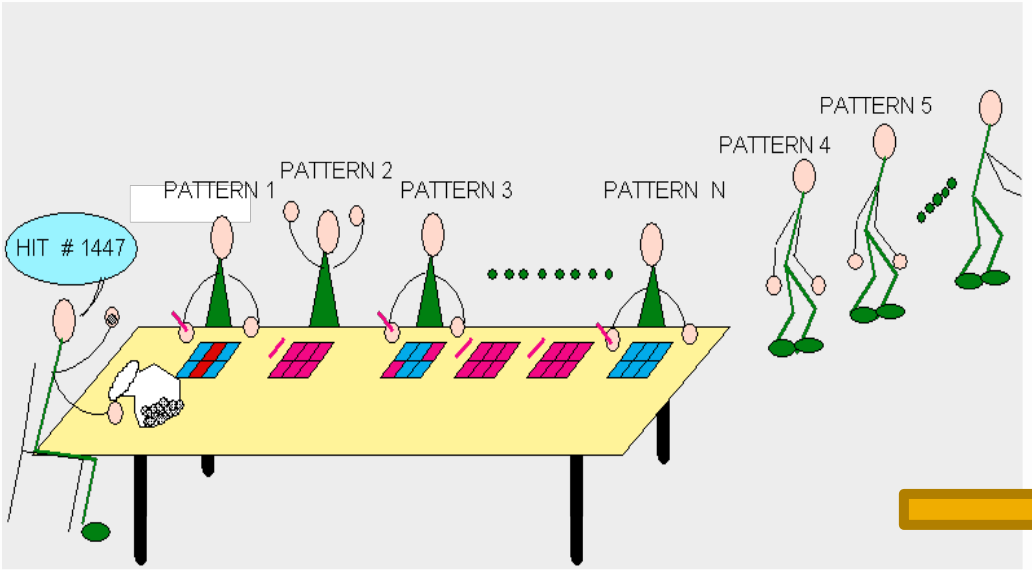
- recognize collection of hits belonging to a particle trajectory (pattern recognition)

- reconstruct particle trajectory parameters (origin, momentum) (track fit)



FTK/SVT ALGORITHM





An associative memory (AM) chip allows a full parallel search in the pattern bank.

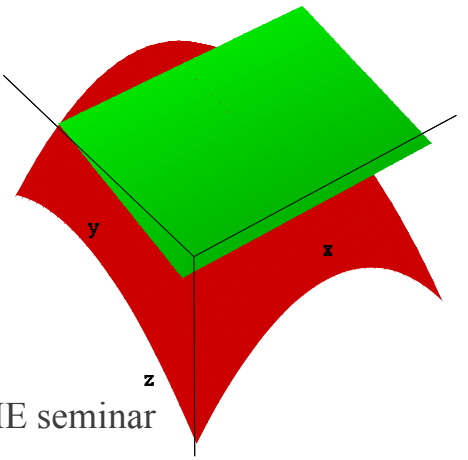
Similar to commercial CAMs + extended functionalities.

In FTK, tracking is divided into two separate steps

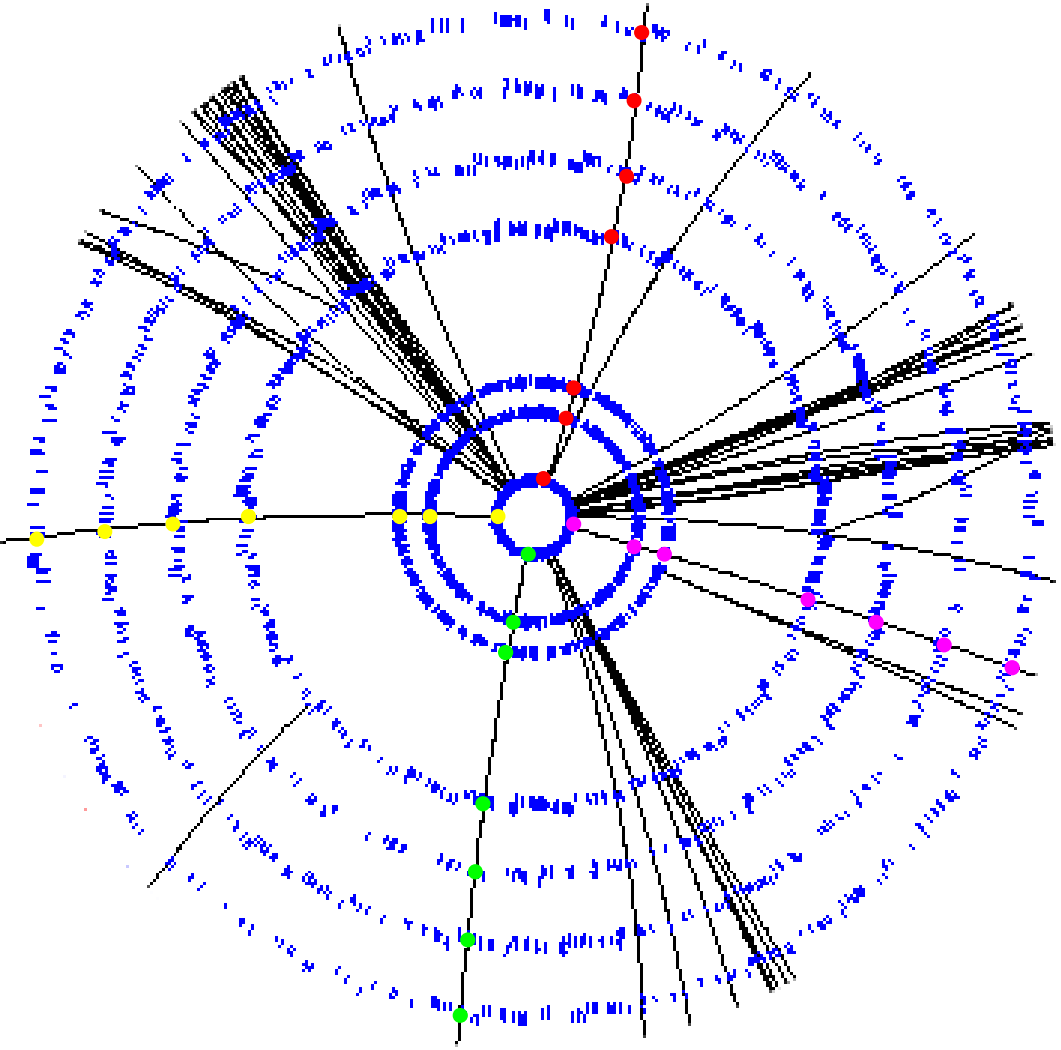
- The pattern matching uses a large bank of patterns and special hardware (associative memory - AM)
- The track's parameters are evaluated using a linear Principal Component Analysis algorithm (DSP MACC units in FPGA)

(j.nima.2003.11.078)

$$p_i = \sum C_{ij} \cdot x_j + q_i$$



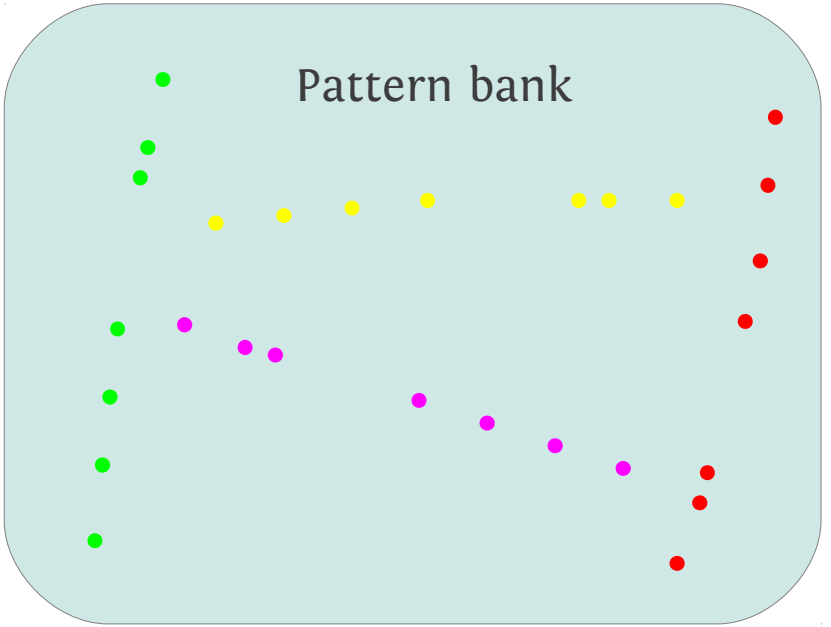
Fast pattern recognition with dedicated HW



Compare the event with a pre-computed collection of template tracks (pattern bank).

1) Find templates that matches hits of the event

2) Avoid computing combination of hits → time consuming, does not scale with occupancy

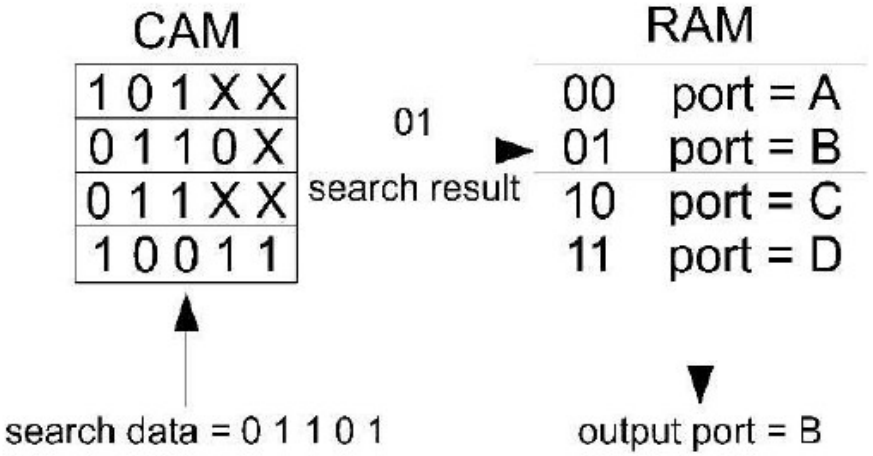


Fast pattern recognition with dedicated HW

Random Access Memory (RAM)

- Internal storage

- Input: address Output: data



Content Addressable Memory (CAM)

- Internal storage and built-in compare logic

- Input: search data Output: address

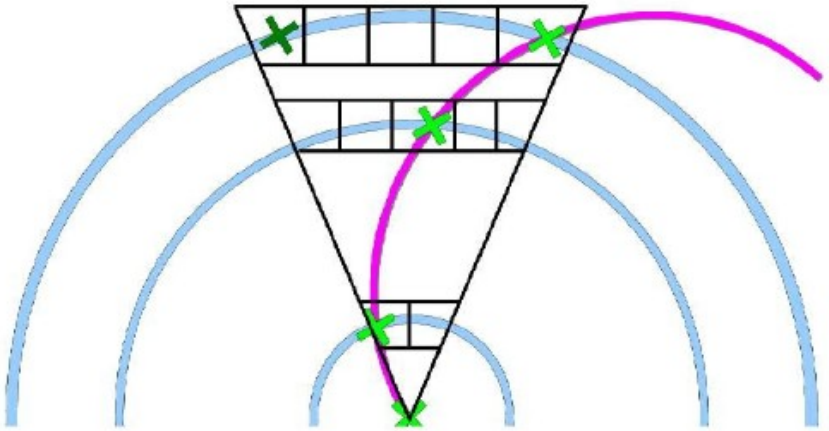
- Ternary logic: 0, 1, X “don’t care”

Commercially available

CAM are widely used in network hardware applications.

(also used inside processors: cache, mmu)

Fast pattern recognition with dedicated HW



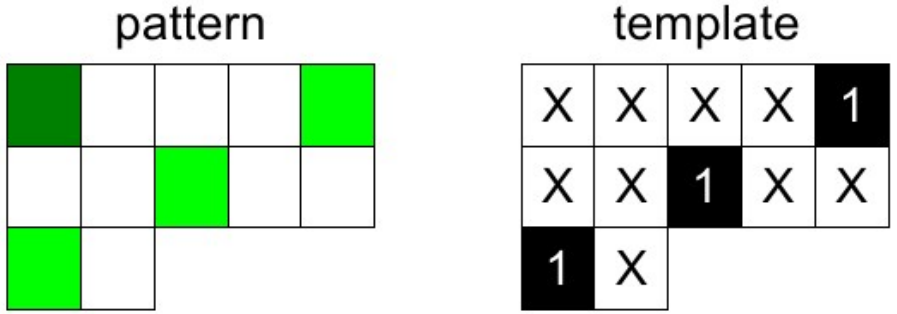
Data word: one bit per possible hit location, concatenated layers

Store template tracks using 1s and Xs

Pro: possible with commercially available CAM chips

Cons:
- Very long words in modern silicon detector applications
(ie. an ATLAS SCT module → 768 bits)

- Many templates for redundancy (taking into accout layer efficiency, store templates with missing layer)



Template: XXXX1XX1XX1X

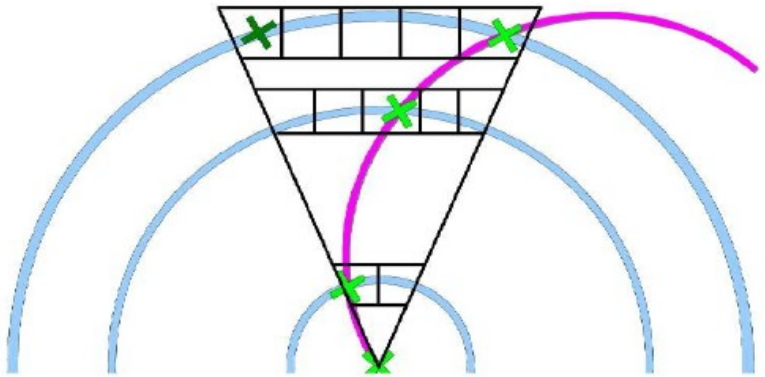
Search data: 100010010010

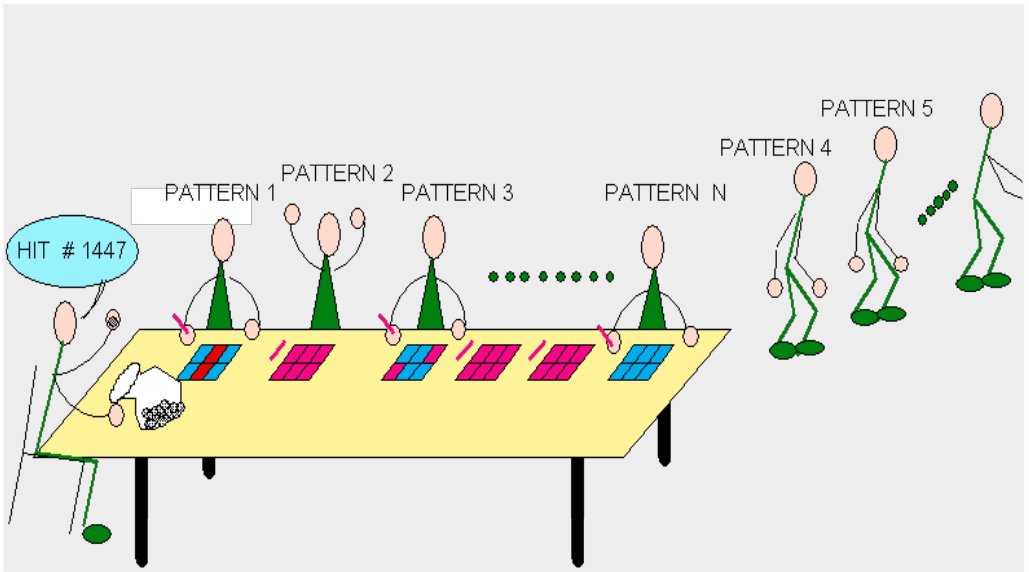
(from A. Schoening, S. Schmitt ideas for ATLAS L1 tracking trigger)

Fast pattern recognition with dedicated HW

SVT Associative Memory idea: build a custom kind of CAM more suitable for HEP applications.

Key features:
- store templates in encoded form, subdivided by detector layers
- take into account efficiency without template redundancy
- do not wait for complete event readout





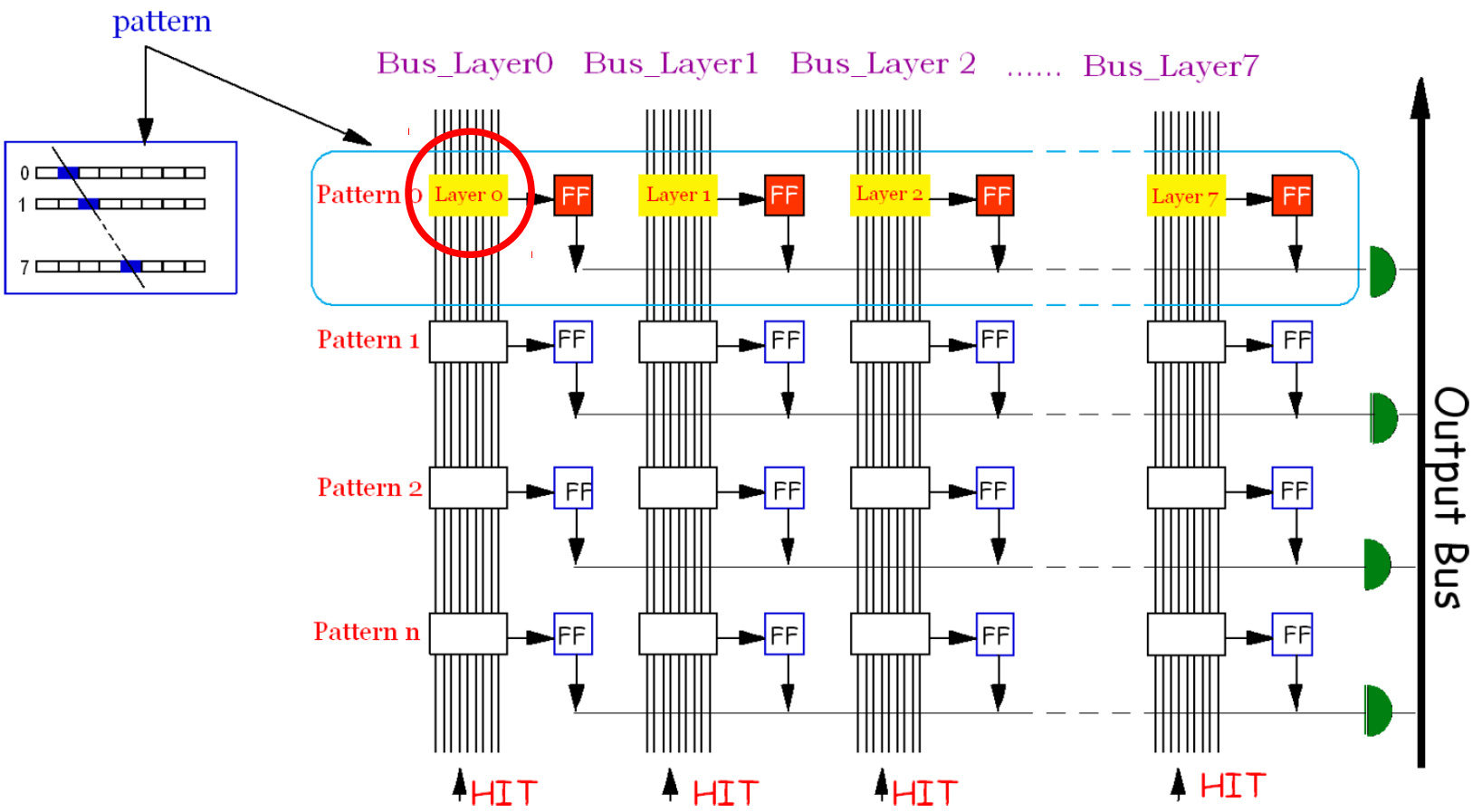
	Lay0	Lay1	Lay2
Template:	01	10	000
Search data:	01	10	000
			100

Advanced built-in compare logic:

"Smart" patterns, like bingo players

AMchip features

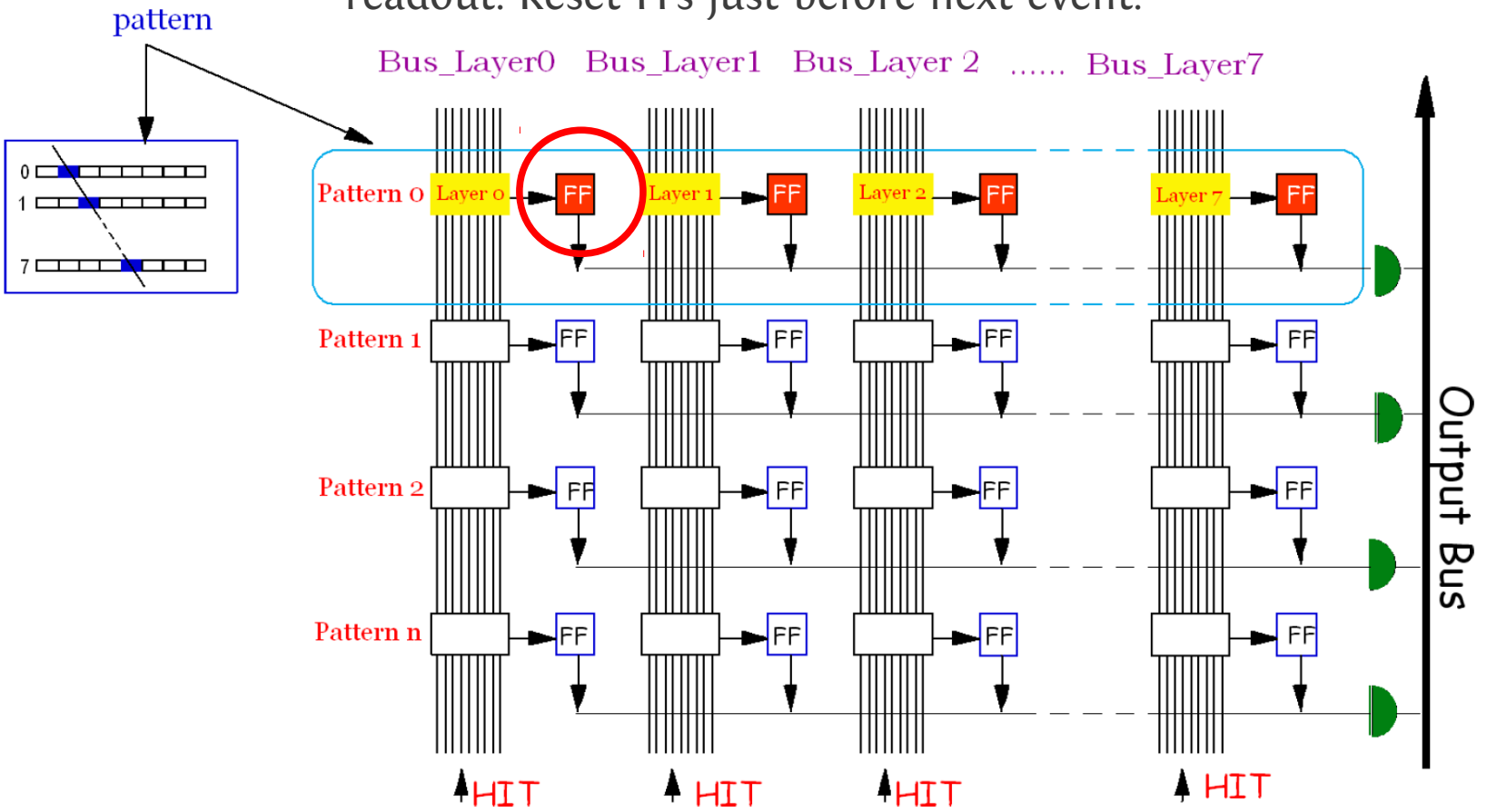
Patterns are divided per layer, a partial CAM for each one



Hit is stored in encoded form. Ie. 15-bit partial CAM can store hit position with 2^{15} resolution
Wider detector area covered with respect to standard CAM logic.

AMchip features

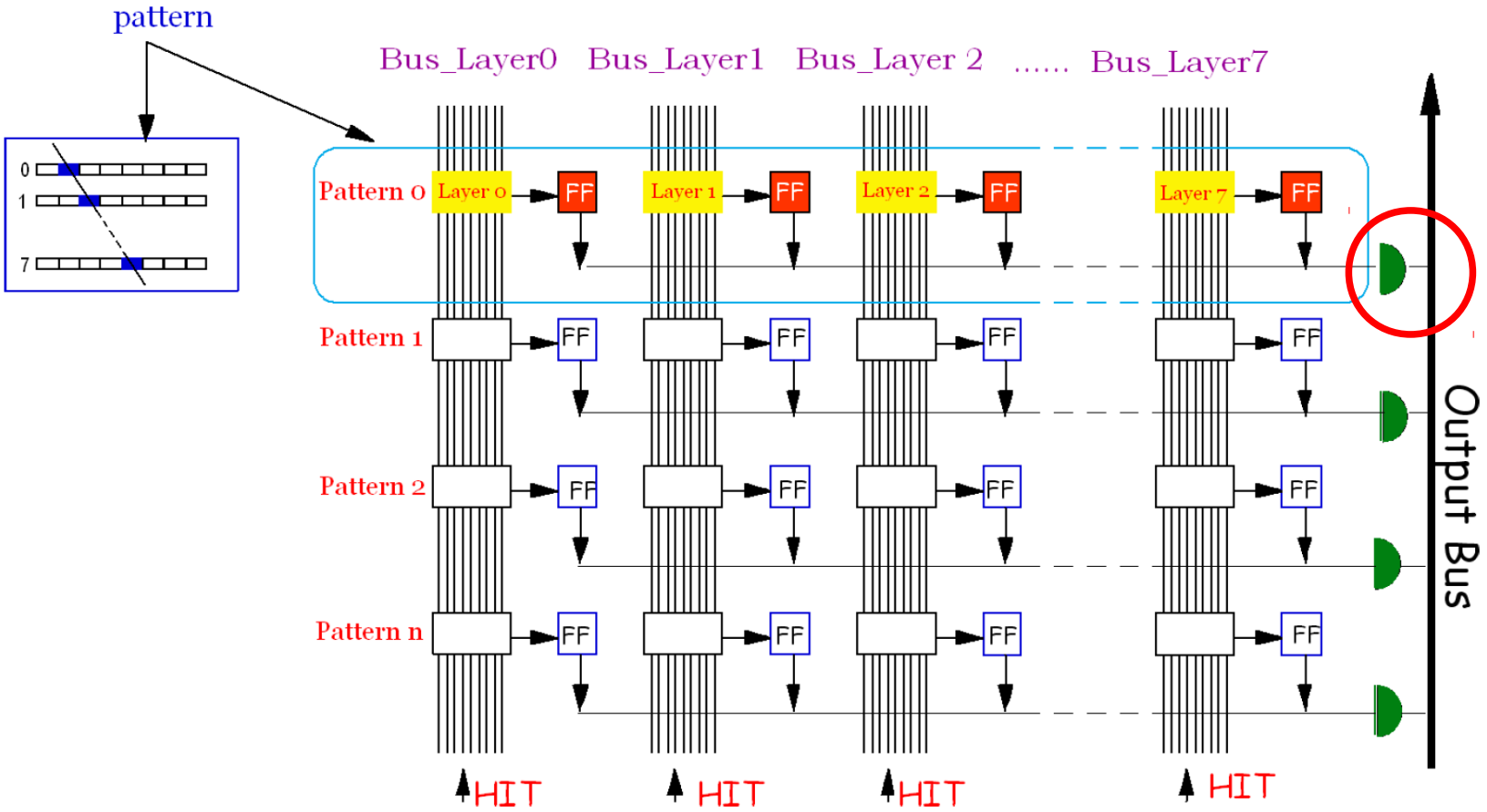
Partial CAM results is stored in a local memory (FF). Compare the event to the templates during detector readout. Reset FFs just before next event.



Separate input buses. No need to build the search data word from different layers, just send hits to the CAM when they are available.

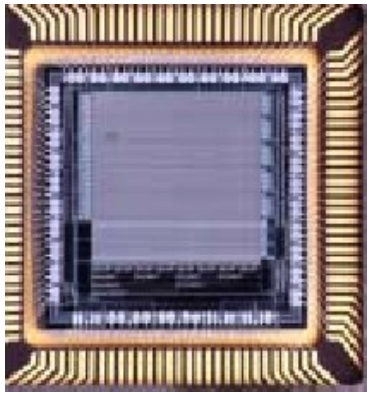
AMchip features

Majority logic: match even if not all layers are matched.
Output the bitmap of matched layer for post-processing.



Majority logic is programmable and dynamical. Lowering threshold during readout is possible for ordered output (full match, one missing, two missing, ...)

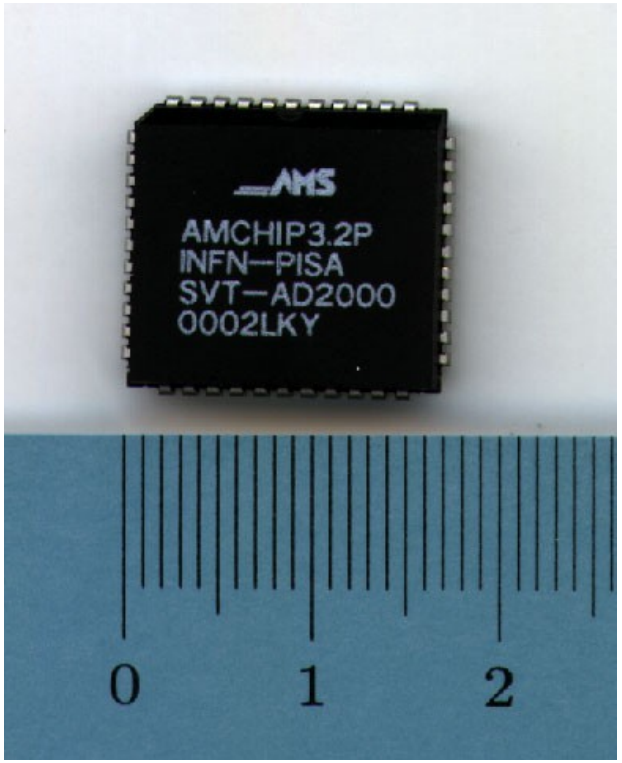
AMchip history



Full-custom VLSI chip
700 nm technology

128 patterns, 6 x 12 bit

Working @ 40 MHz



Development started in the 90s

~5000 chips produced for SVT in 2000

Used by SVT from 2001 to 2006

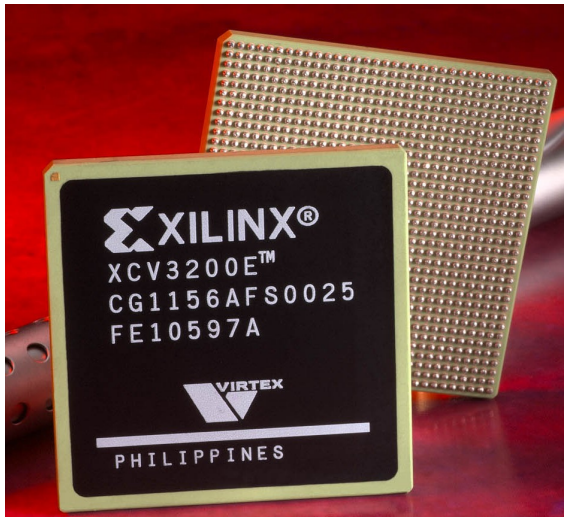
The development and production proved to be quite hard.
many iterations of prototypes, debug and simulations were needed.

What to do for the SVT upgrade?

First article:

S.R. Amendolia, S. Galeotti, F. Morsani, D. Passuello, L. Ristori, N. Turini **The AMchip: a VLSI associative memory for track finding** Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, Volume 315, Issues 1-3, 1 May 1992, Pages 446-448

AMchip history



In the late 90s a functionally compatible version of the full-custom AMchip was developed using programmable devices.

Using a Xilinx FPGA in 350 nm technology it was achieved the same pattern density (128 patt/chip) as the full-custom version.

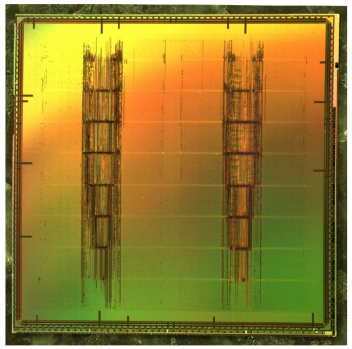
The FPGA version was flexible and a lot faster to develop and debug …

… but FPGA industry evolved in unpredicted ways (changes in packages, CLB technology, …) and future versions couldn't achieve required pattern density for the SVT upgrade.

AM FPGA ideas & firmware has been used in low density applications that required flexibility for the SVT upgrade (RoadWarrior)

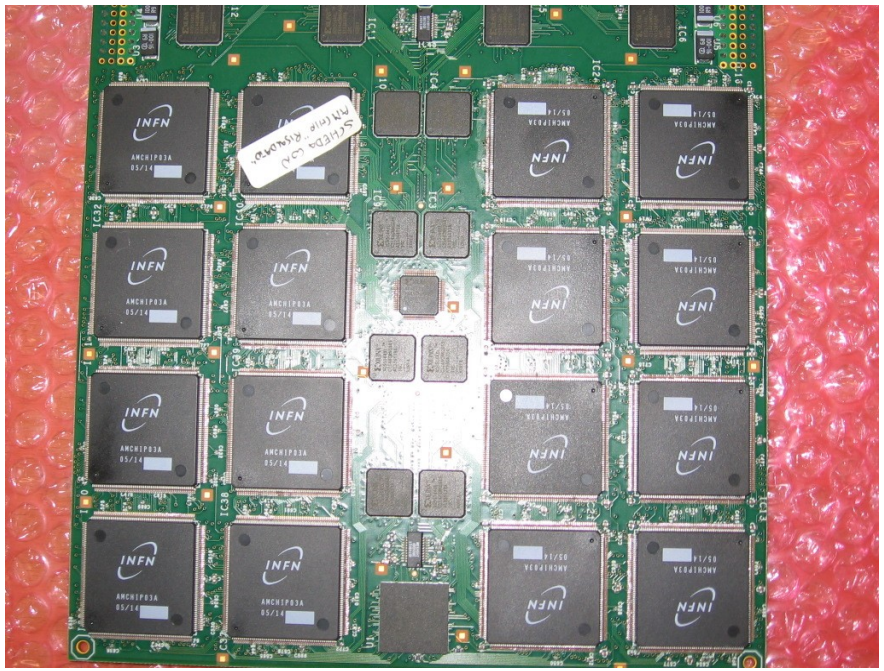
A. Bardi, S. Belforte, A. Cerri, M. Dell'Orso, S. Donati, S. Galeotti, P. Giannetti, , A. Leger, E. Meschi, F. Morsani, D. Passuello, G. Punzi, L. Ristori, T. Speer, F. Spinella, X. Wu **A programmable associative memory for track finding** Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Volume 413, Issues 2-3, 21 August 1998, Pages 367-373

AMchip history



SVT Upgrade chip AMchip03 (2006): **Standard Cell 180 nm**
5000 pattern/chip for 6-layer patterns,
2500 pattern/chip for 12-layer patterns
40 MHz clock

"A VLSI Processor for Fast Track Finding Based on Content Addressable Memories",
IEEE Transactions on Nuclear Science, Volume 53, Issue 4, Part 2, Aug. **2006** Page(s):2428 - 2433



16x AMchip03 on a LAMB mezzanine

Standard Cell approach was the right compromise between development speed and reliability and pattern density.

2 years from project approval to Installed version

64 chips * 24 boards running in the upgraded SVT at CDF since 2006 until **today** (literally)

AMCHIP04

- Major improvements:
  - ~80k patt/chip with 1.2x1.2 cm^2 area
  - Low power consumption
    (128 chips/VME board @ 100 MHz)
  - 8 input buses
- New features:
  - Ternary logic for variable resolution
  - Pattern output while next event is loaded
- Technology TSMC 65nm Low Power
- First prototype: 8k patterns, 13x13 mm^2

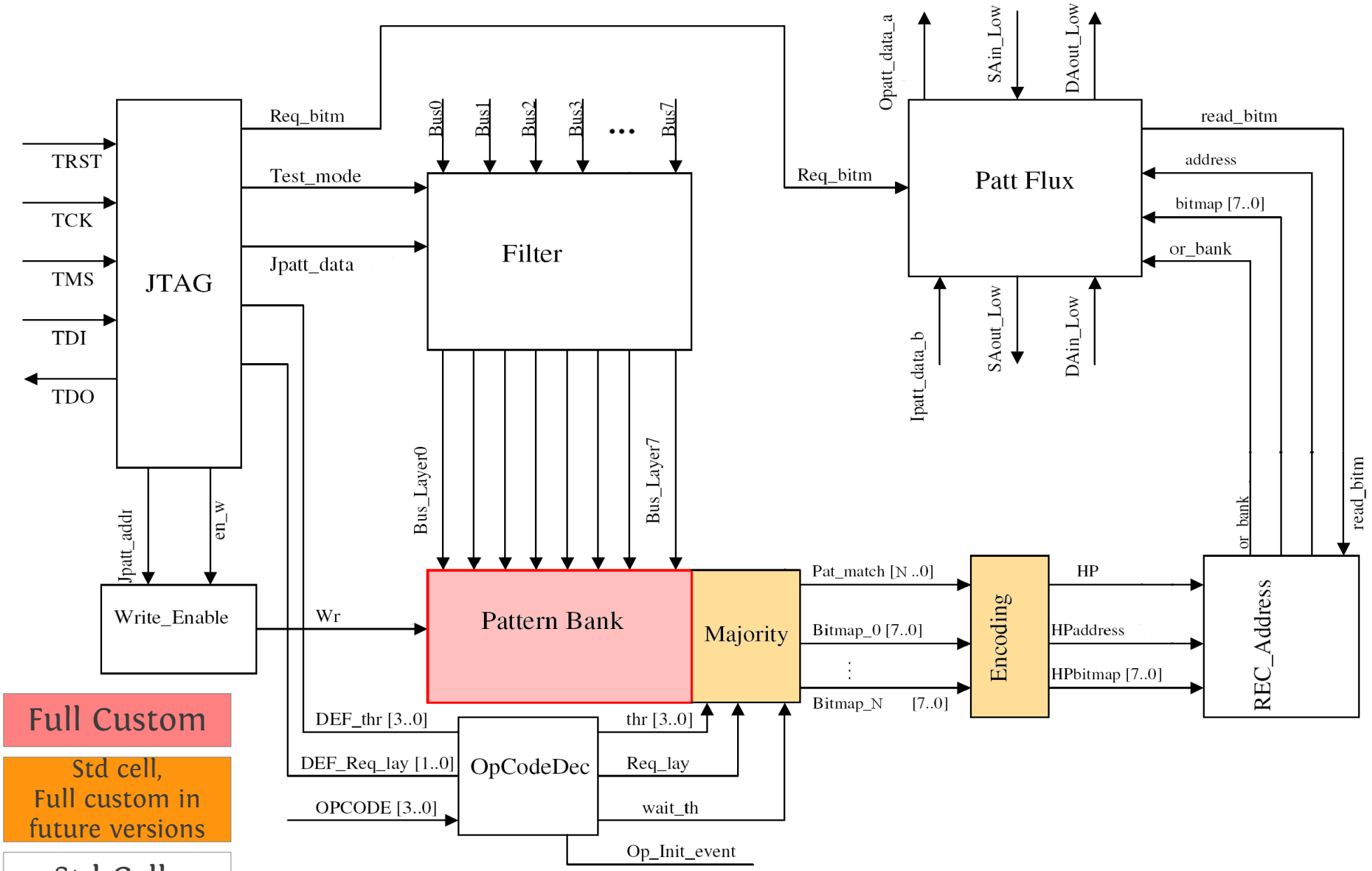
CAM layer full custom

Control logic in Standard cells

Main dev:
INFN Pisa
INFN Milano
INFN Frascati

Minor contr.
Fermilab
Heidelberg

AMCHIP04

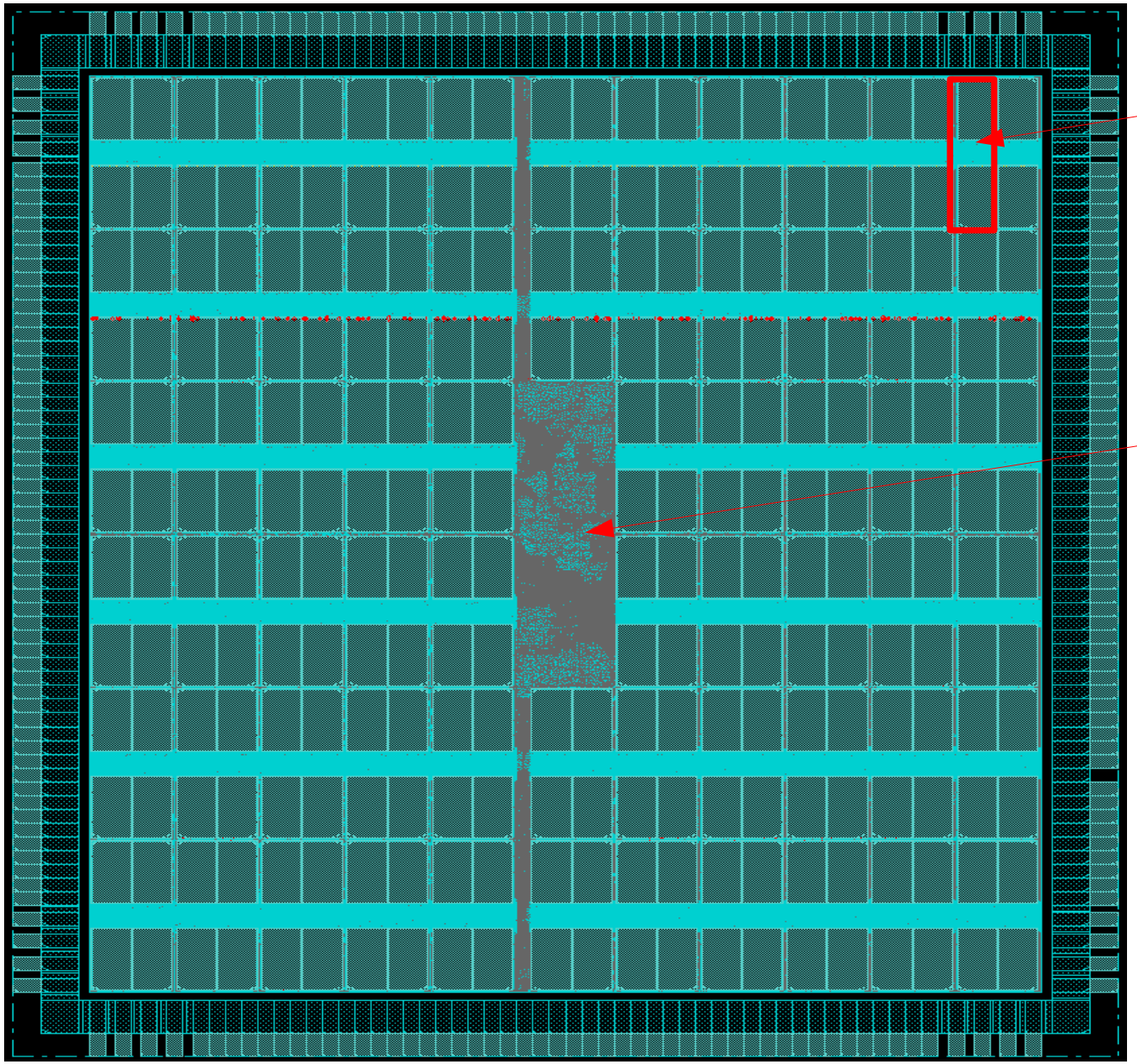


Full Custom

Std cell, Full custom in future versions

Std Cells

AMchip04: floorplan (8k)

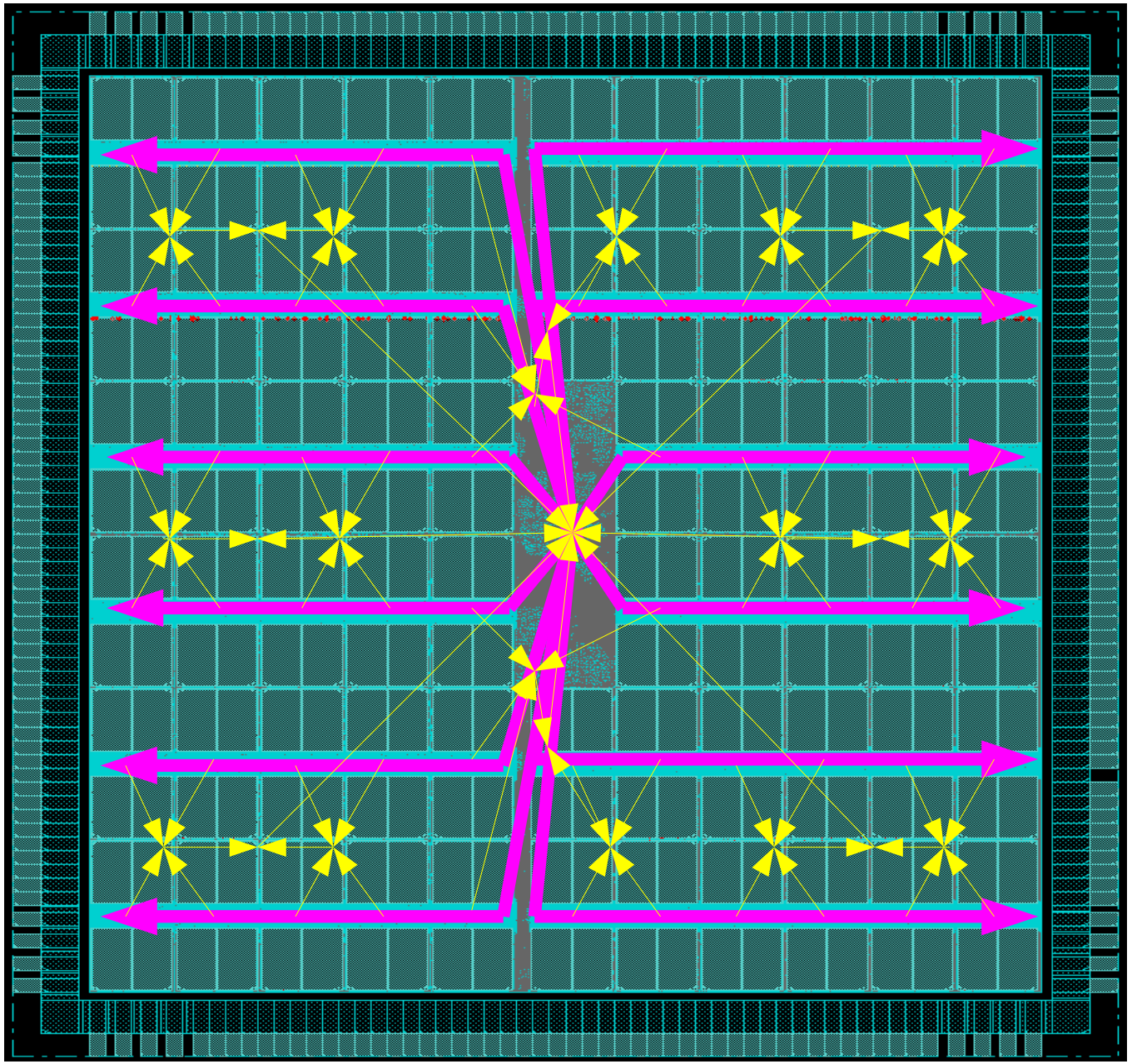


64 patterns CAM layers block + Local majority logic and readout tree + Write enable row decoders

All remaining control Logic (JTAG, bus filters, pattern flux control, opcode control)

The 8k design & floorplan have been conceived to minimize the work for the final 80k version.

AMchip04: floorplan (8k)



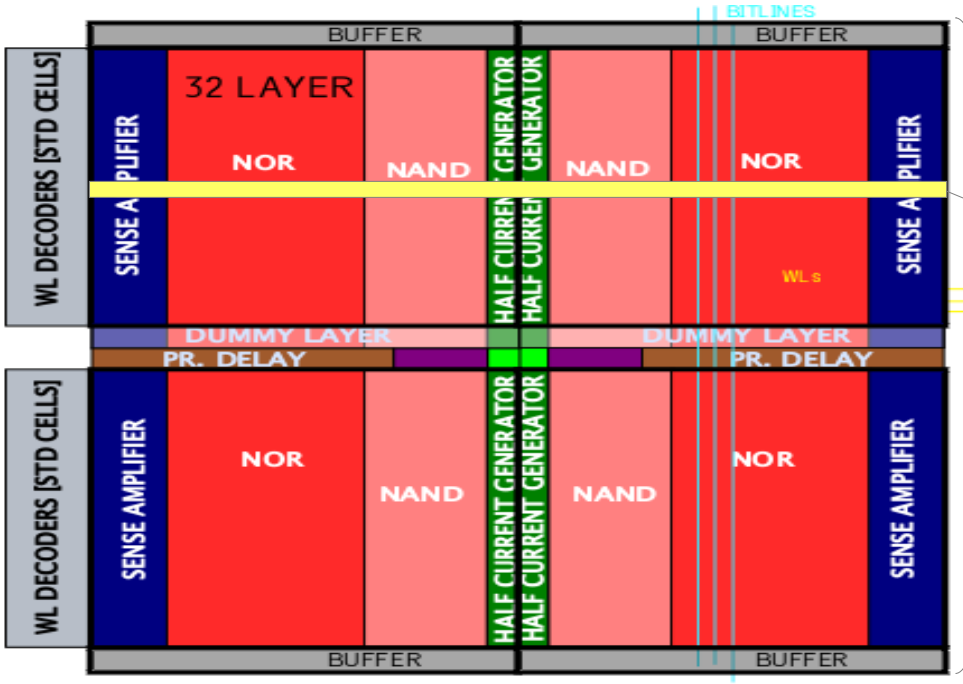
Hit buses distribution

From the center control logic to 12 rows of AM memory.
Designed to keep net delays uniform and under control without burden too much the routing (18 x 8 x 2 bits bus)

Matched patterns tree

Blocks are grouped in 128 → 256 → 512 → 1024 Patterns and then routed To the center logic.
This layout is natural with our readout tree structure and not a problem for routing (at each step: $\log_2$(patterns)+2 bits)

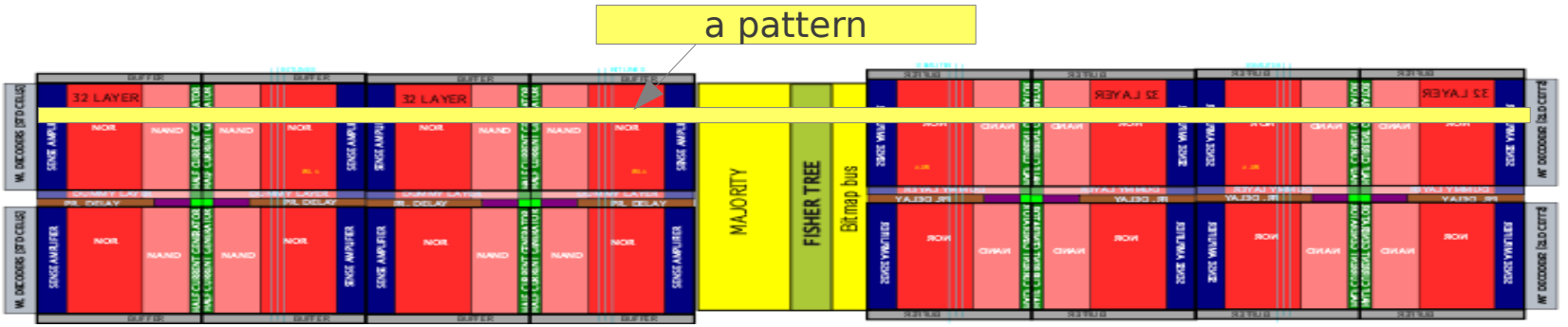
AMchip04: 64 patterns block



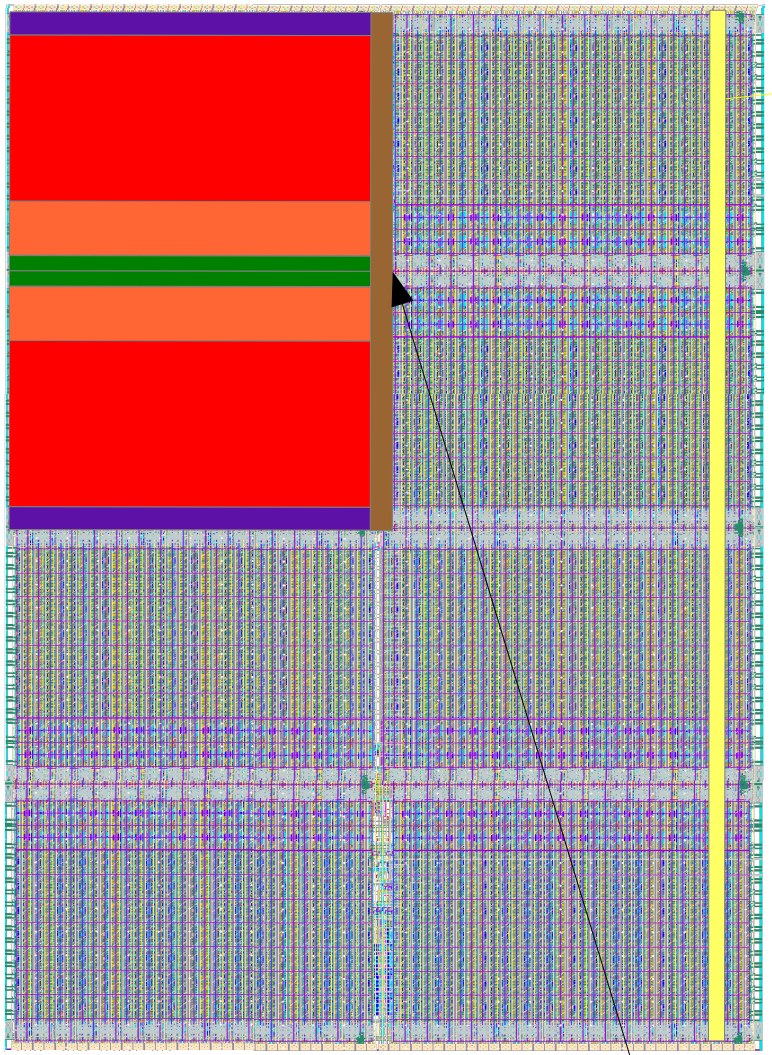
VBIAS REFERENCE
RESISTANCE

Two Layers = 1 / 4 pattern

64 Layers + 1 Dummy layer in the middle



AMchip04: 64 patterns full-custom layout



4 Layers = 1/2 pattern

Full custom Layout of 64 x 4 CAM layers (half pattern):w=~226 µm X h=~123 µm

without including
- major logic
- readout logic
- control logic

Six metal layers are used to route signals, power supply and ground.

Bit lines are routed horizzontally while control lines and memory output are routed vertically

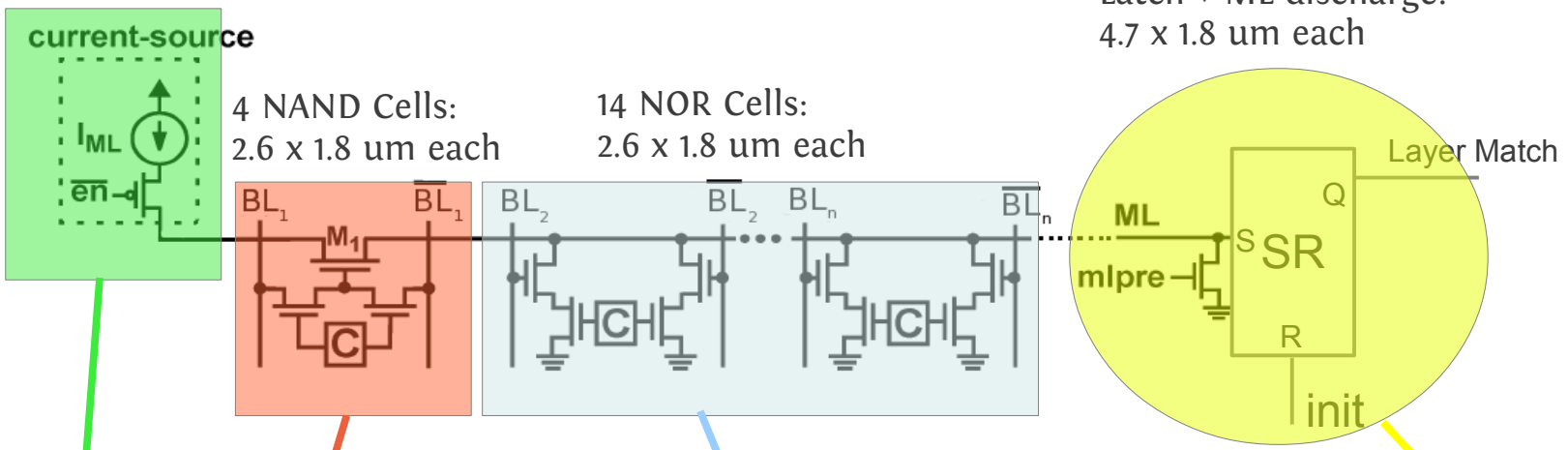
AMCHIP04: POWER SAVING

To save power we have used two different match line driving scheme:

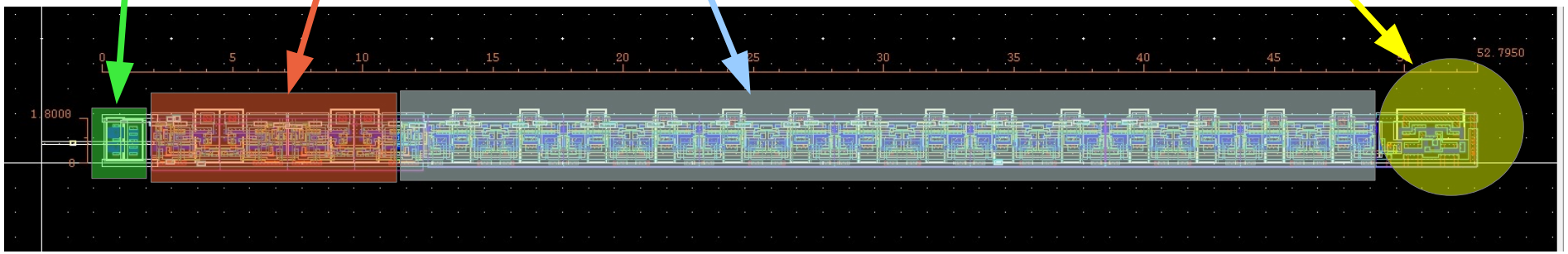
- *Current race scheme*
- *Selective precharge scheme*

Current source:
3.7 x 1.8 um each

Latch + ML discharge:
4.7 x 1.8 um each



Full Layer layout : w=~53µm X h=1.8 µm



AMchip04: power saving

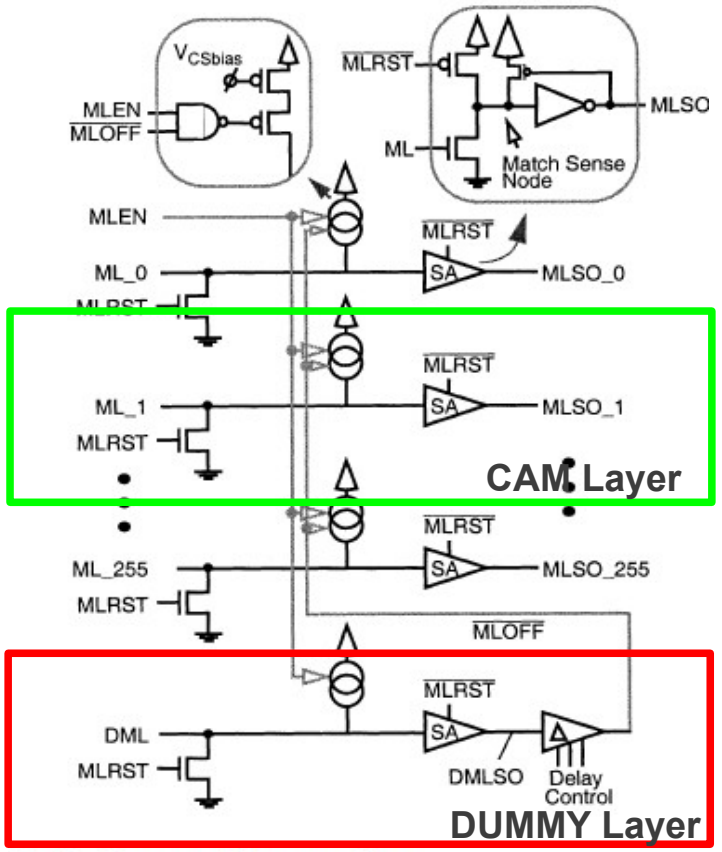


Fig. 5. Current-race ML sensing scheme.

Scheme from: *"A ternary content-addressable memory (TCAM) based on 4T static storage and including a Current-Race sensing scheme"*, Ali Sheikholeslamiet Al. IEEE Journal of Solid-State Circuits, Vol. 38, NO. 1, January 2003

Current race for a layer CAMs column:

1) When a new hit is to be compared an enable signal (MLEN) is asserted for half a clock cycle

2) The column enable is the AND of MLEN and the negated output ($\overline{\text{MLOFF}}$) of an always matching layer (dummy layer)

3) The dummy layer "measures" the local matching time (+ a programmable delay) optimizing the time the current generator is on (power consuming)

AMchip04 has one dummy layer every **64 patterns** in a layer CAMs column

The dummy layer is placed in the middle to optimize $\overline{\text{MLOFF}}$ distribution to the column

A four steps **programmable delay** is available to optimize $\overline{\text{MLOFF}}$ timing and measure current race effect on power consumption in the first prototype

AMchip04: power saving

Selective precharge:

1) The first bits of the pattern word are implemented with NAND type cells

2) If a NAND type cell doesn't match the match-line is not charged, saving power

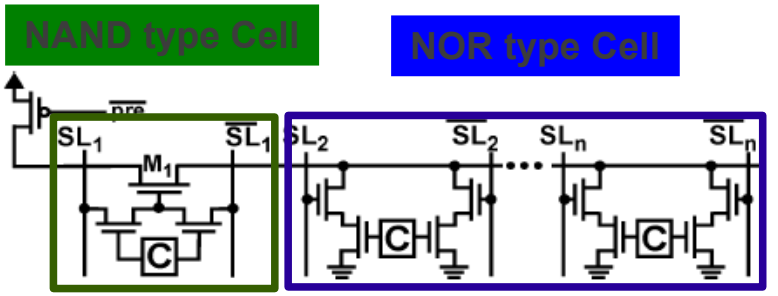


Fig. 16. Sample implementation of the selective-precharge matchline technique [43]. The first cell on the matchline is a NAND cell, while the other cells are NOR cells. Precharge occurs only in the case where there is a match in the first cell. If there is no match in the first cell, the precharge transistor is disconnected from the matchline, thus saving power.

Scheme from: *"Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey"*, Kostas Pagiamtzis and Ali Sheikholeslami IEEE Journal of Solid-State Circuits, Vol. 41, NO. 3, March 2006

NAND cells are slow, a good balance between NAND and NOR cells is needed to have both low power consumption and operational speed

AMchip04 layer CAMs are made by 4 NAND cells and 14 NOR cells

AMchip04: timing performances

Simulation done in nominal conditions:
Transistors models → TT
VDD → 1.2V
Temperature→ 27 °C

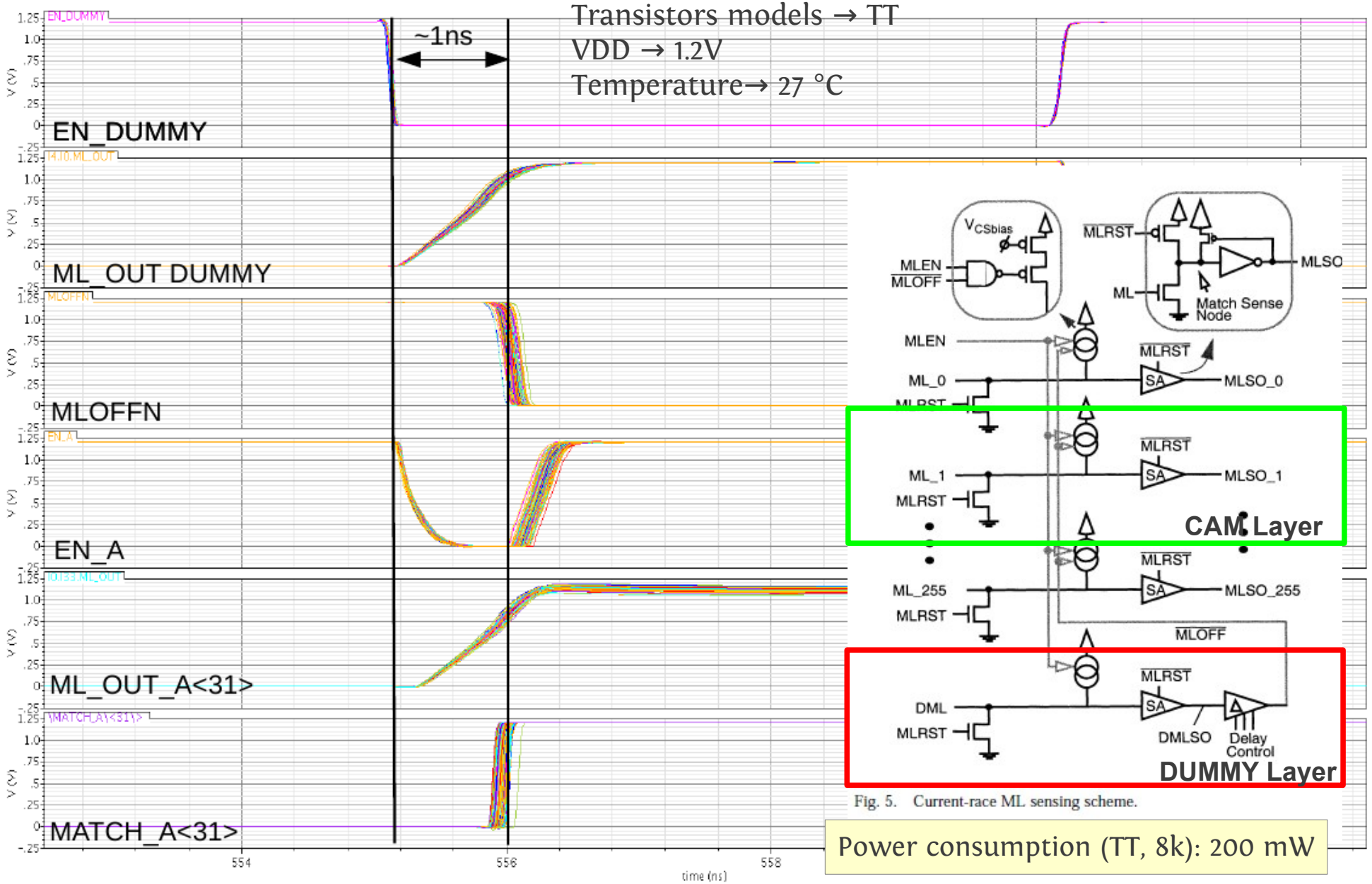
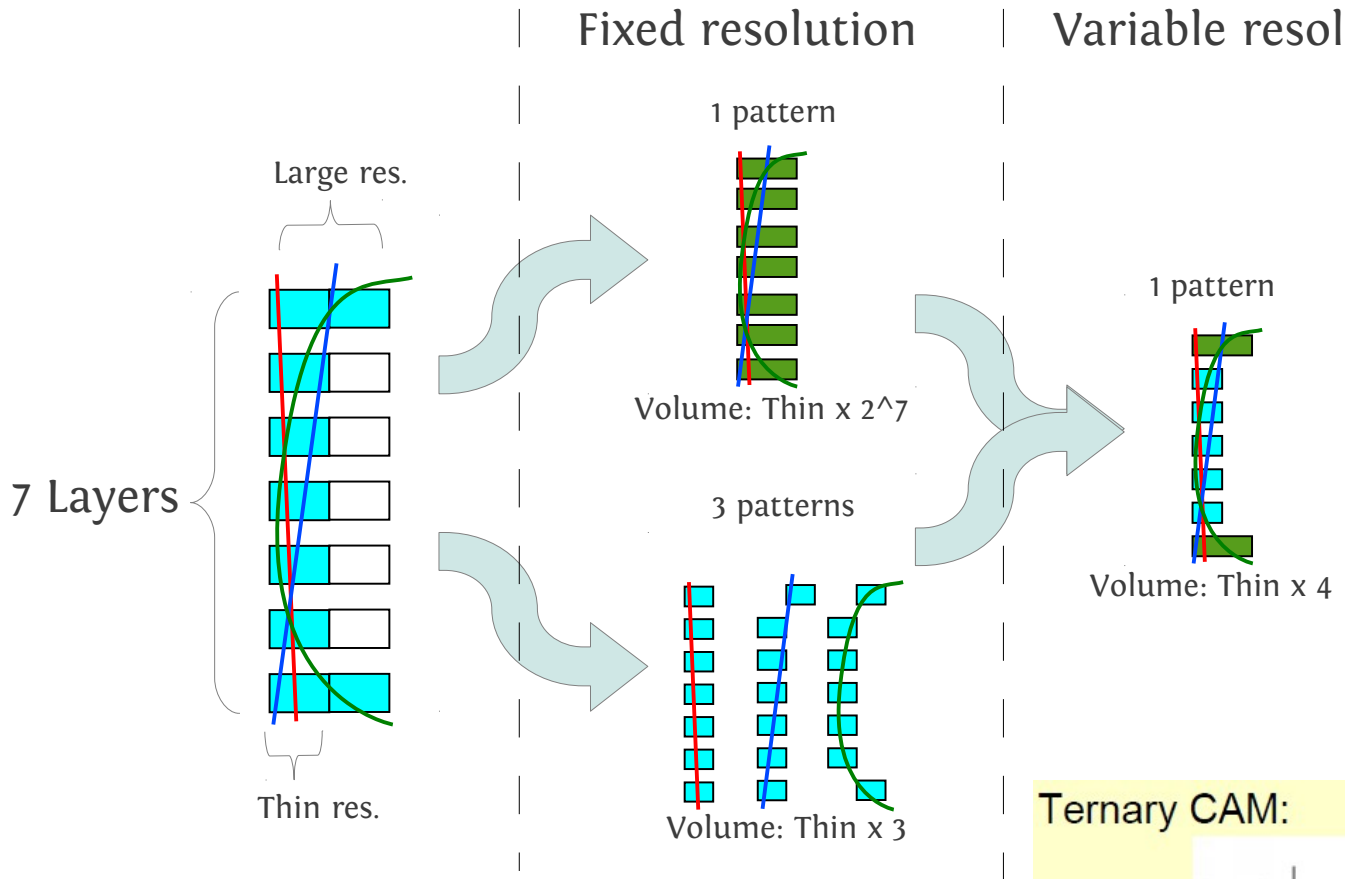


Fig. 5. Current-race ML sensing scheme.

Power consumption (TT, 8k): 200 mW

AMCHIP04: VARIABLE RESOLUTION

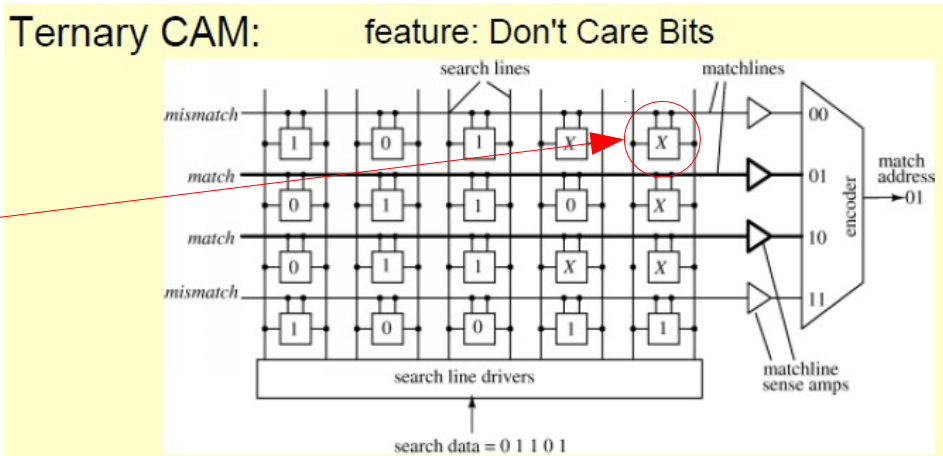


Good rejection and occupy only one pattern location.

Per-pattern choice of optimal resolution.

We can use **don't care** on the least significant bit when we want to match the **pattern layer @ Large resolution** or use all the bits to match it **@ Thin resolution**

Coincidence window is programmable layer by layer and pattern by pattern



AMchip04: ternary cells

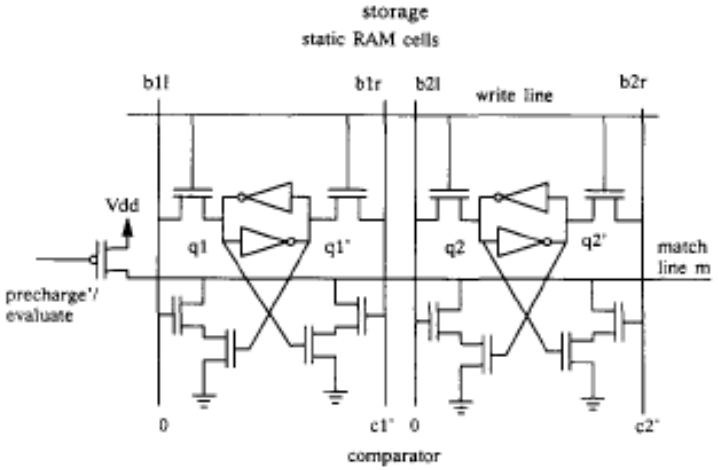


Fig. 8. Two adjacent static binary CAM cells.

Images from: *"Encoding Don't Cares in Static and Dynamic Content-Addressable Memories"*, Sergio R. Ramirez-Chavez, IEEE Transactions on circuits and system-II: Analog and Digital Signal Processing, Vol. 39 NO. 8, August 1992

AMchip04 has 14 NOR cells
Up to 7 ternary logic bits

No difference in CAM layer:
the number of ternary bits is configurable via JTAG

Two standard NOR cells implement a ternary logic cell out-of-the-box.

Special drive of BL lines is needed.

storage scheme
stored values

	q1q2
0	01
1	10
*	00

(a)

retrieval scheme
presented values

presented ternary value	encoded value in the bit lines of two binary static CAMS.					binary CAM equivalent operation
	c1c2	b1l	b1r	b2l	b2r	l r
0	01	0	1	0	0	0 M*
1	10	0	0	0	1	M 0
*	11	0	0	0	0	M M

*M is the masking of a bit operation common in commercial binary CAMS.

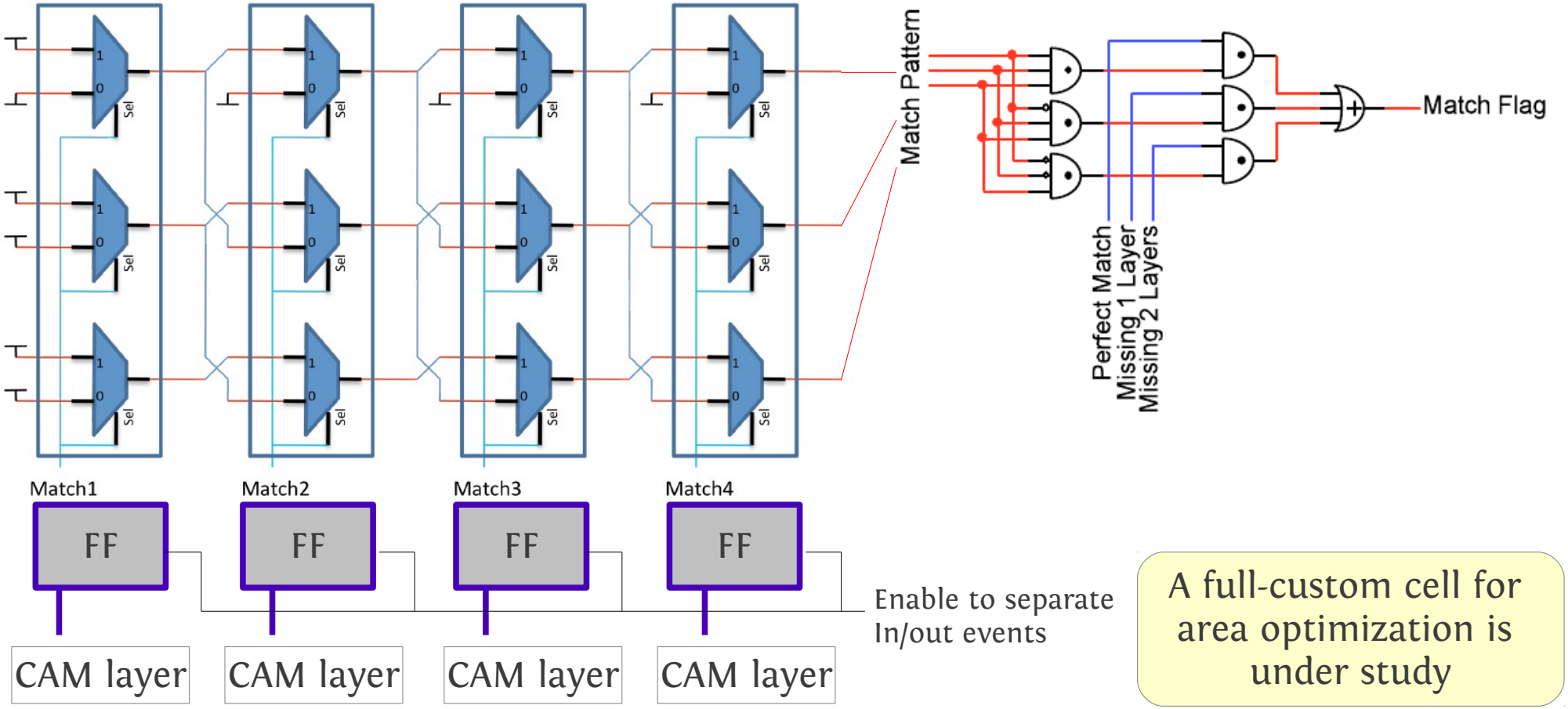
(b)

Fig. 9. Encoding and retrieval schemes for don't-care in two static binary CAM's cells with masking capability. (a) Encoding scheme. (b) Retrieval scheme.

AMCHIP04: MAJORITY LOGIC

Pattern match logic is made by identical logic for each layer:
Receives in input 3 bits, if not matching shift down the output.

At the end of the chain the 3 bits are compared with the majority requirements: perfect match, 1 or 2 missing



A full-custom cell for area optimization is under study

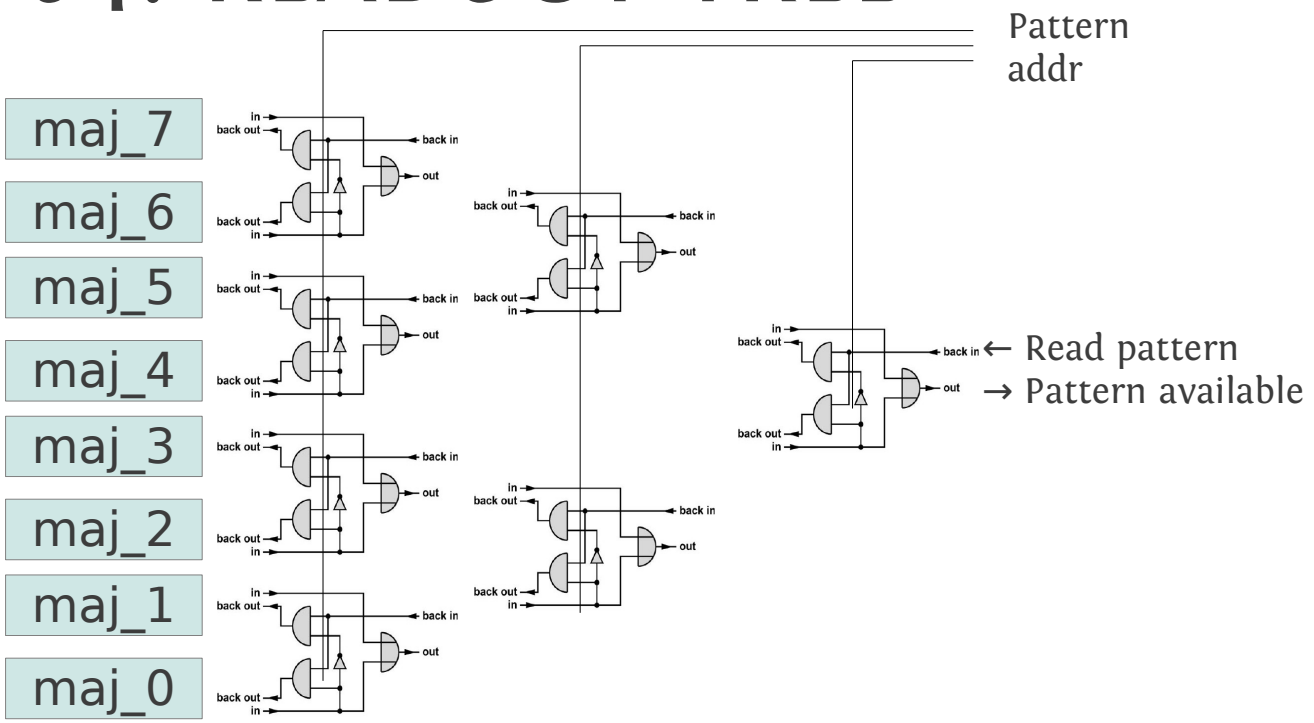
Layer matches from CAM layers are stored in a FF just before resetting the CAM layers.
We can load an event in the CAM layers while we are reading the patterns found in the previous event.

AMCHIP04: READOUT TREE

The readout tree is made by identical cells that selects priority between two branches of the tree.

Address is built by an OR of all lesser-priority selection lines at each tree level.

This design is simple and modular, it is easy to merge pattern sub-banks readout.



Simple OR-based logic
For priority encoding and readout

A full-custom cell for area optimization is under study

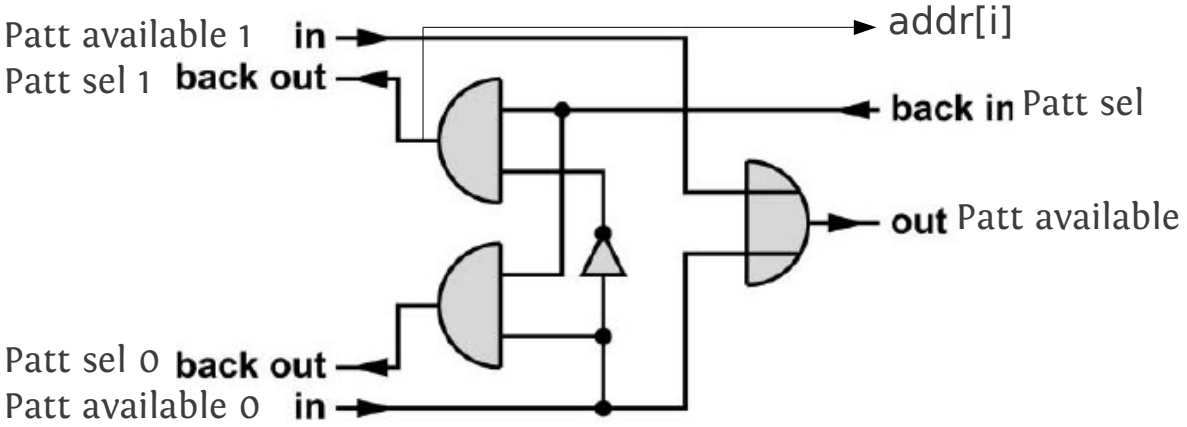
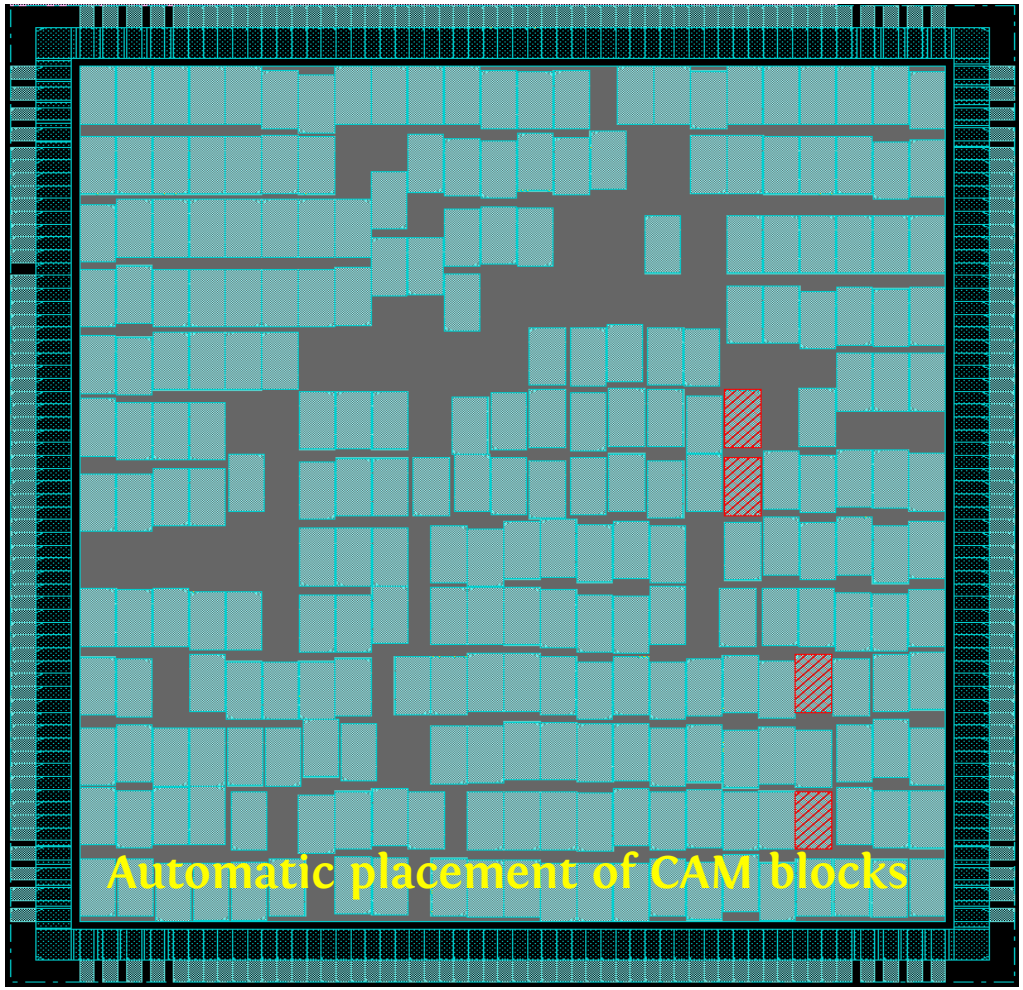


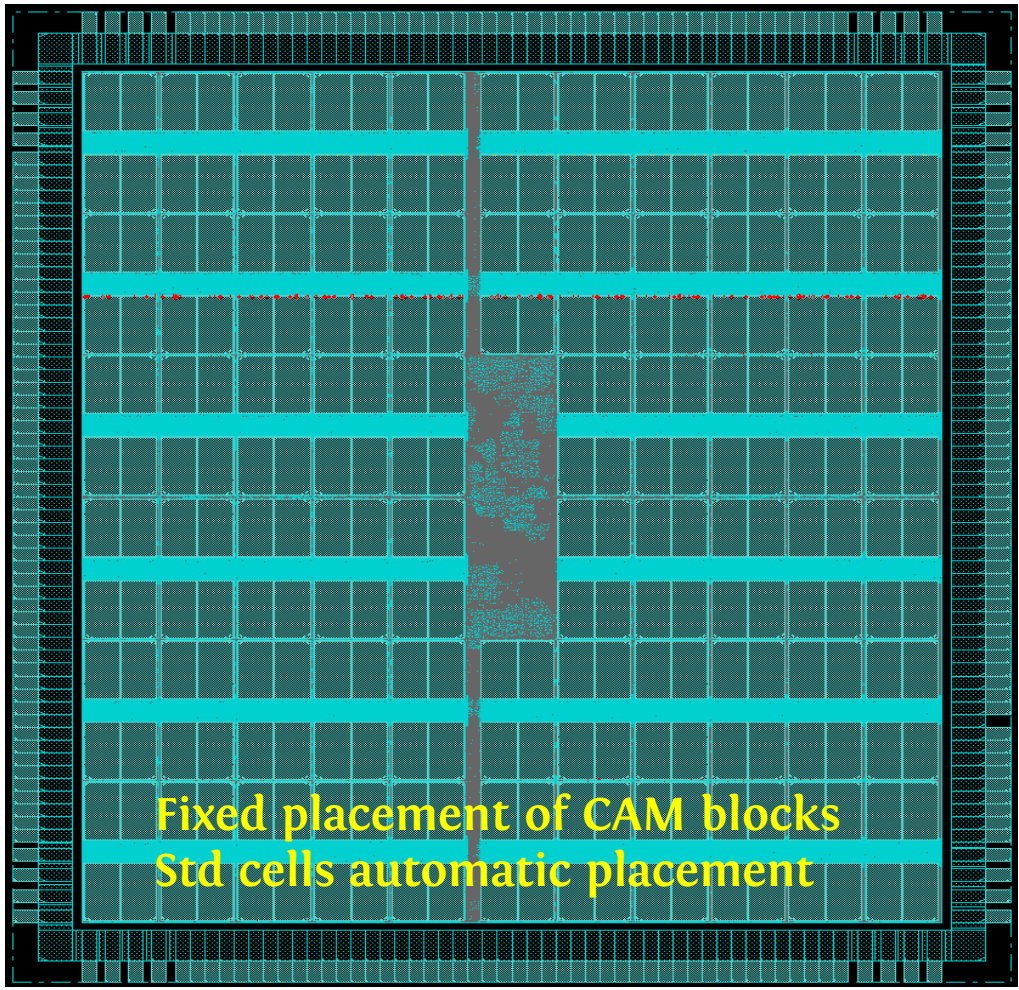
Image & cell scheme from:
P. Fischer, **First implementation of the MEPHISTO binary readout architecture for strip detectors**, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Volume 461, Issues 1-3, 1 April 2001, Pages 499-504

AMchip04: Back End

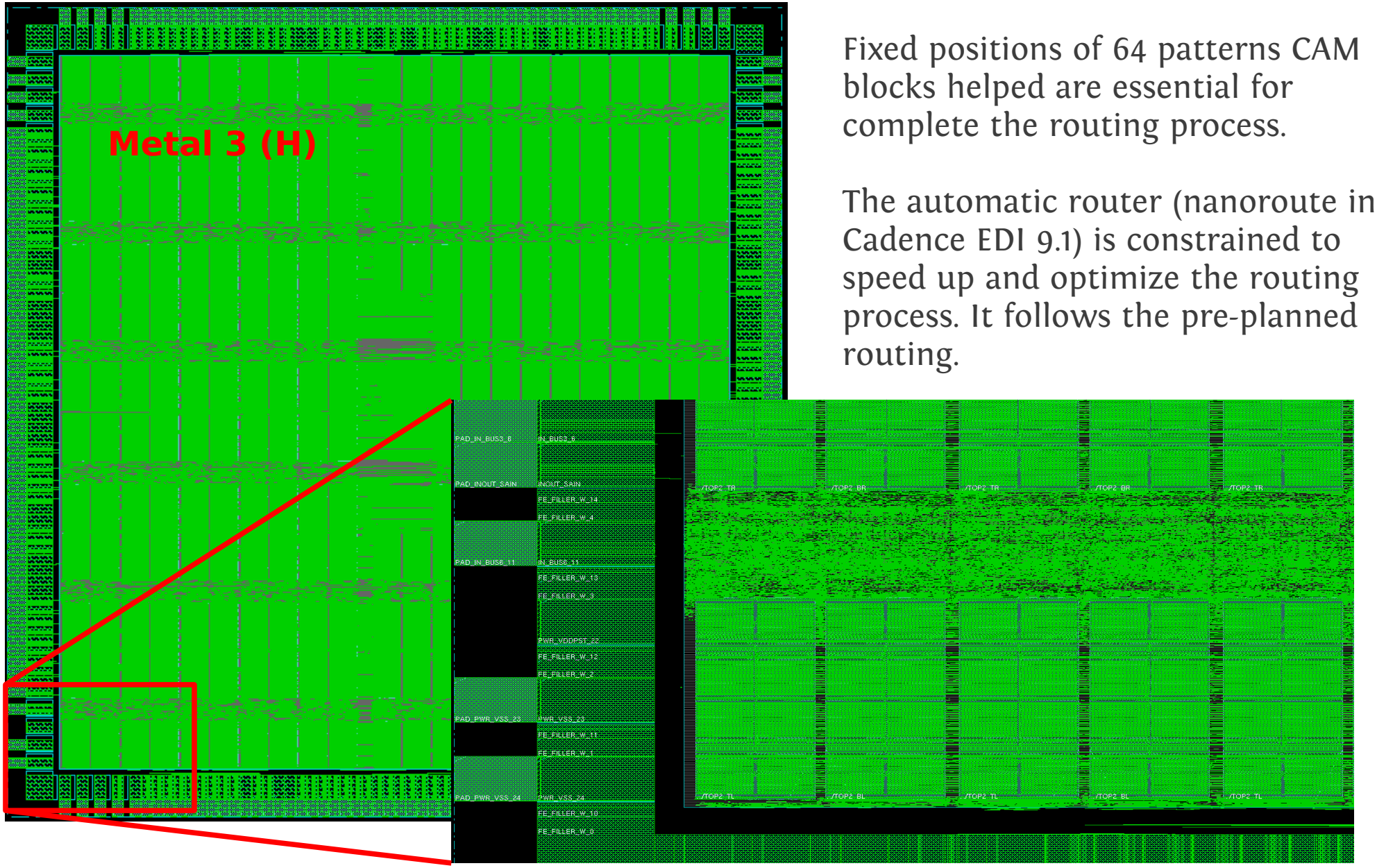
The AMchip04 is a challenging design from the place&route point of view.

The 64 patterns half-CAM block is placed to follow the desired general routing idea. Standard cells are placed by the automatic placer of Cadence EDI 9.1





AMchip04: back end



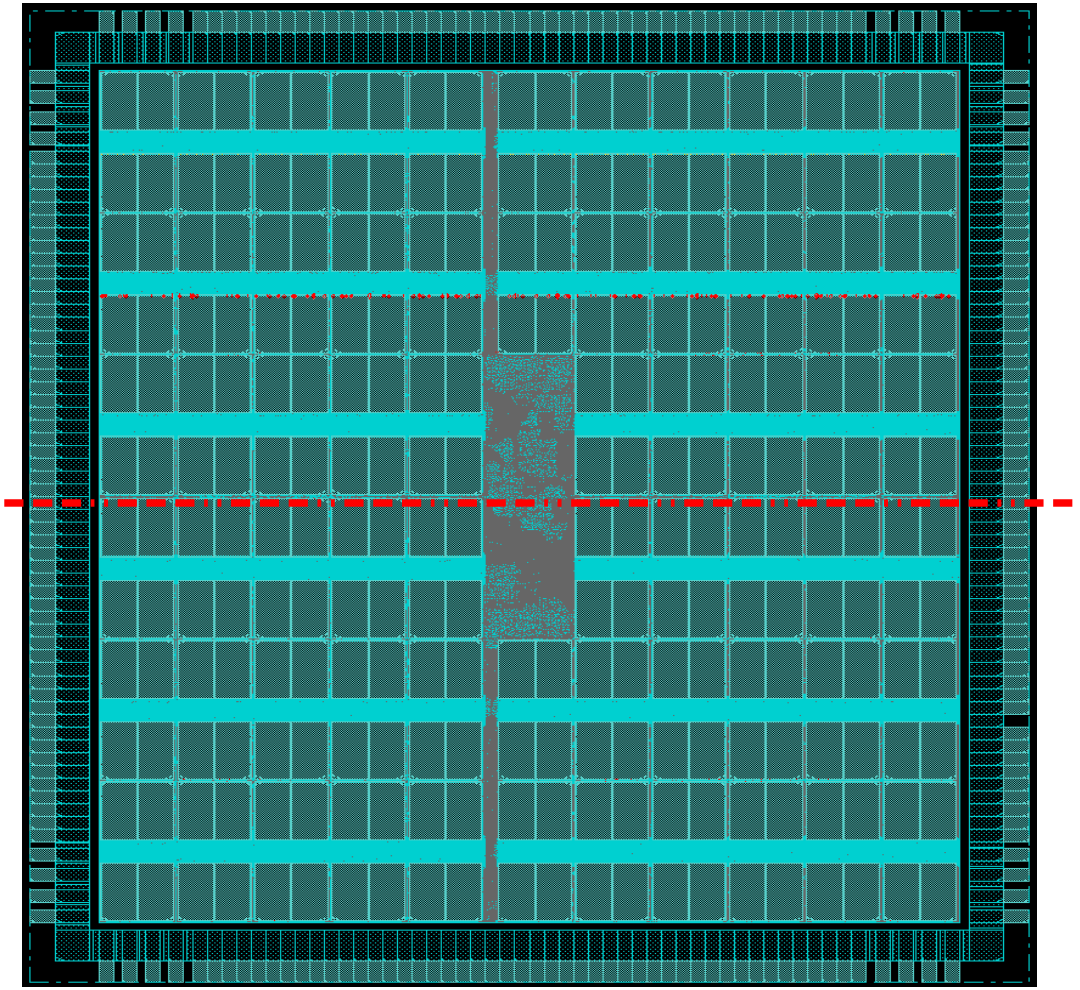
Fixed positions of 64 patterns CAM blocks helped are essential for complete the routing process.

The automatic router (nanoroute in Cadence EDI 9.1) is constrained to speed up and optimize the routing process. It follows the pre-planned routing.

AMchip04 near future plans

- New features, pattern density and simulated power consumption are within the stated goals of the chip
- FrontEnd & BackEnd almost finished
- First prototype of the new FTK AMchip04 will be submitted to the foundry in autumn
- A second prototype is foreseen to test new technology (full-custom majority and readout tree) and to correct possible bugs/problems

Future evolutions in 2.5D



AMchip04 has been designed to be horizontally symmetric.

- In/out buses for pattern output pipeline can change direction

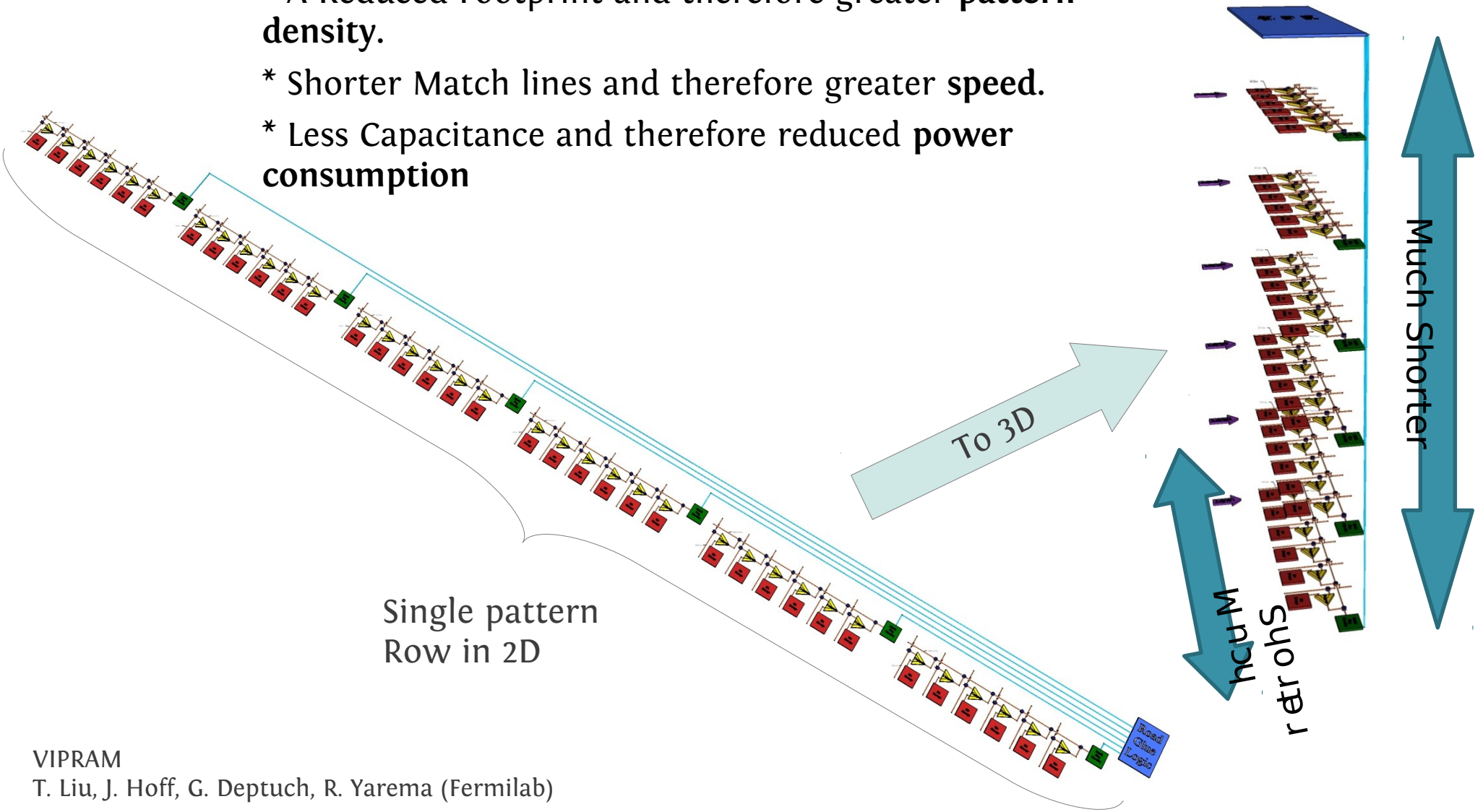
- Buses are swapped internally to maintain consistency

Symmetry helps in designing and routing mezzanines for 2D chips, but also enables vertical stacking:

2.5D

Future evolutions in 3D

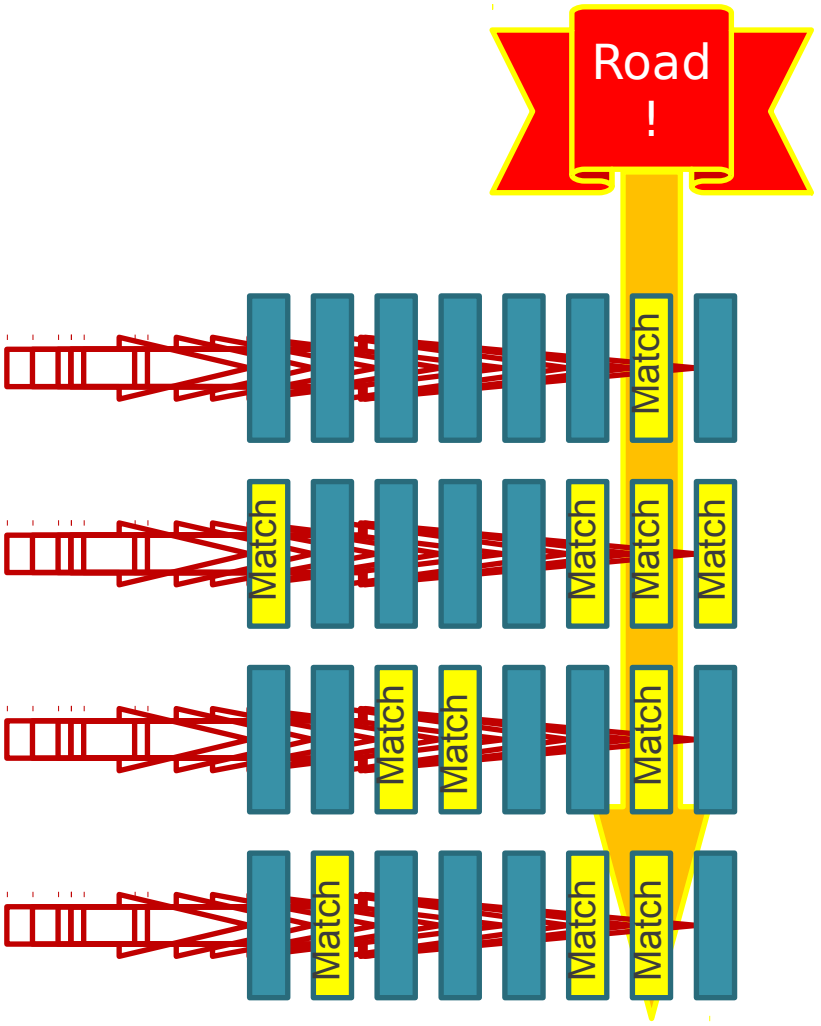
* A Reduced Footprint and therefore greater **pattern density.**
* Shorter Match lines and therefore greater **speed.**
* Less Capacitance and therefore reduced **power consumption**

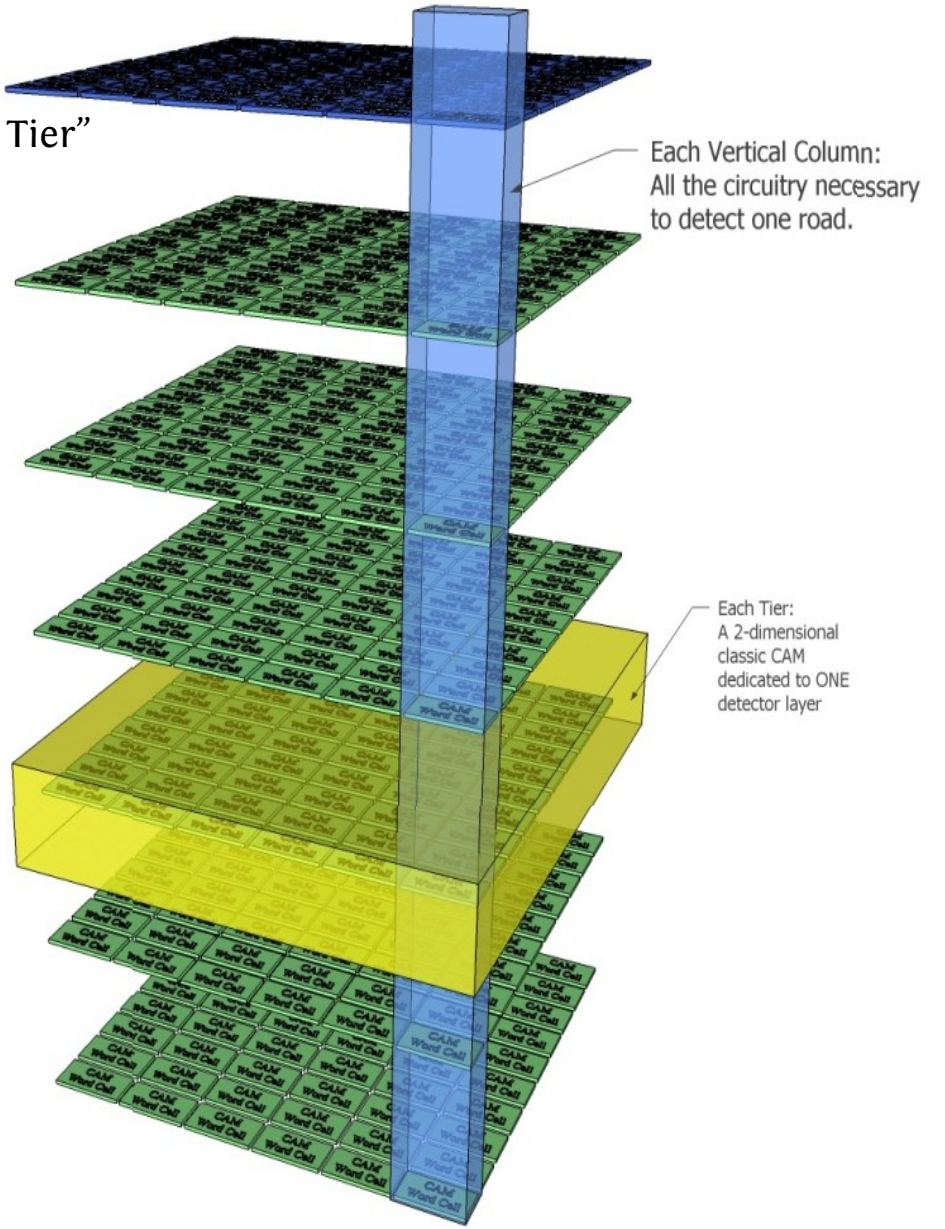


VIPRAM
T. Liu, J. Hoff, G. Deptuch, R. Yarema (Fermilab)

Future evolutions in 3D

* Each detector layer corresponds to a single tier
* All communication from “CAM Tiers” to the single “Control Tier”





VIPRAM
T. Liu, J. Hoff, G. Deptuch, R. Yarema (Fermilab)

Conclusions

- AMchip is a device for solving pattern recognition problems in HEP applications
  - Conceived in the late 80s, developed since 90s
    - AMchip (SVT), AMFPGA, AMchip03 (SVT Upgrade), **AMchip04 (FTK)**
  - Similar to a CAM, but with extended functionalities
    - Separate buses, majority logic, variable resolution
    - It should be considered as a different kind of memory
- AMchip has been developed for use in specific online hardware tracking processors …
  - SVT & Upgraded SVT at the CDF experiment
  - FTK at the ATLAS experiment

  … but it's a general purpose device
    - (other HEP tracking applications: L1 triggers, future 3D devices; outside HEP applications: computational vision)

Conclusions

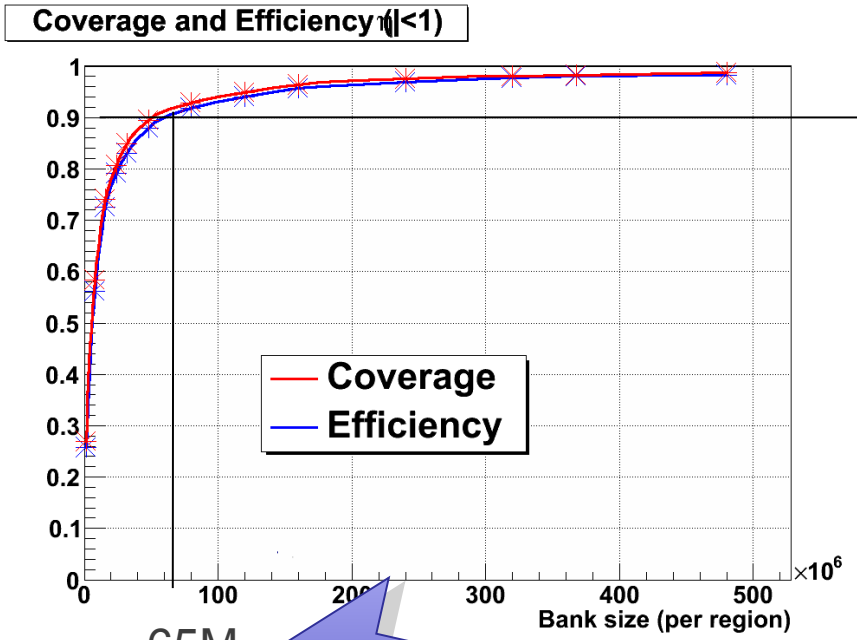
- AMchip04 represents a major improvement in the AMchip family
  - Pattern by pattern variable resolution ("don't care" bits)
  - Matched patterns readout / next event loading at the same time
  - Mixed full-custom / standard cells design
    - CAM blocks as full-custom hard blocks, optimized for low power consumption and area efficiency
    - Control logic in standard cells (easy to develop & debug)
  - New design of majority and readout tree (simple, modular, full-custom version is foreseen)
  - Performance improvements wrt AMchip03: >10x pattern density, <1/10 W/pattern, 100 MHz, 8 buses
- Designed with future evolutions in mind (vertical stacking, full 3D design, …)
- First prototype (8k patterns) will be available by Q1 2012

Backups

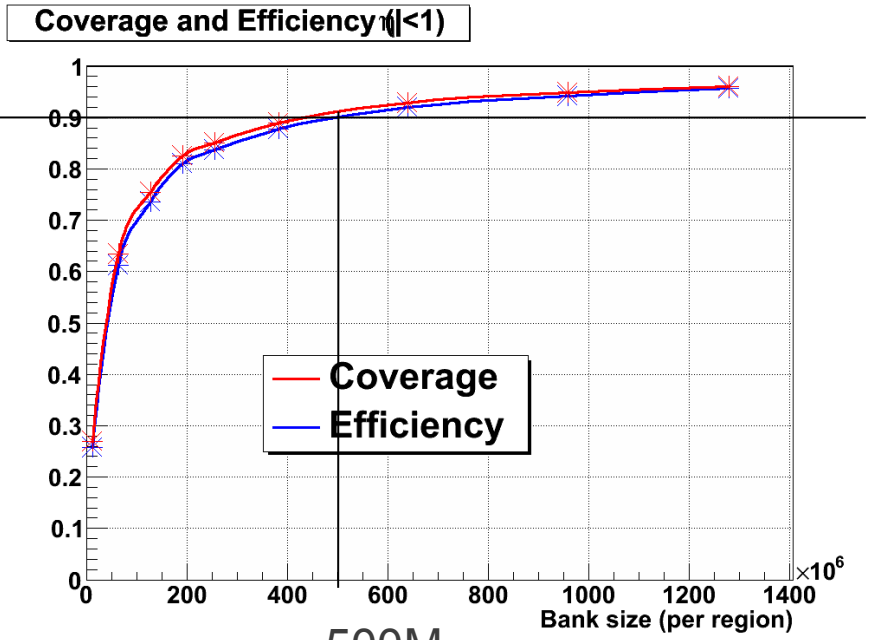
Pattern efficiency

Pattern size
r-ϕ: 24 pixel, 20 SCT
36 pix z

Pattern size
r-ϕ: 12 pixel, 10 SCT
36 pix z



65M



500M

90%

c (barrel only, 45 ϕ degress)

Want this

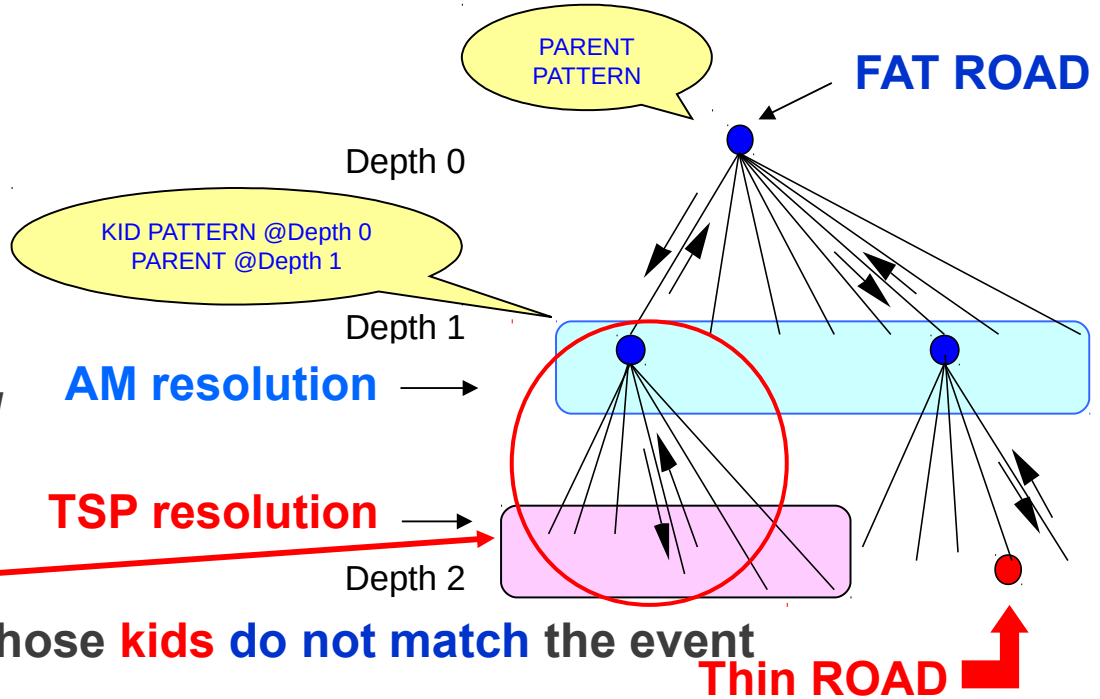
<# roads/event @ 3E34> = 342k

<# roads/event @ 3E34> = 40k

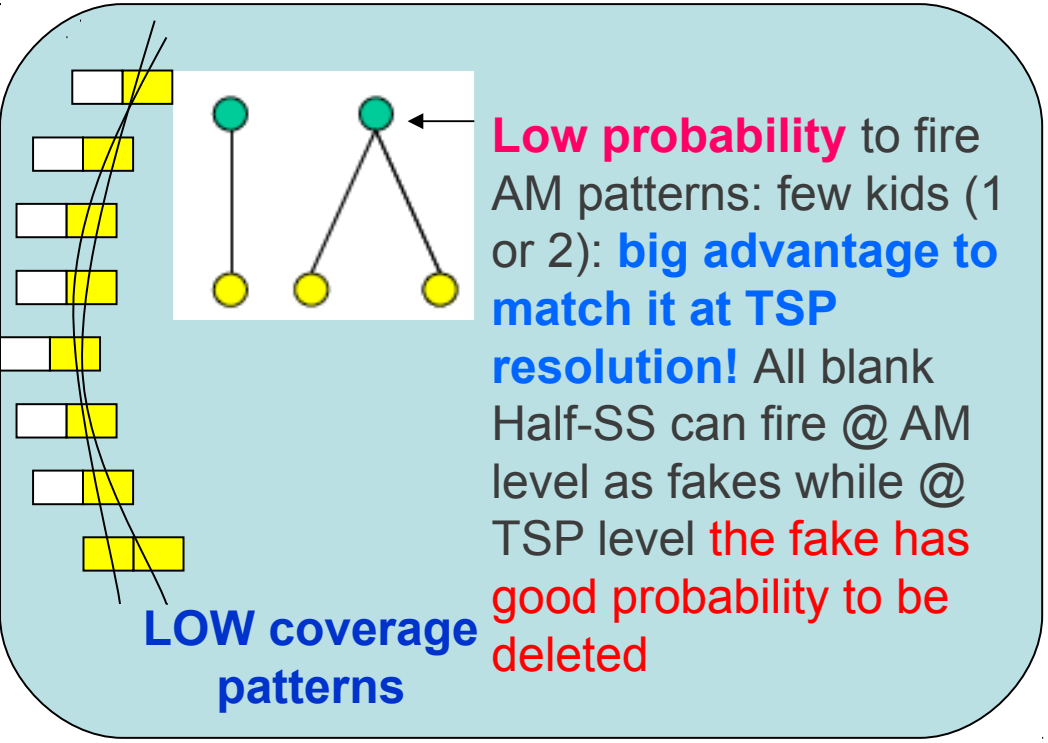
TSP simulation & varying-resolution pattern banks

Guido Volpi & Roberto Vitillo - Pisa

We do have now a **structured** "pattern bank", where each thin road is connected to its parent pattern in FTKsim.
Ongoing tests for TSP algo after the RoadFinder (AMsim) in FTKsim;
we have studied the ***bank composition*** ***and*** ***AM FAKE roads***.



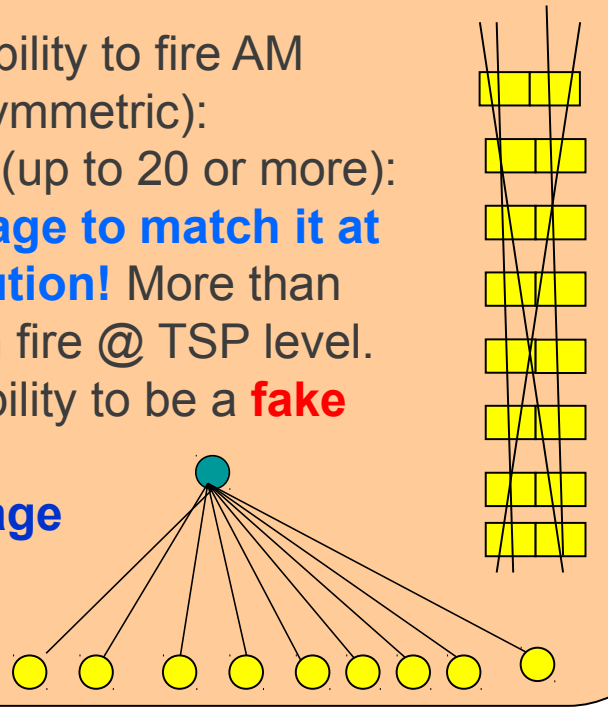
AM Fake road is a AM matched pattern whose kids do not match the event



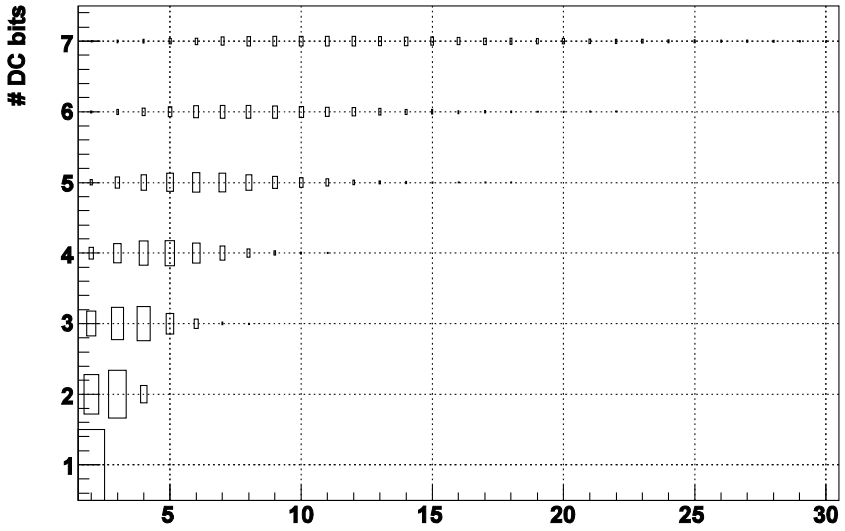
Low probability to fire AM patterns: few kids (1 or 2): **big advantage to match it at TSP resolution!** All blank Half-SS can fire @ AM level as fakes while @ TSP level the fake has good probability to be deleted

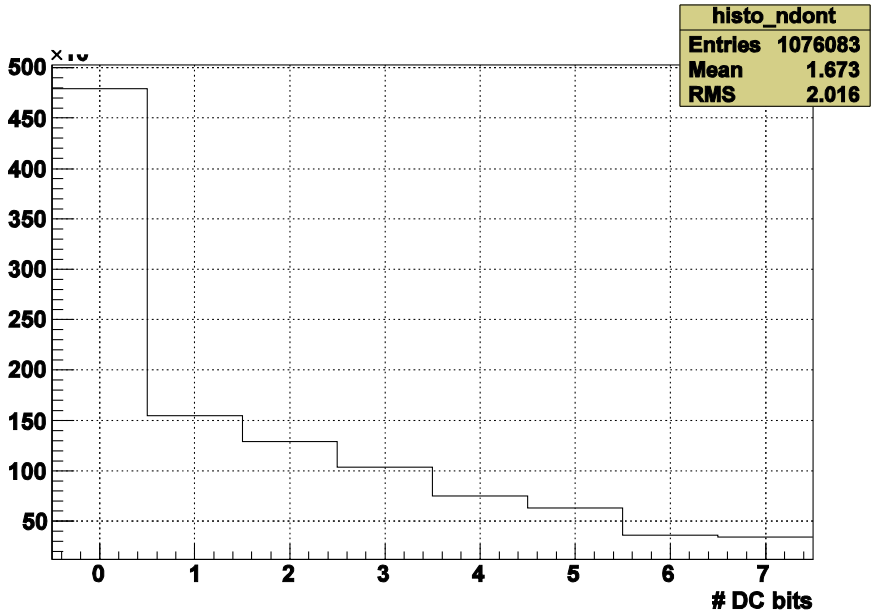
High probability to fire AM patterns (symmetric): many kids (up to 20 or more): **no advantage to match it at TSP resolution!** More than one kid can fire @ TSP level. Low probability to be a **fake AM road**

HIGH coverage patterns



AM with care/don't care





	<roads/event>
TSP	38000
AM@TSP	28000
AM@DC	44000
AM	342000

Care/don't care very effective to reduce the number of roads.

Area cost on the chip approx. 1 extra cell for each DC bit.

Now 15 cells/layers.

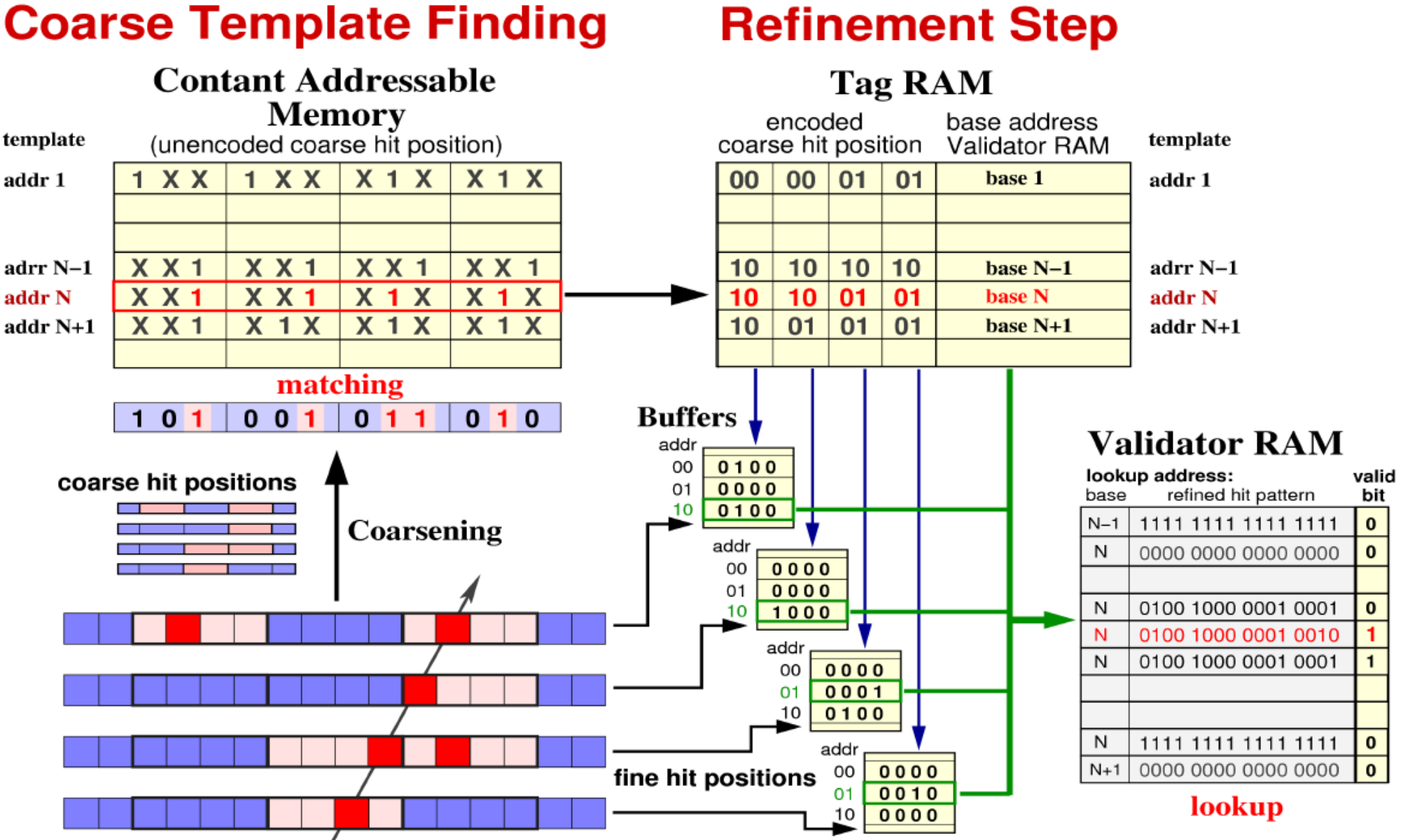
With 1 DC bit area increases by 1/15 ~ 7%.

For comparison going to TSP resolution would require 3x patterns.

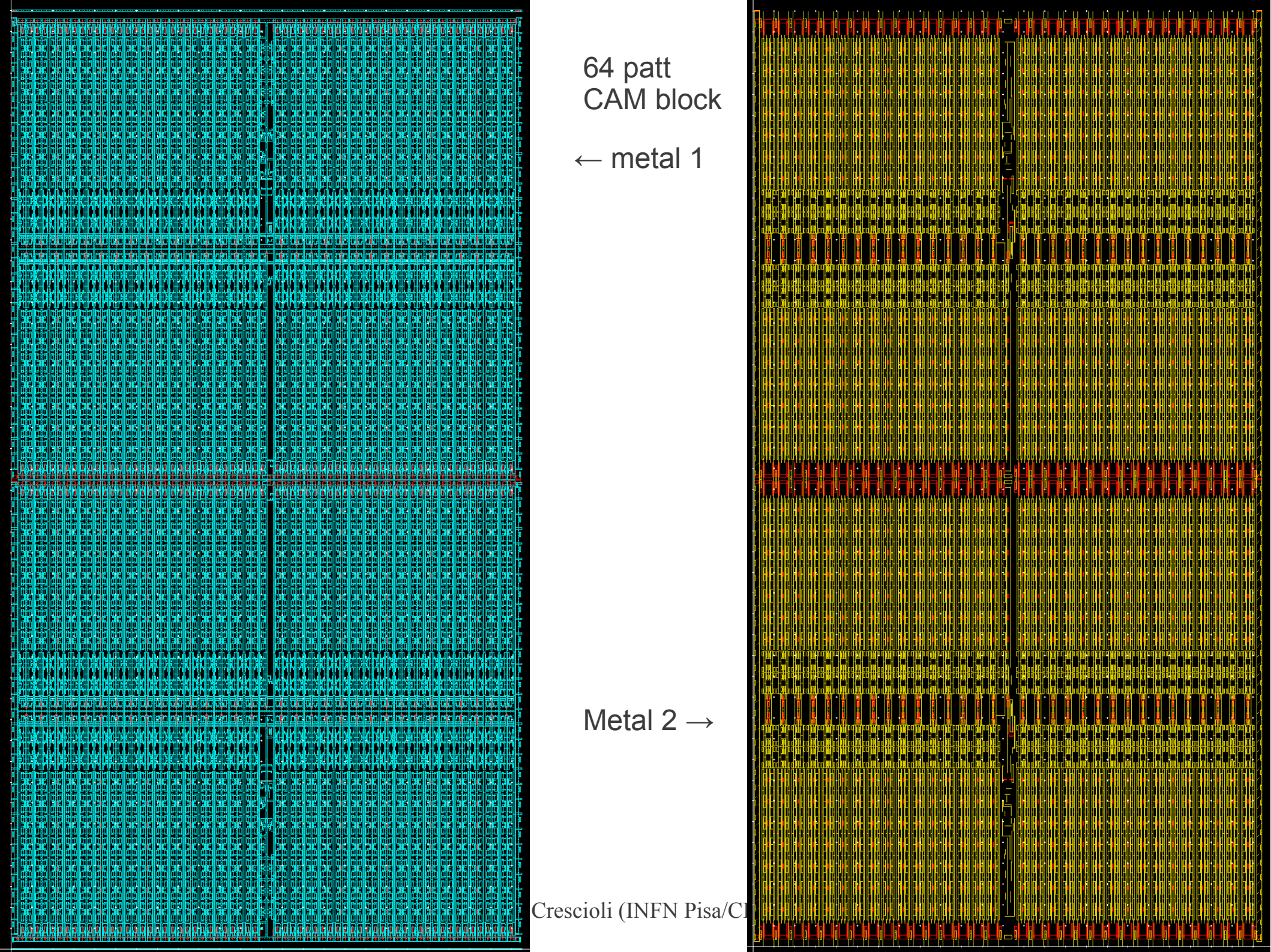
Main developers

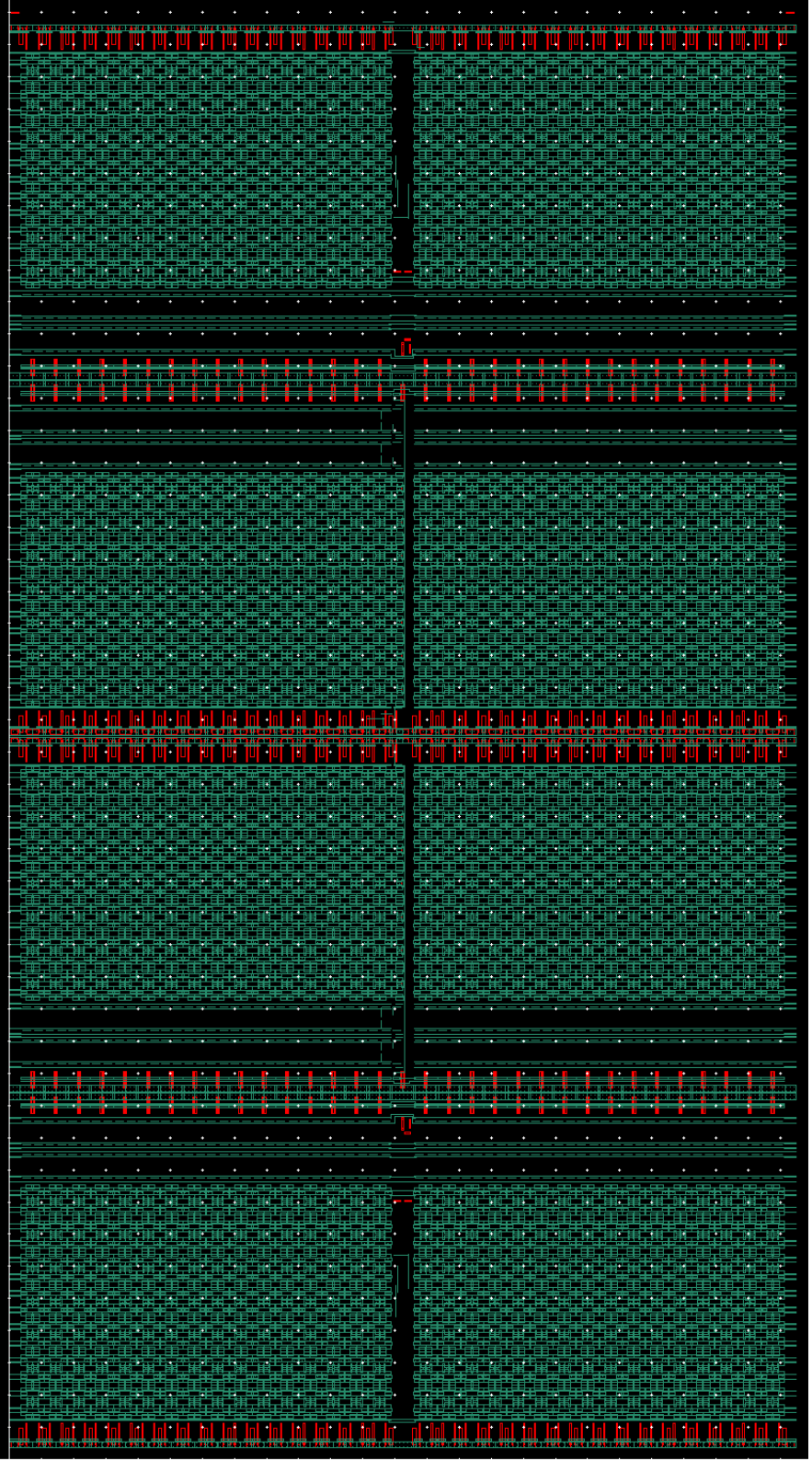
- Full custom design and layout: M. Beretta, A. Stabile
  - Tools: Virtuoso
- Back end implementation: A. Stabile, F. Crescioli
  - Tools: EDI 9.1
- Front end implementation: F. Crescioli
  - Tools: Design Compiler
- C++/Verilog simulation: F. Crescioli, I. Sacco
  - Tools: custom C++ emulator, NC Sim
- Coordination & ideation: A. Annovi, P. Giannetti, V. Liberali

Fast pattern recognition with dedicated HW



(from A. Schoening, S. Schmitt ideas for ATLAS L1 tracking trigger)

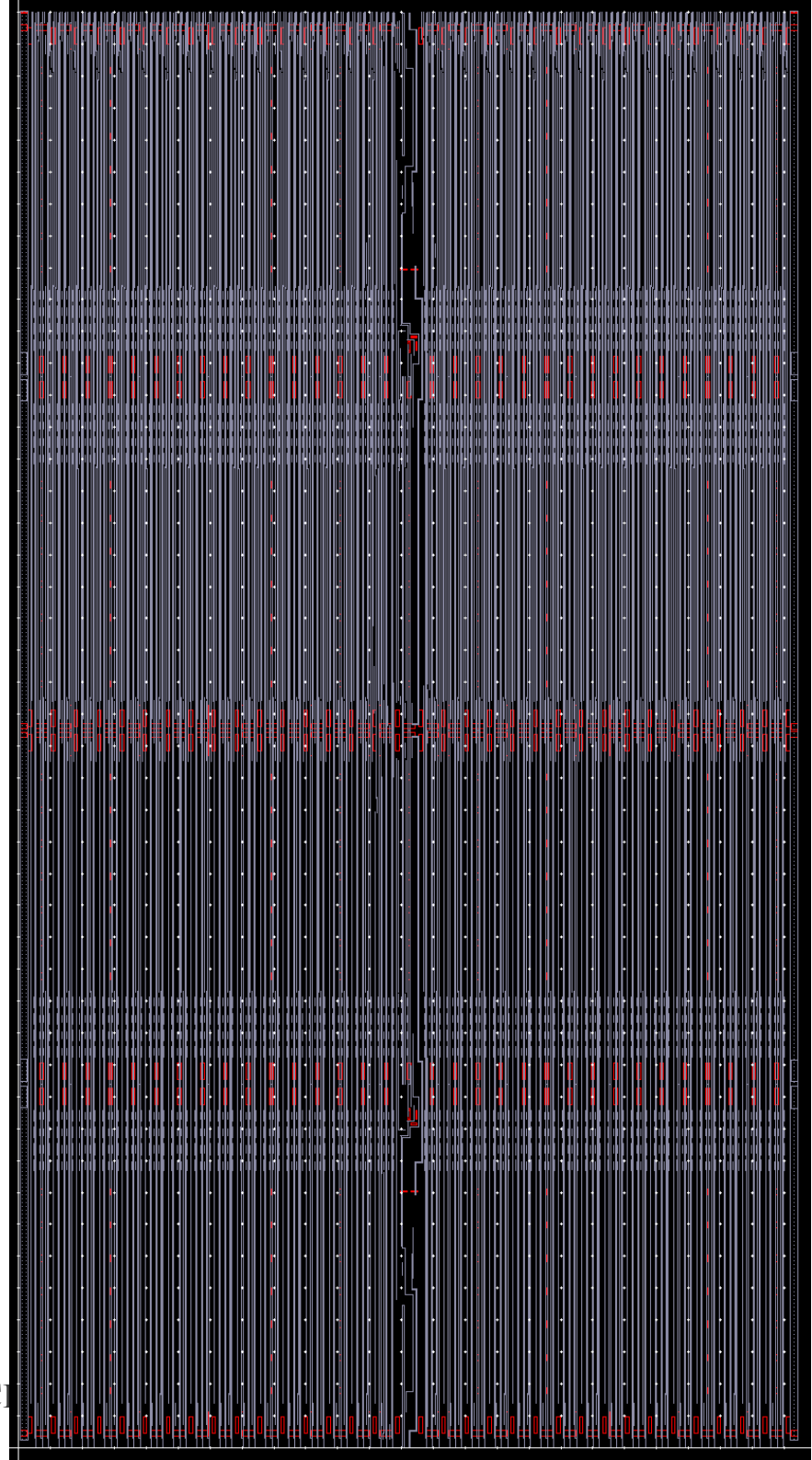

64 patt
CAM block
← metal 1
Metal 2 →

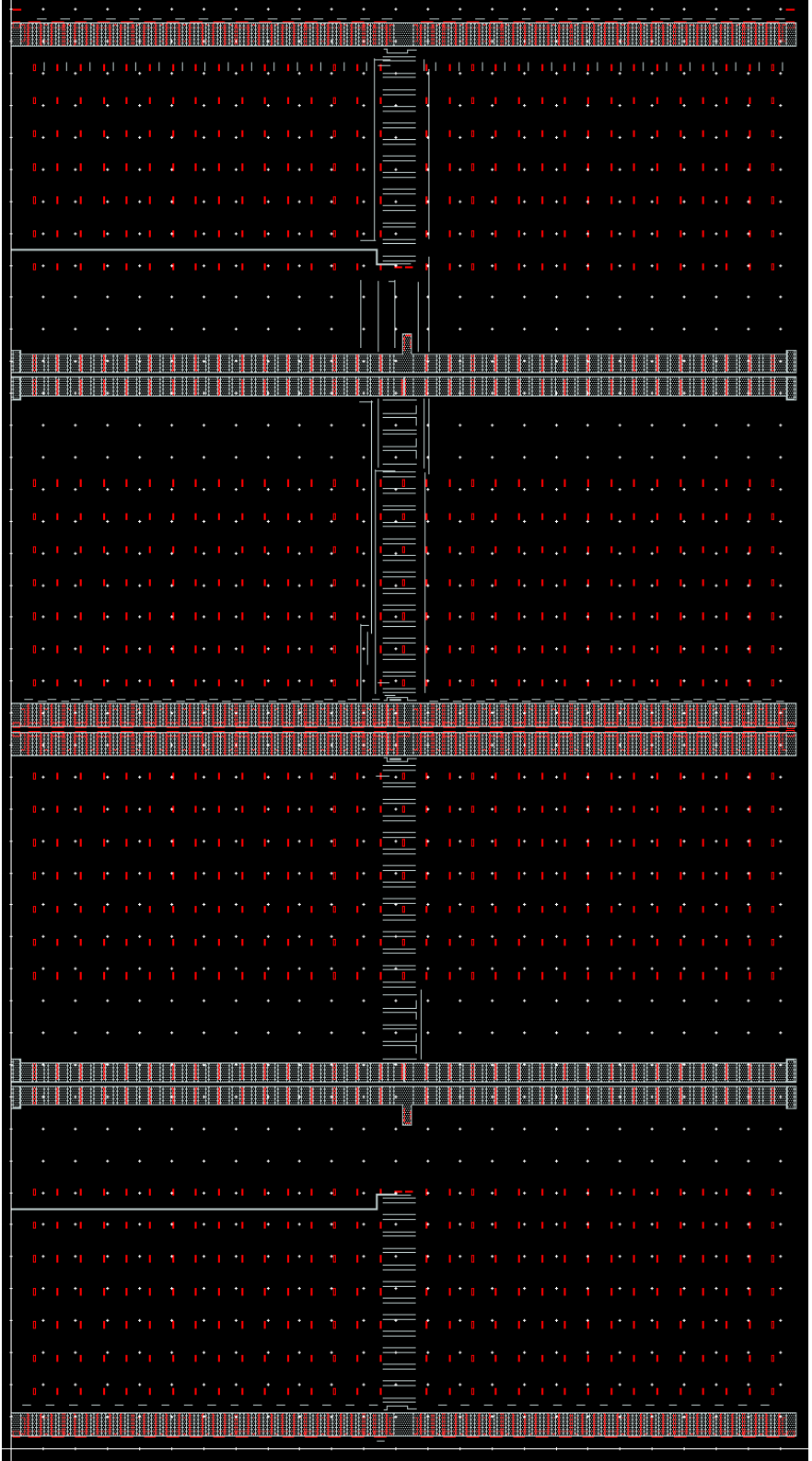


64 patt
CAM block

← metal 3

Metal 4 →

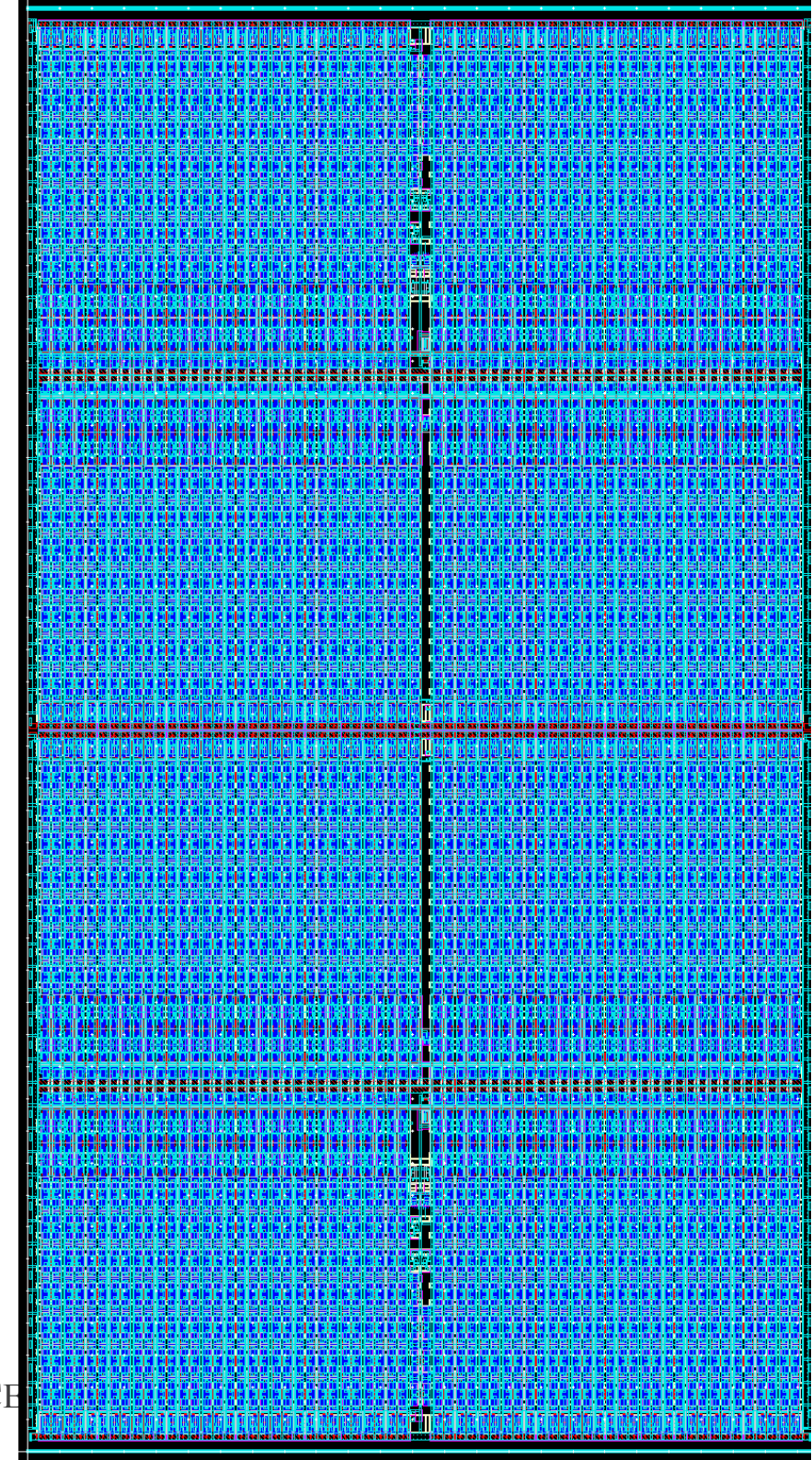


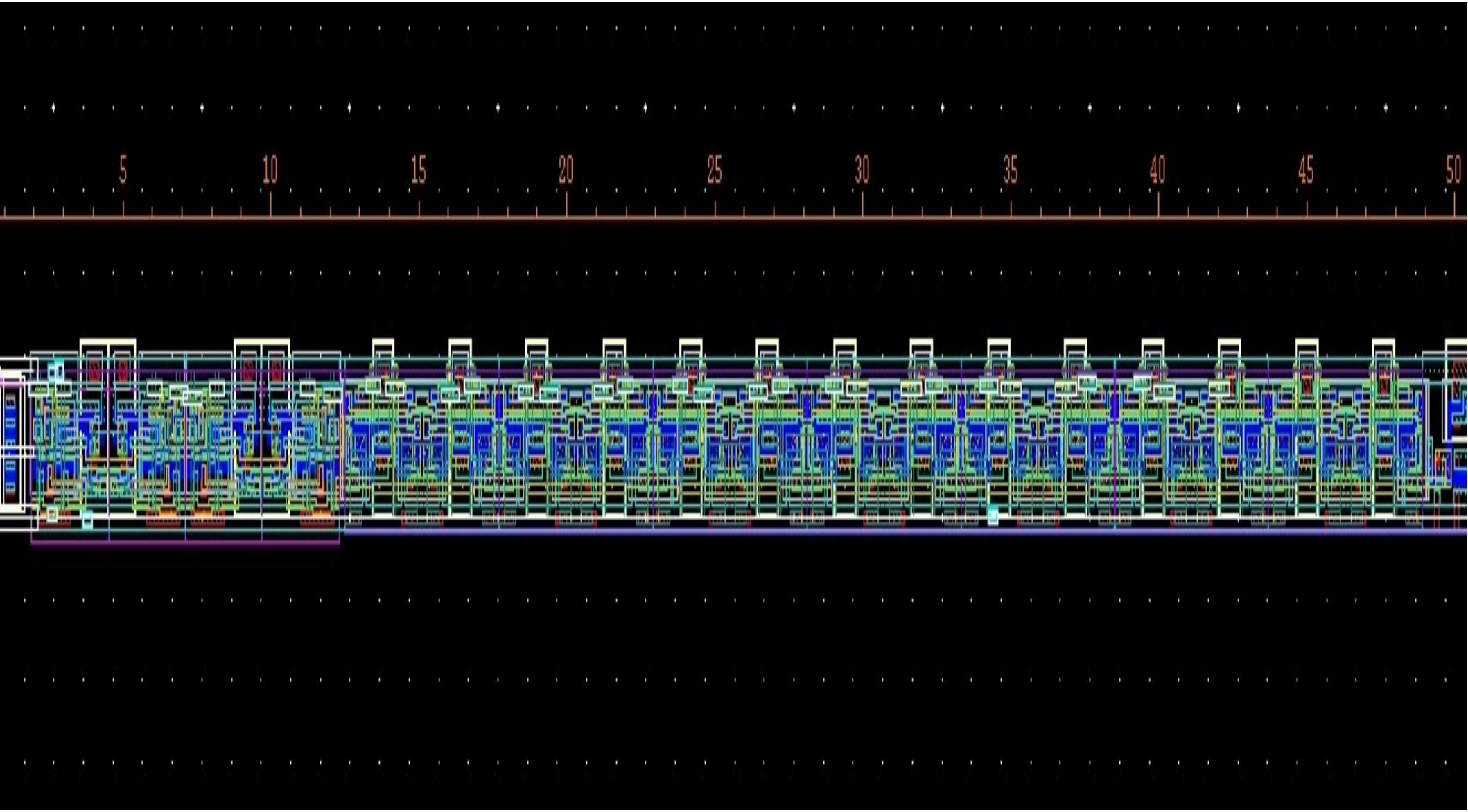


64 patt
CAM block

← metal 5

trans →




5
10
15
20
25
30
35
40
45
50