# AM chip pattern recognition

## with optimized ternary bit usage


Connecting the Dots
March 2017, Orsay, France

Stefan Schmitt, DESY
on behalf of the ATLAS Collaboration

# Outline

- The FTK system for ATLAS
- The AM chip
- Ternary bits in the AM chip
- Producing patterns from constants
- Basic algorithm to set ternary bits
- Refined algorithm to set ternary bits
- Performance comparison

Figures and further material about the FTK project:
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/FTKPublicResults

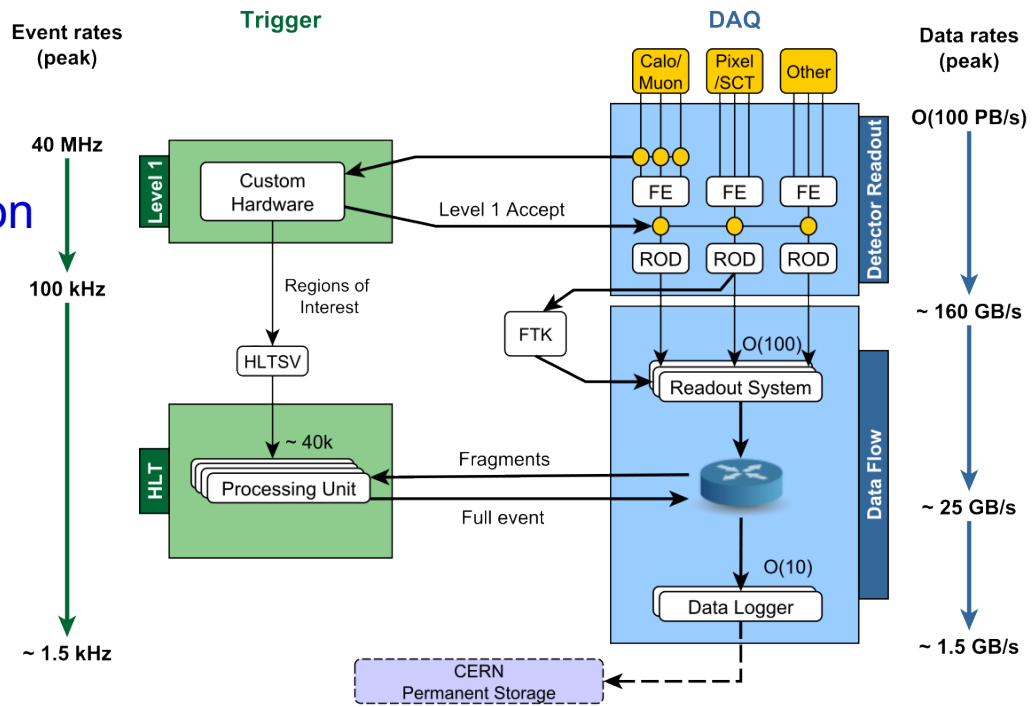
# ATLAS and the FTK system

## ATLAS trigger:

- First-level trigger (L1): based on calorimeter and muon chambers. Rate 100 KHz
- High-level trigger (HLT): verify L1 decision in regions of interest. Refinement using event fragments, final trigger decision at ~1.5 KHz output rate

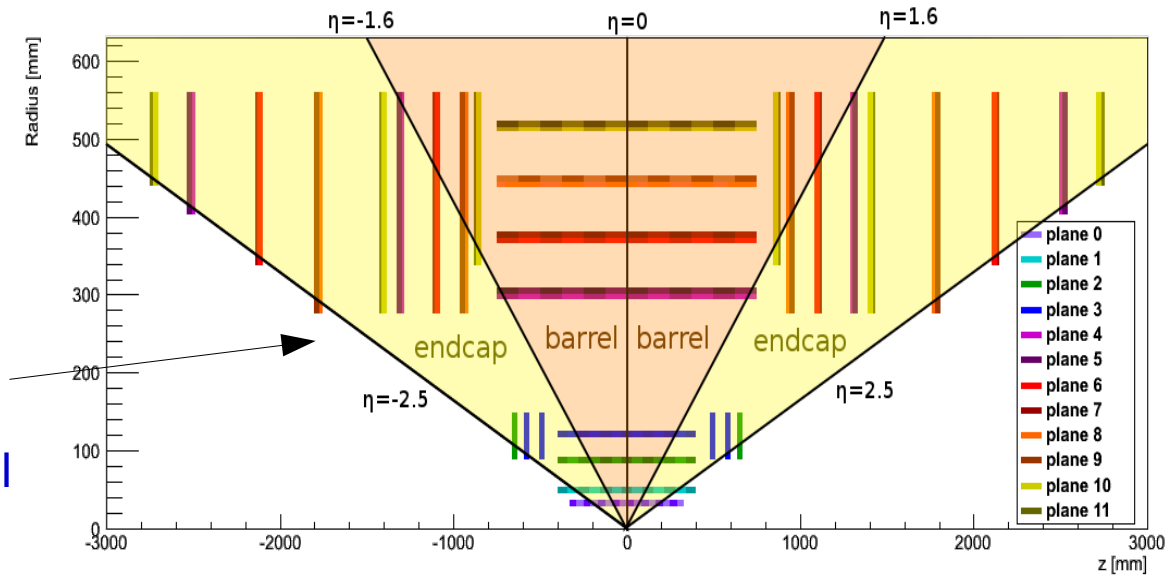
## ATLAS Fast TracKer (FTK):

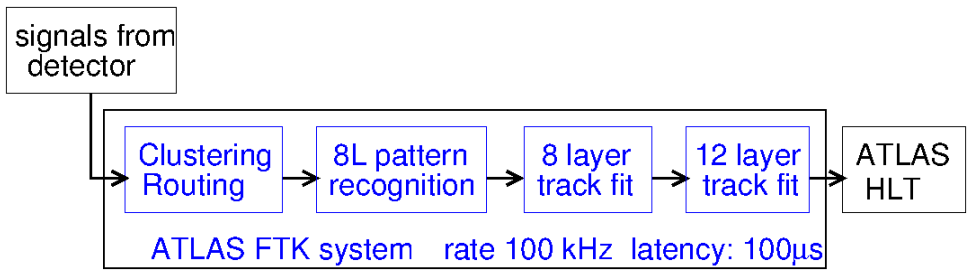
- Reconstruct tracks with $P_T$>1 GeV at 100 Khz rate
- Track-based decisions early in the HLT → improved performance
- Hardware installation and commissioning ongoing



# The FTK system

- Detector elements are assigned to 12 planes or layers
- 4 pixel, 8 strip (sct) layers
- Clustering and routing to 64 towers 16×4 segments in φ×η
- η segmentation: barrel and endcap
- Pattern recognition: 8 layers (3 pixel + 5 sct) at coarse position resolution→ AM chip
- First-stage fit: 8 layers
- Second stage fit: 12 layers
- Latency: 100 μs

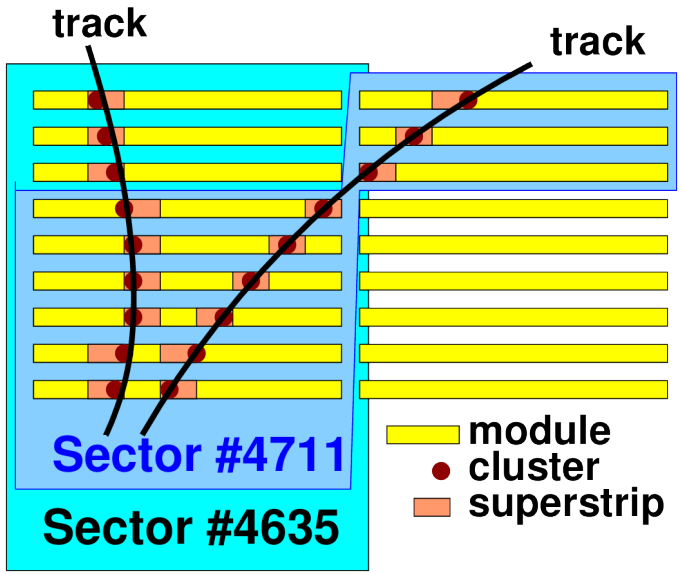


# FTK Track fits

- Track fits are based on a linearized model:

  track parameters ($d_0$,φ,$1/p_T$,$z_0$,cotθ) are related to hit positions by linear equations
- Model is valid for small geometrical regions (sectors). There are more than $10^5$ sectors used in the FTK system
- Each sector has its own fit constants (coefficients of the linear equations)
- Hit positions at coarse resolution (superstrips) are used for the pattern recognition in the AM chip

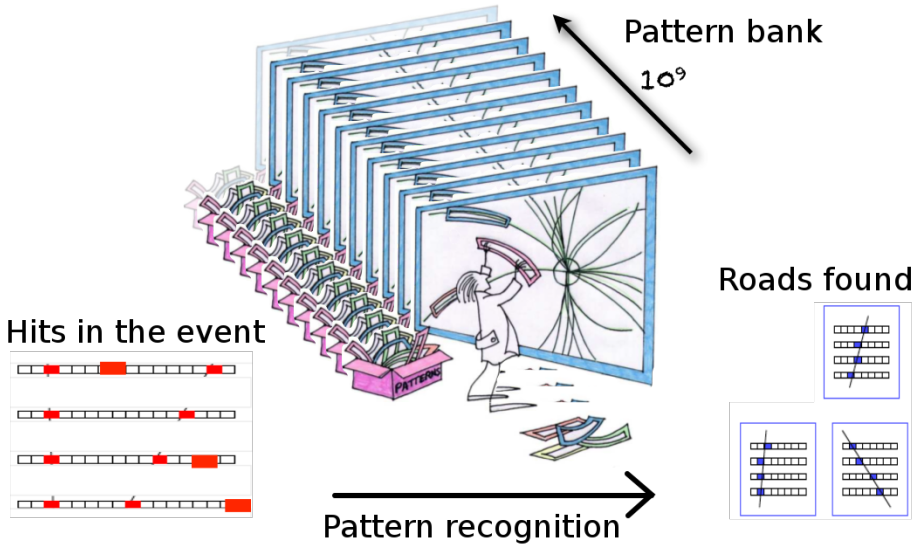



Sector: unique combination of detector modules across layers

Superstrip: coarse hit position for pattern recognition, encoded in a 15-bit word

# FTK pattern recognition

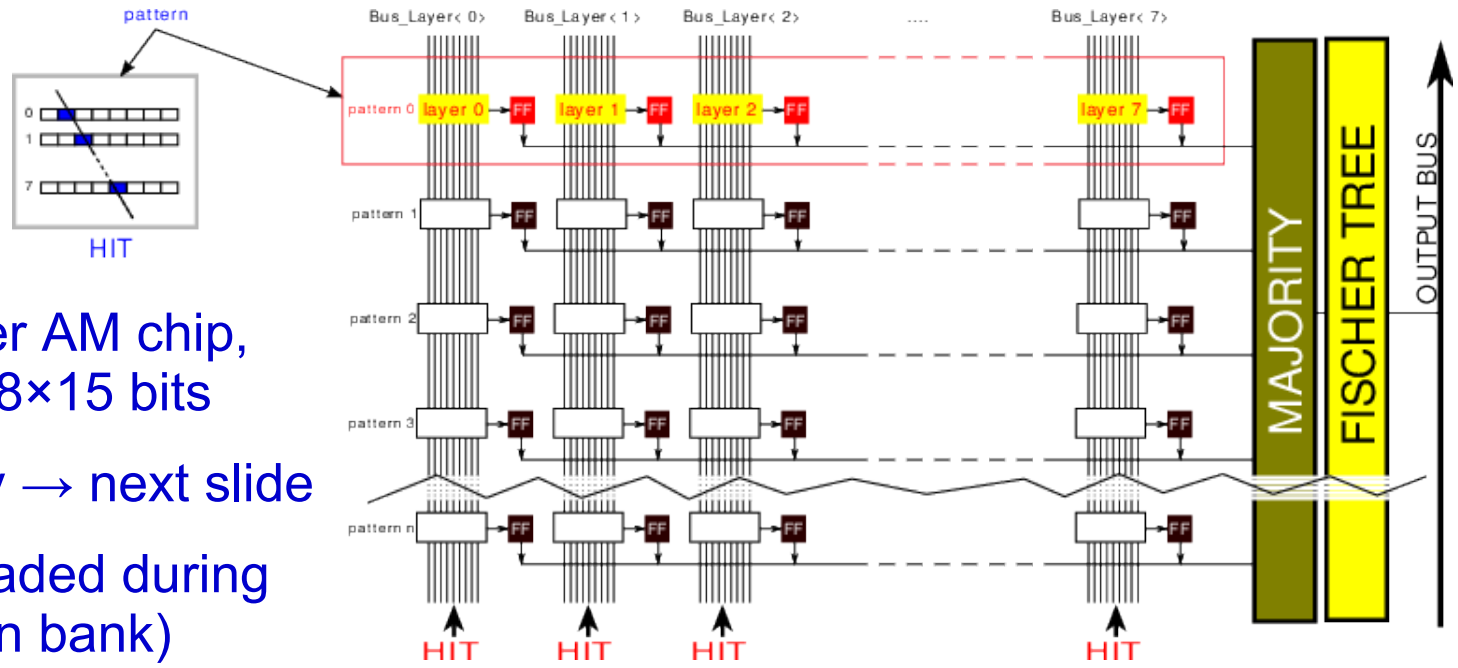
- Pattern: a set of eight 15-bit words (coordinates), one from each detector layer used for the pattern recognition
- Set of predefined patterns: “pattern bank”
- The superstrips of a given event are compared to the pattern bank.
- Valid patterns (“roads”) where 7 or 8 layers have a match are used for the track fit
- The track fit proceeds using hits at full position resolution




- This talk: about creating the pattern bank
- Emphasis: use of the ternary bits provided by the AM chip

# The AM chip

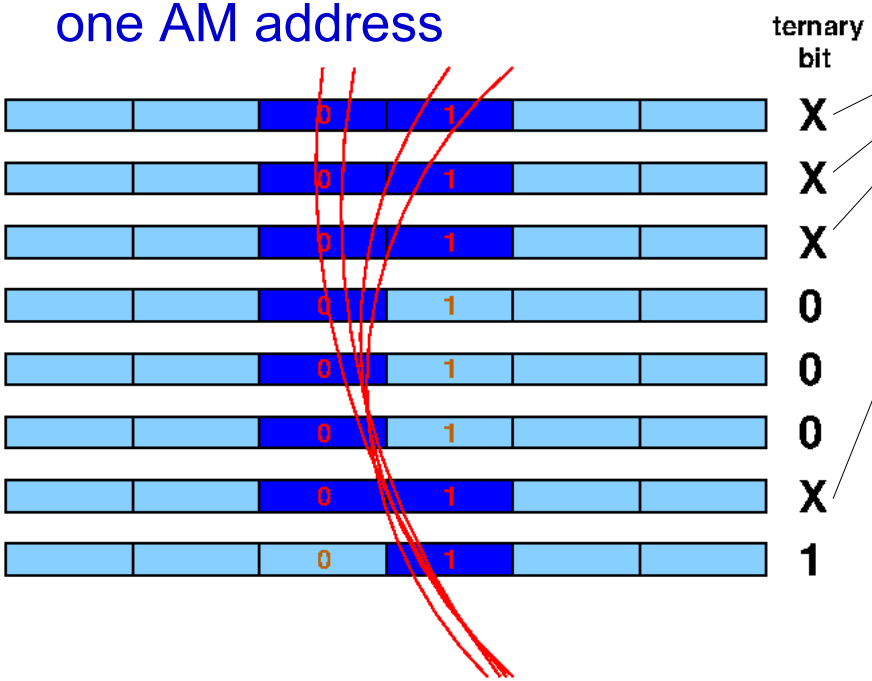
- Based on eight associative memories, one per layer
- 128K addresses per AM chip, each address with 8×15 bits
- 8x3 bits are ternary → next slide
- Memory content loaded during initialization (pattern bank)




- Associate memory: hit data are compared at all addresses simultaneously → very fast
- Patterns with 7 or 8 matching hits are read out

# Ternary bits in the AM chip

- Ternary bit: three states {0,1,X}
- When comparing to a hit, X matches both 0 and 1
- Used to encode similar patterns in one AM address




## How large is a pattern?

- For a given pattern: count number of bits in state X: $N_x$

  Example: four layers with one bit in state X, $N_x$=4
- Given $N_x$, count number of valid bit combinations

  Example $N_x$=4: XXXX=0000,0001,...1111

  → 16 combinations
- General formula: $2^{N_x}$ (in this talk: "pattern volume")
- For the AM chip: 8 layers and 3 ternary bits each →maximum pattern volume is

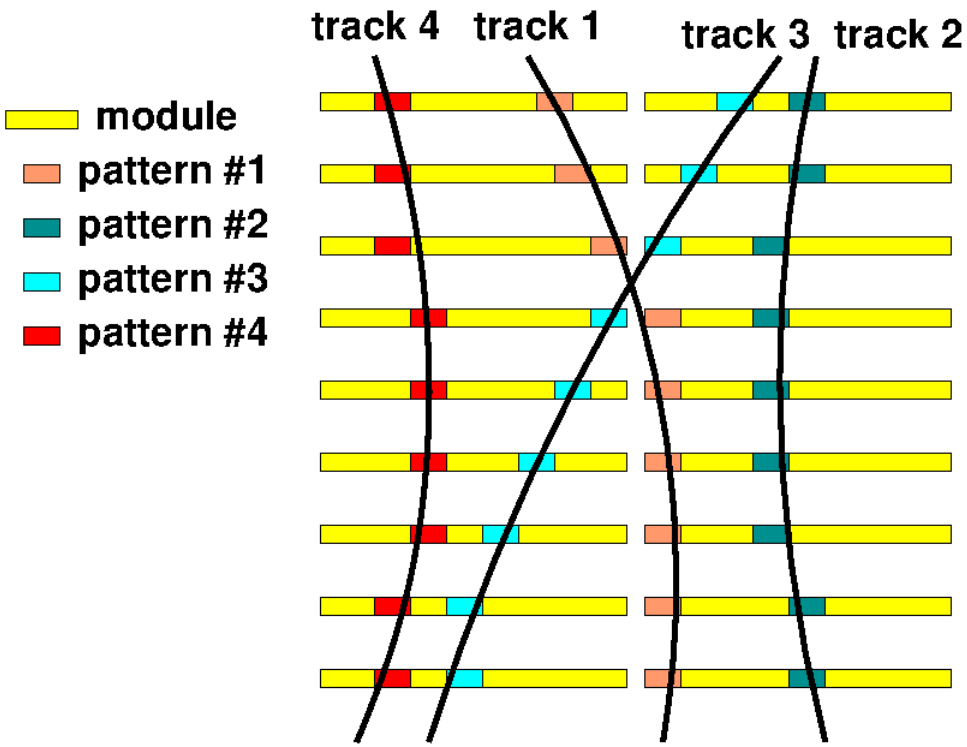
  $2^{3\times8}$ ~ 4 million patterns

# Creating large pattern samples

- FTK fit constants can be used to predict positions in tracker

  generate large samples ($10^9$ per tower) of valid track positions using the fit constants and randomly generated track parameters

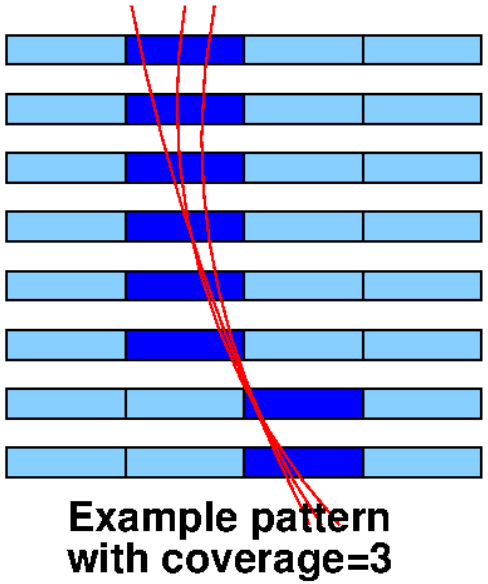
- Each track has a corresponding pattern [set of superstrips in 8 layers]
- Store patterns and count duplicates

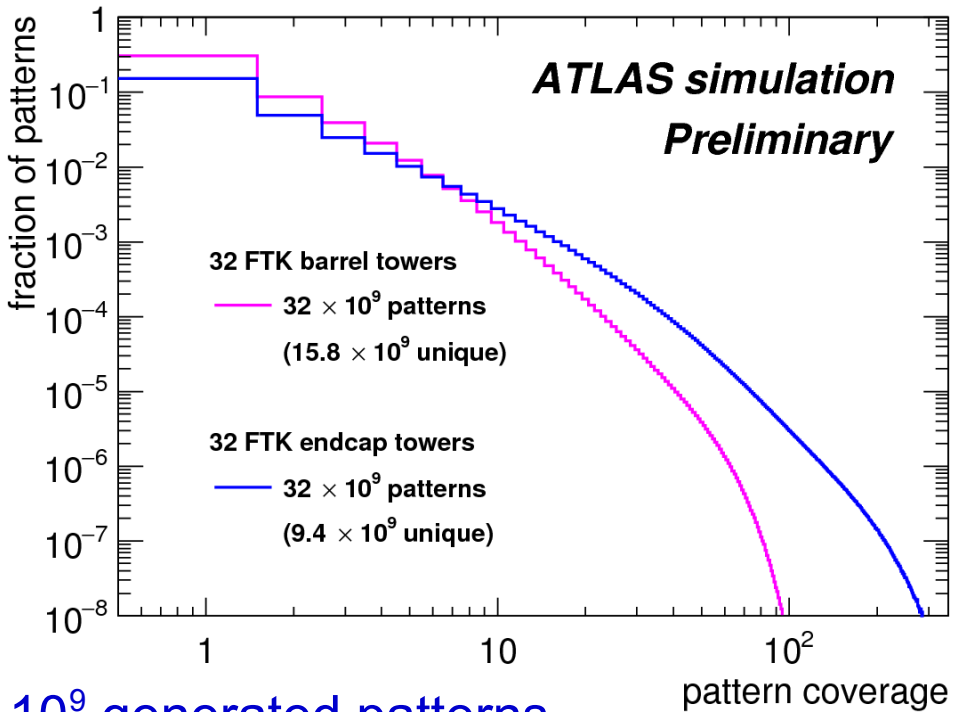



Example: generate four random tracks and determine the patterns

# Creating large pattern samples

- Several generated tracks may correspond to the same pattern
- Number of generated tracks per pattern: coverage

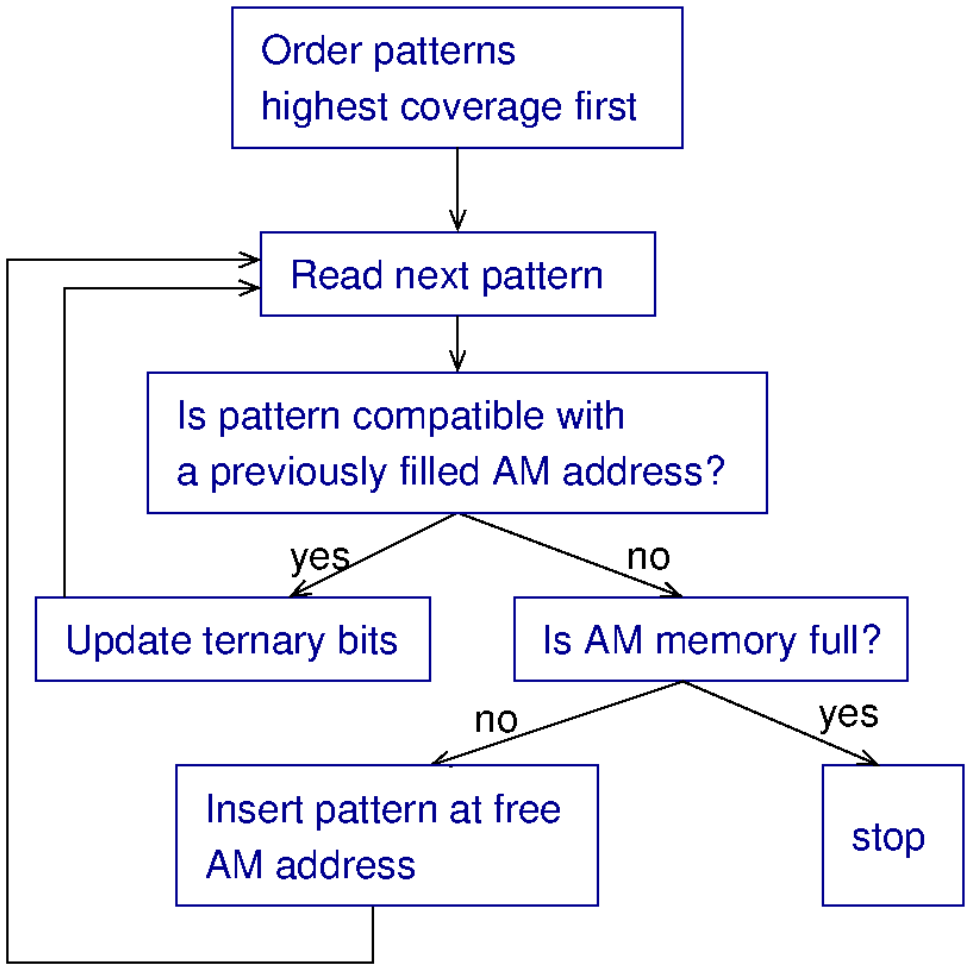



- Shown here: coverage distribution of the $10^9$ generated patterns
- High coverage patterns: most relevant for high efficiency
- How to best fit the $10^9$ generated patterns in the 16.8M available addresses?

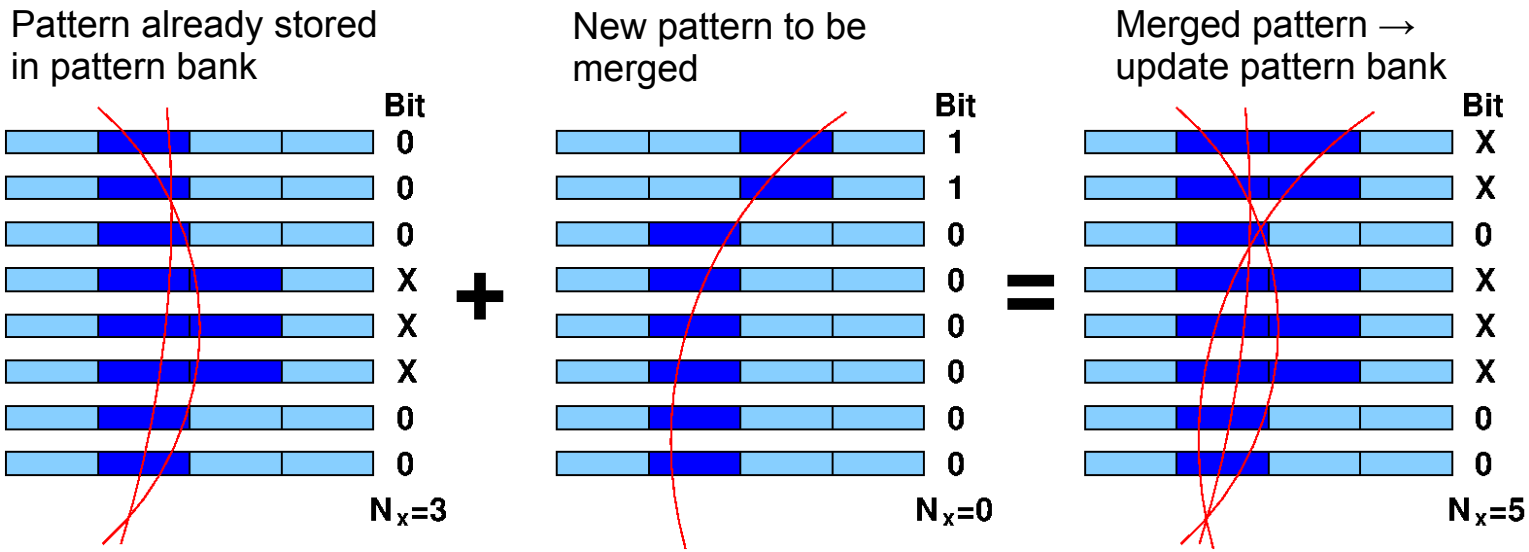
16.8M patterns: 128 AM chips (128K each) per tower

# Basic AM pattern algorithm




- One billion patterns per tower generated
- Have to select the most important ones and construct ternary bits for the available 16.8 million AM addresses per tower
- Algorithm: assign patterns to AM addresses, starting with highest coverage
- If possible, merge similar patterns in a single address using ternary bits

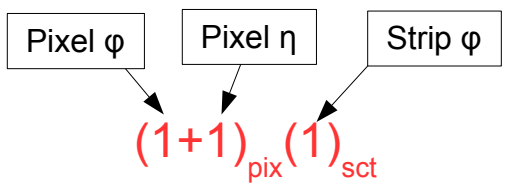
# Ternary bit merging




- Patterns which differ only at the ternary bit positions can be packed into a single AM address
- Example: a pattern with $N_X$=3 changes to $N_X$=5 after adding another pattern

# $N_x$ distribution

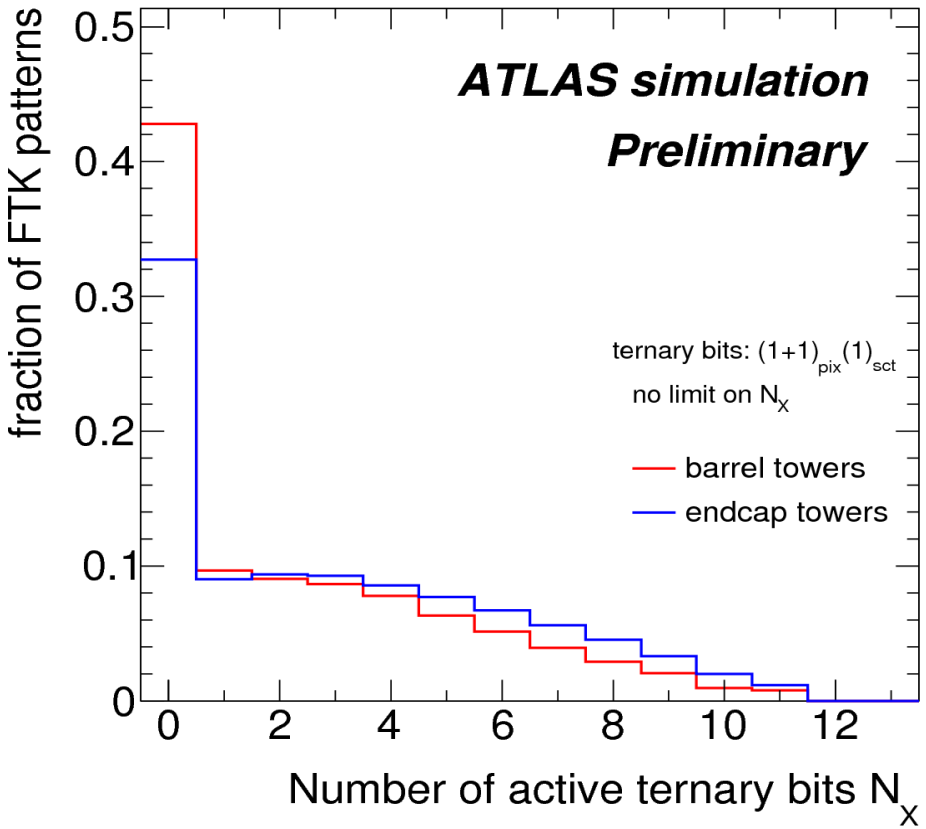
- Shown here: distribution of $N_x$ after processing the $64\times10^9$ patterns
- Configuration used has 11 ternary bits per pattern:




3 pixel (pix) layers
5 strip (sct) layers

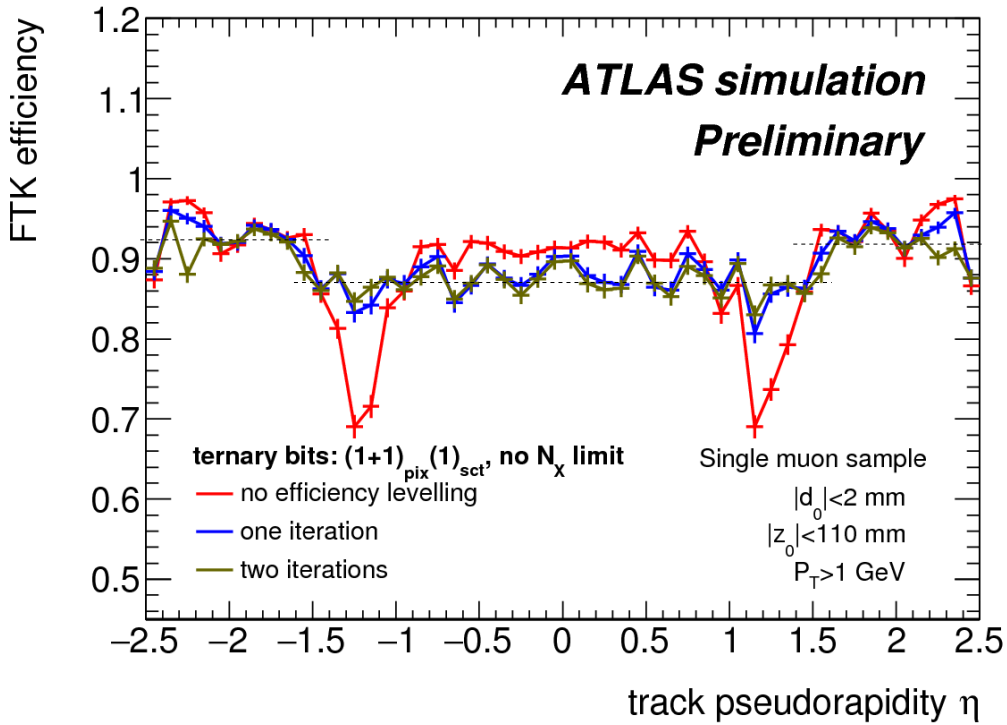
- Many addresses have only one pattern encoded ($N_x$=0)
- Different distribution for barrel and endcap towers
- Tails to $N_x$=11 → large pattern volumes up to $2^{11}$=2048

→ danger of random coincidences & fake tracks



# Basic algorithm: efficiency

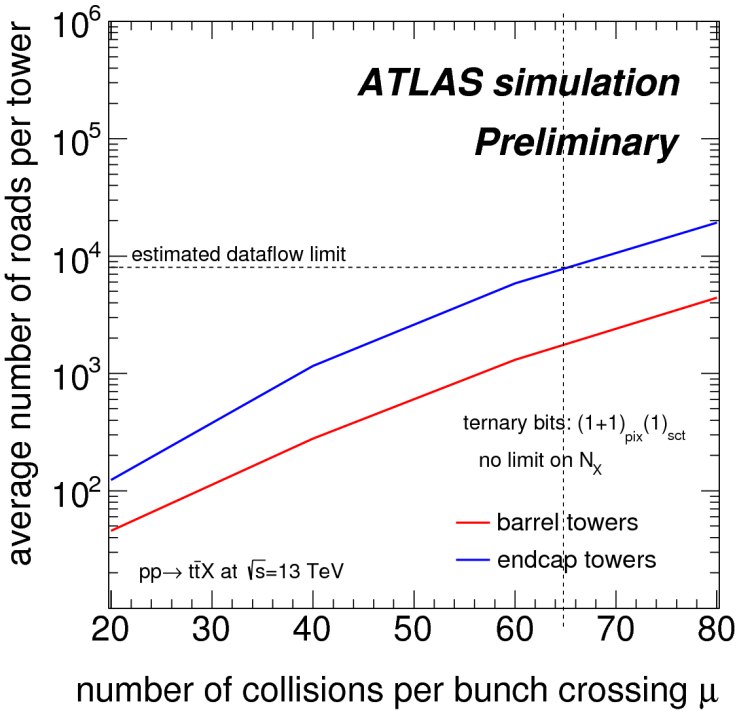
- Quantities to predict the FTK perfomance:
  - Track reconstruction efficiency
  - Data flow: compare to hardware specifications




- Efficiency: around 90%, on the low side in the barrel region
- Inefficiencies near η=±1.2 are related to change in geometry (barrel→disk)
- Resolved by iteratively improving the pattern bank → backup slides

# Basic algorithm: dataflow

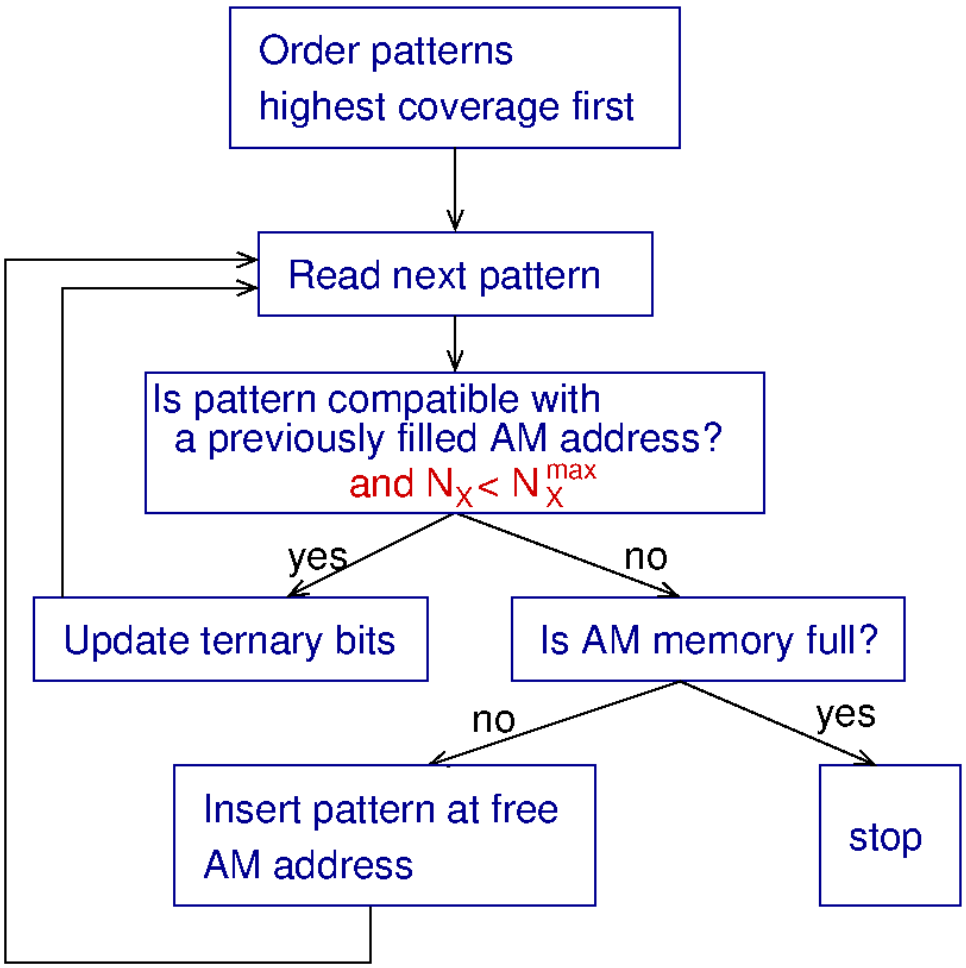
- Quantities to monitor the perfomance:
  - Track reconstruction efficiency
  - Data flow: how does it compare to hardware specifications




- Shown here: number of roads (patterns with 7 or 8 matches) as a function of the number of collisions per crossing [μ]
- Number of roads grows fast with μ
- Much higher dataflow in endcap wrt barrel
- In this example, hardware limitation is reached at μ~65 for endcap

  [exceeding the limit → deadtime]

# Refined algorithm




- Idea: when loading patterns, avoid creating large pattern volumes $2^{N_X}$
- Simple extension of the basic algorithm with adjustable parameter

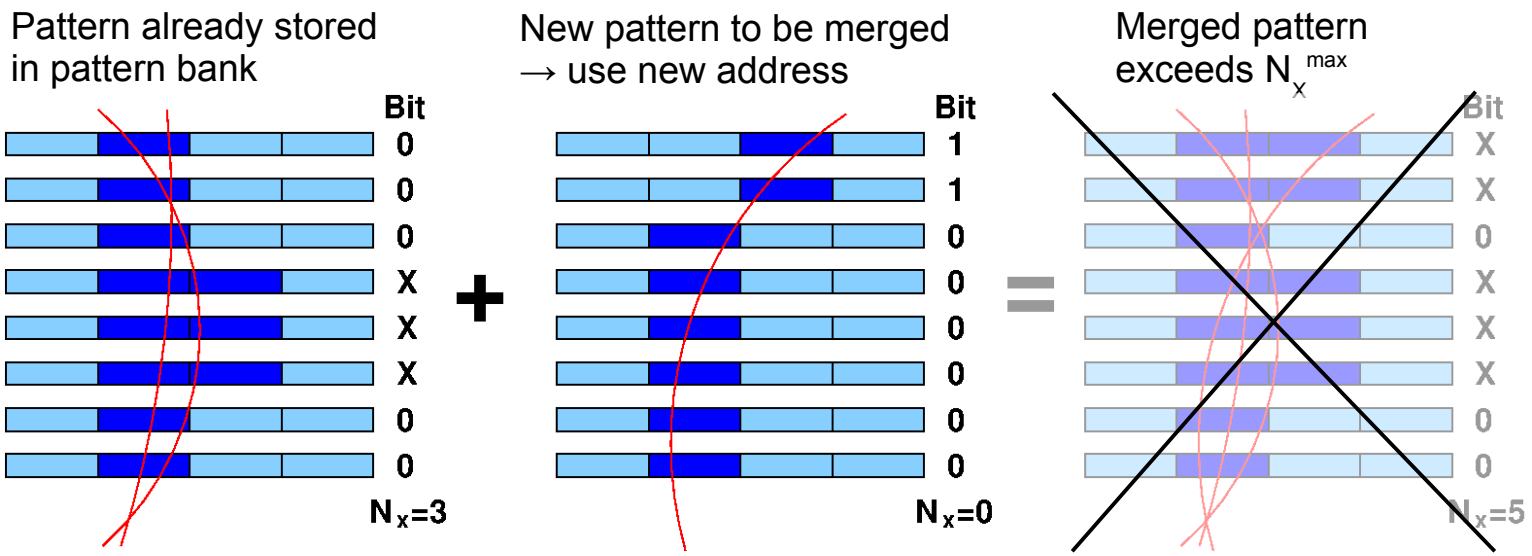
  $N_X^{max}$

  - Merge pattern ternary bits only if pattern matches

    **and if $N_x < N_x^{max}$**

  - Otherwise insert pattern at a new AM address
- Fine-tune $N_x^{max}$ to optimize efficiency and dataflow

# Ternary bit merging with limit




- Patterns which differ only at the ternary bit positions can be packed into a single address
- Addition: avoid large $N_x$ by starting a new pattern if $N_x \geq N_x^{max}$

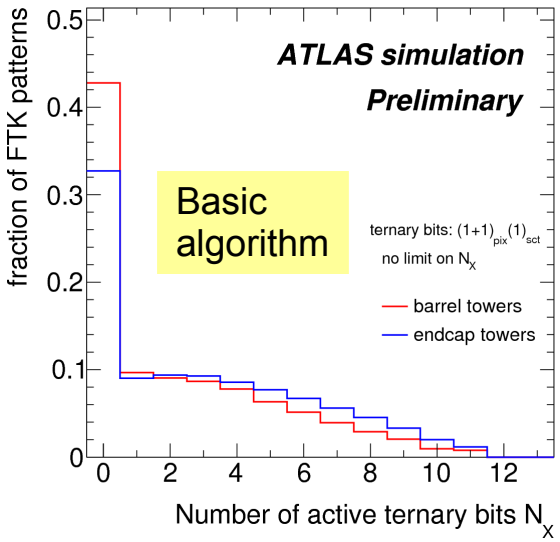
  [Example given: $N_x^{max}=5$, so the merged pattern with $N_x=5$ is not allowed]

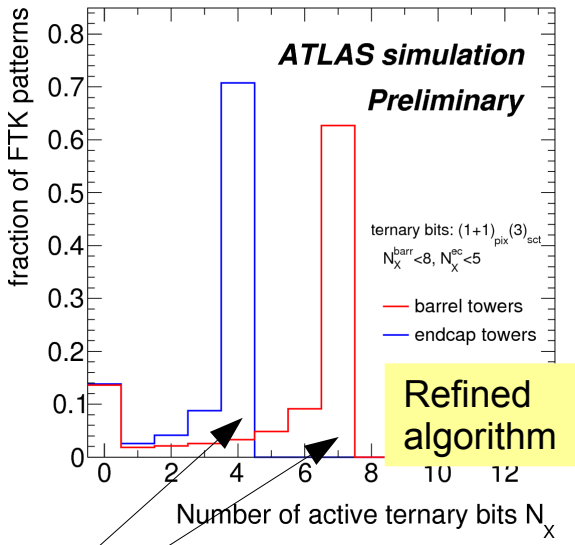
# $N_X$ distribution comparison

- Comparison of the two algorithms using settings which result in similar efficiencies
- Refined algorithm $(1+1)_{pix}(3)_{sct}$

  2 ternary bits per pixel, 3 per strip, 21 total
  Limit $N_x$<8 (barrel), $N_x$<5 (endcap)
- Basic algorithm $(1+1)_{pix}(1)_{sct}$

  2 ternary bits per pixel, 1 per strip, 11 total
  No limit on $N_x$




- Effect of limiting $N_x$ is clearly visible: Patterns accumulate near the limit

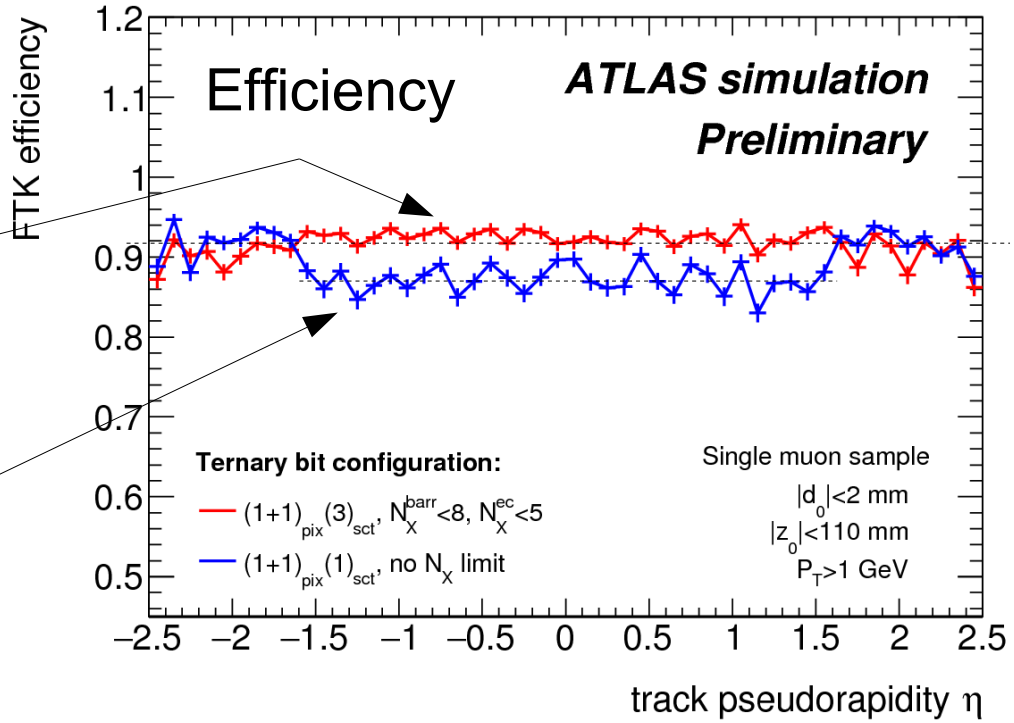
  Basic algorithm: $N_x$ varies a lot between AM addresses.
  Refined algorithm: $N_x$ is similar for most AM addresses.

# Efficiency comparison

- Comparison of the two algorithms using settings which result in similar efficiencies
- Refined algorithm $(1+1)_{pix}(3)_{sct}$

  2 ternary bits per pixel, 3 per strip, 21 total
  Limit $N_X$<8 (barrel), $N_X$<5 (endcap)
- Basic algorithm $(1+1)_{pix}(1)_{sct}$

  2 ternary bits per pixel, 1 per strip, 11 total
  No limit on $N_X$




- Efficiency is improved in the barrel region |η|<1.6 when using the refined algorithm with appropriate settings

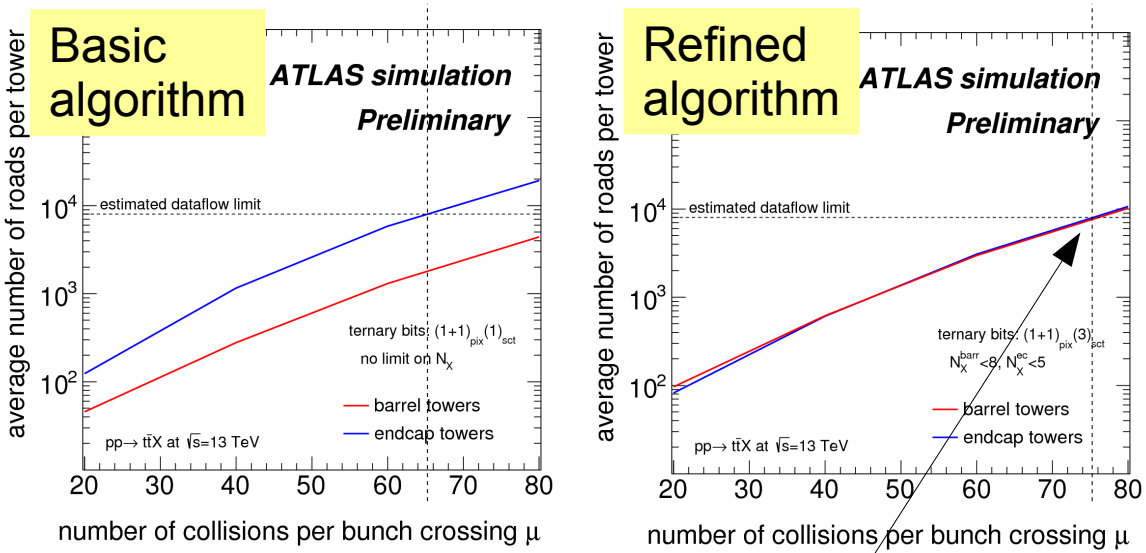
# Dataflow comparison

- Comparison of the two algorithms using settings which result in similar efficiencies
- Refined algorithm $(1+1)_{pix}(3)_{sct}$

  2 ternary bits per pixel, 3 per strip, 21 total
  Limit $N_x$<8 (barrel), $N_x$<5 (endcap)
- Basic algorithm $(1+1)_{pix}(1)_{sct}$

  2 ternary bits per pixel, 1 per strip, 11 total
  No limit on $N_x$

Average number of roads per tower and event, as a function of the pileup μ

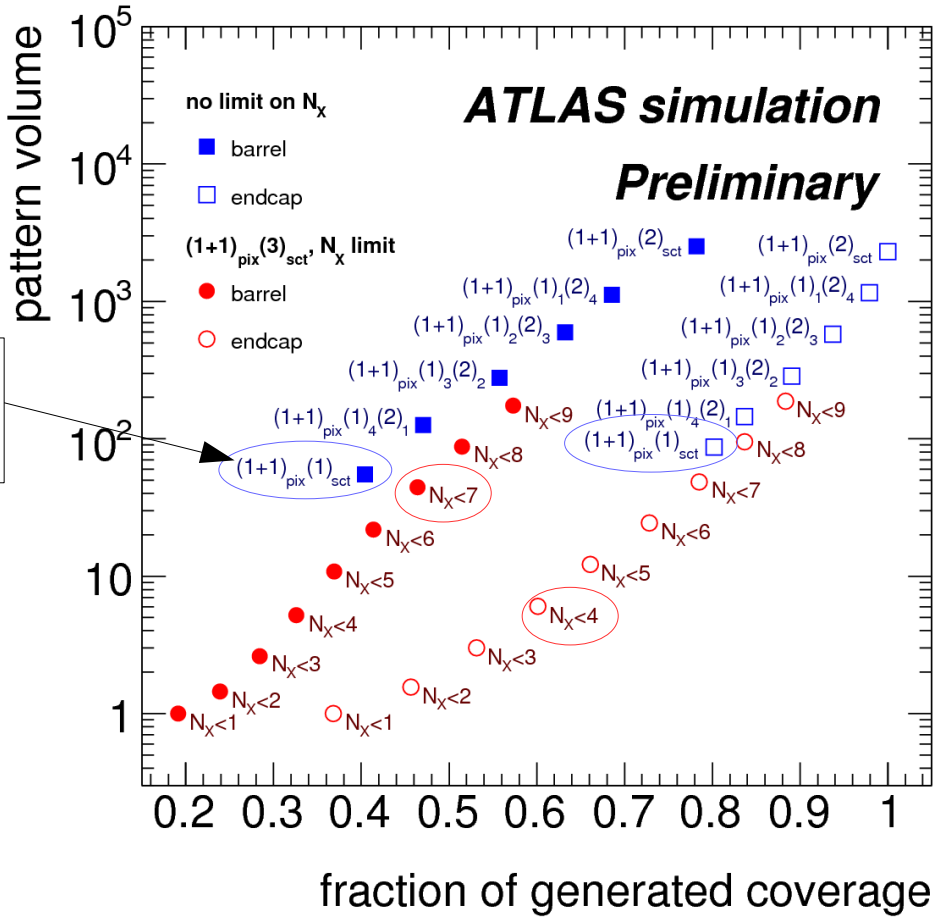



- Dataflow was adjusted using different limits on $N_x$ for barrel and endcap
- This example: same dataflow achieved at 15% higher pileup μ

# Survey of configurations

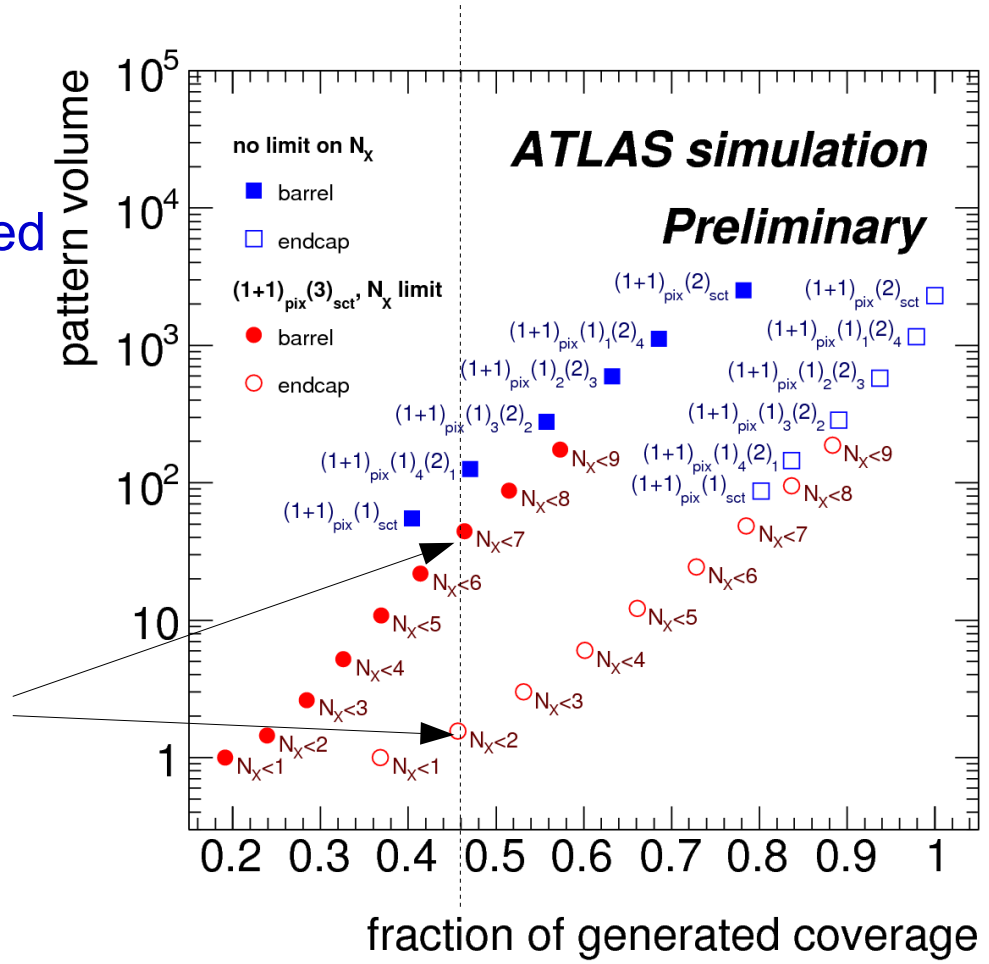
- Shown here: survey of various settings
  - blue: basic algorithm, vary number of ternary bits for sct layers
  - Red: refined algorithm, vary limits on $N_X$
- Variables studied:
  - Pattern volume $2^{N_X}$
  - Fraction of generated coverage (used fraction of the $10^9$ generated tracks)

Settings studied on previous slides are indicated



# Survey of configurations

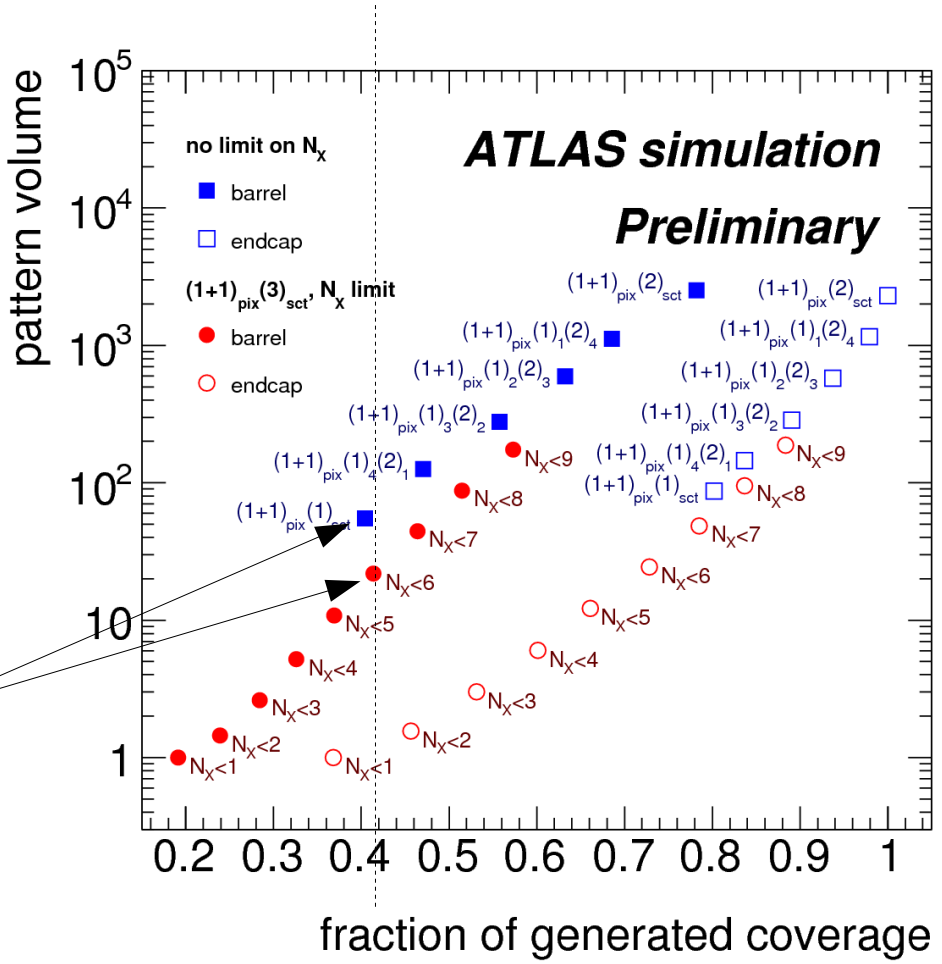
- Variables studied:
  - Pattern volume $2^{N_X}$
  - Fraction of generated coverage (used fraction of the $10^9$ generated tracks)
- High fraction ↔ high efficiency
- High volume ↔ high dataflow, random coincidences, fake tracks
- Endcap compared to barrel
  - Much smaller pattern volume in endcap for similar fraction
  - Compensates for higher hit density in endcap (more extreme polar angles)



# Survey of configurations

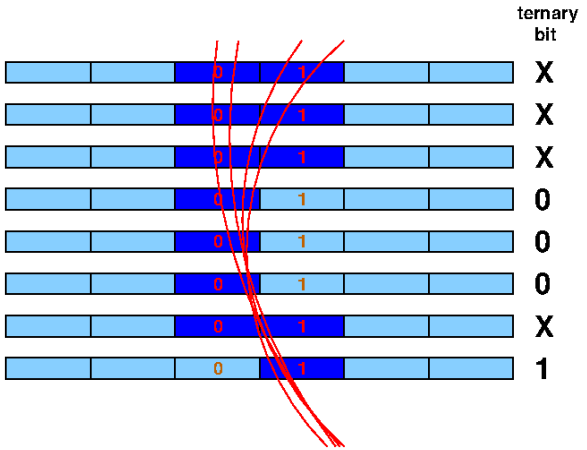
- Variables studied:
  - Pattern volume $2^{N_X}$
  - Fraction of generated coverage (used fraction of the $10^9$ generated tracks)
- High fraction ↔ high efficiency
- High volume ↔ high dataflow, random coincidences, fake tracks
- Basic algorithm compared to refined algorithm
  - For similar fraction, refined algorithm gives smaller volume

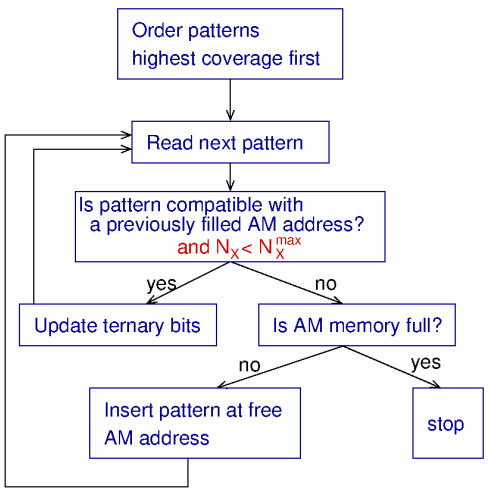
  →Refined algorithm performs better



# Summary

- The ATLAS FTK: a hardware track reconstruction system to provide the tracks with $P_T$>1 GeV for the high-level trigger
- Pattern recognition is based on the AM chip, which uses eight associative memories and ternary logic
- Two algorithms to define patterns for use in the AM chip are compared
- The algorithm which limits the number of ternary bits in the ternary status “X” per pattern performs best without adding significant computational cost



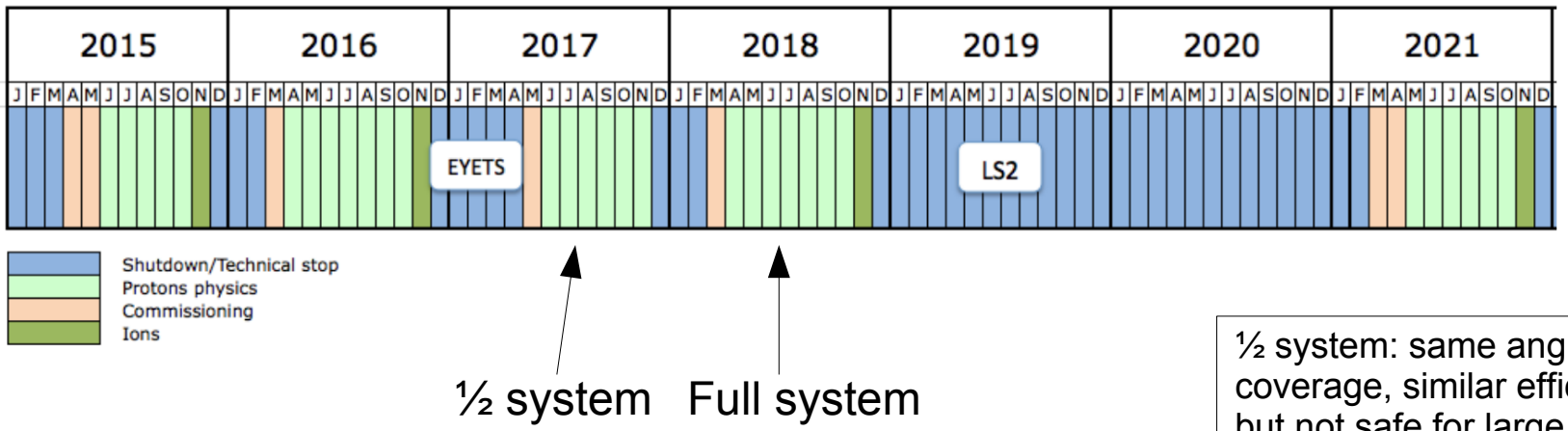




Figures and further material about the FTK project:
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/FTKPublicResults

# Backup

# FTK hardware status

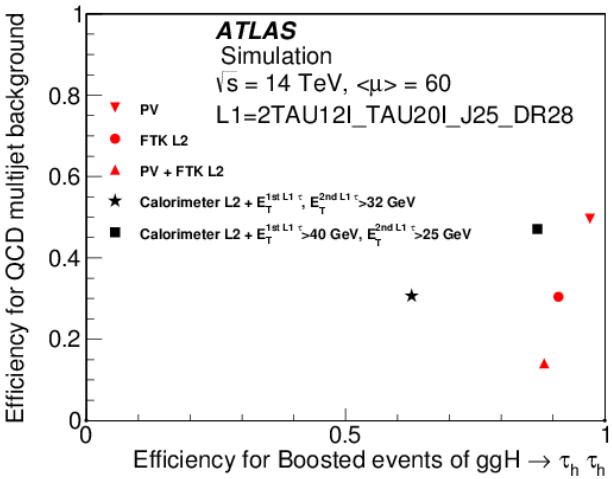
- Present status: hardware being commissioned and installed in ATLAS
- 2017: install ½ of the full system, operate FTK as ATLAS subsystem
- 2018: install full system




½ system: same angular coverage, similar efficency, but not safe for large pileup

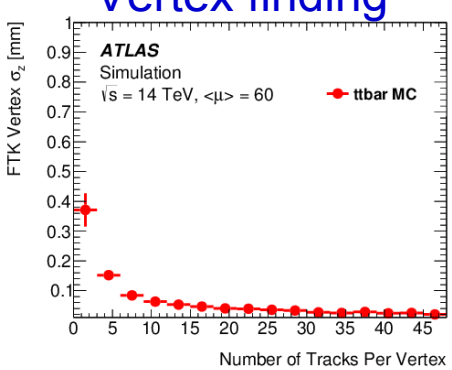
# FTK physics performance

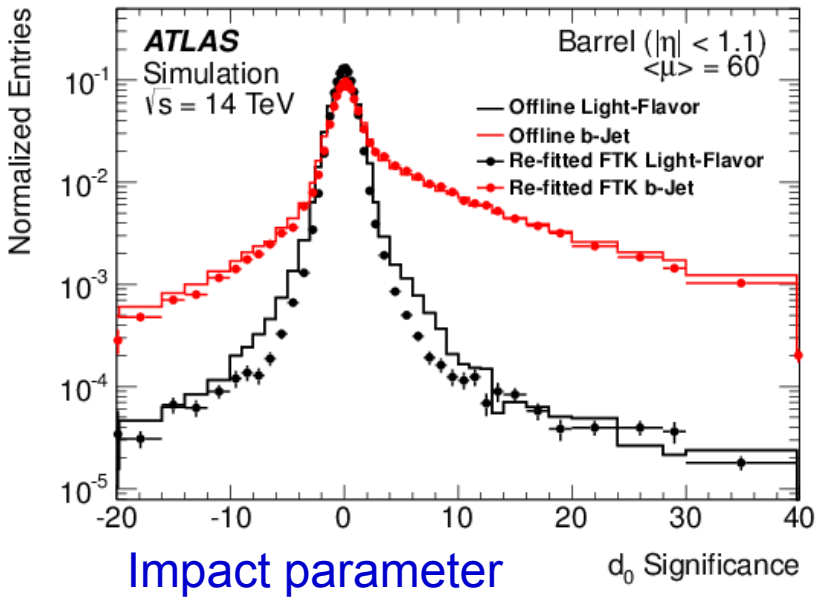
- Examples taken from the technical design report




Tau-lepton identification

Vertex finding

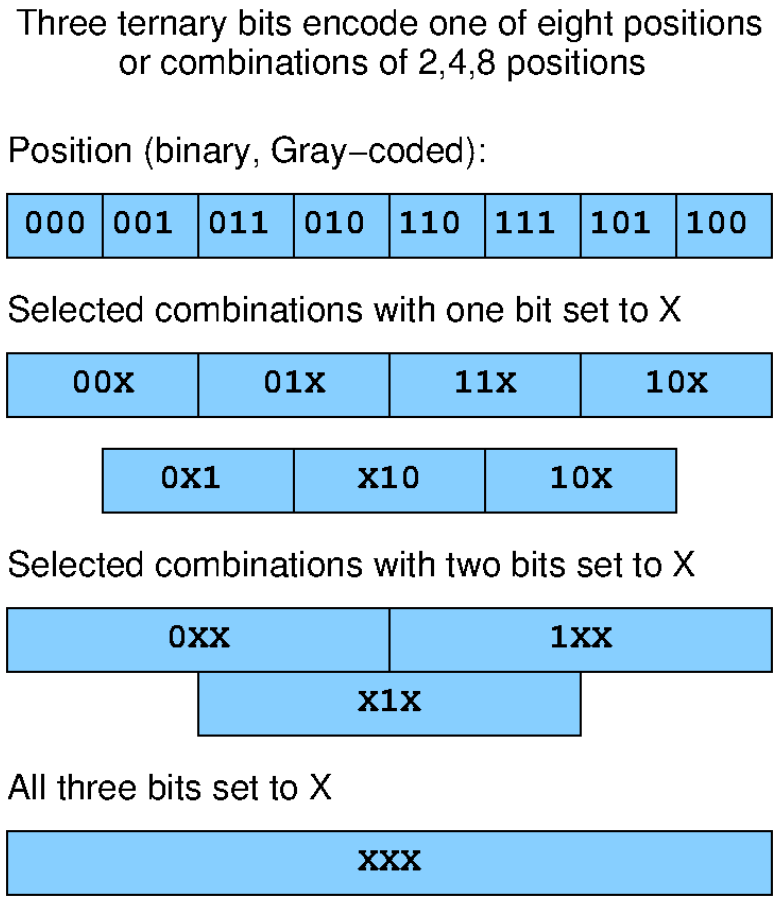



Impact parameter significance: b-tagging

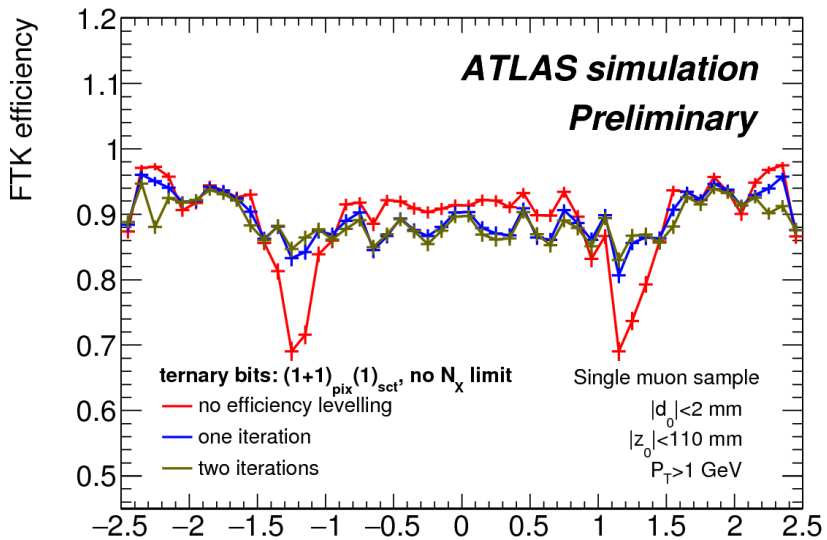
# Ternary bits and Gray code

- For each AM chip address: hits are stored in eight layers, one 15-bit word per layer
- Of the 15 bits, 3 bits are ternary
- Ternary bit can encode three settings: 0, 1, X = {0 or 1}
- Ternary bits → many patterns can be saved in one address
- Shown here: a selection of superstrip sizes and positions which can be encoded using 3 ternary bits



# Efficiency levelling

- Efficiency loss near |η|=1.2 is related to a change of the sensor orientation from “barrel” to “disk”
- Efficiency is recovered by repeating the assignment of patterns to AM addresses in bins of η
- For low-efficiency bins, a larger budget of AM addresses is assigned, for high-efficiency bins, a smaller budget is assigned (overall number of addresses is kept constant)
- The procedure gives a sizable improvement after one iteration, further iterations do not gain much



# Duplicate pattern

- Basic algorithm does not create duplicates: generated patterns are packed into a single AM address, possibly producing large pattern volumes
- Refined algorithm starts a new AM address if the limit in $N_X$ is reached →overlaps (duplicate encoded patterns) may be created
- Figure: for each encoded pattern (each combination allowed by ternary bits), count the multiplicity M.
- Duplicates (M>1) are present but at a moderate level
- Note: the condition to accept patterns with 7 of 8 possible hits also creates duplicate roads.