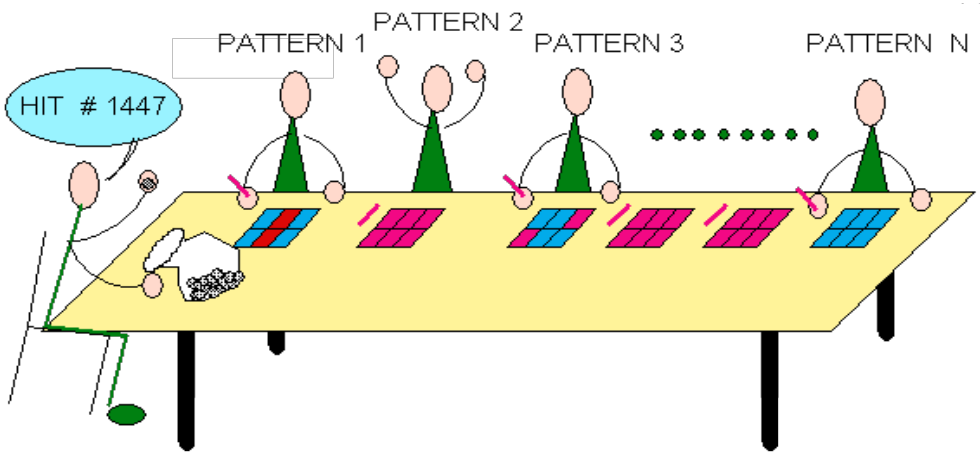
ATLAS FTK challenge: simulation of a billion-fold hardware parallelism



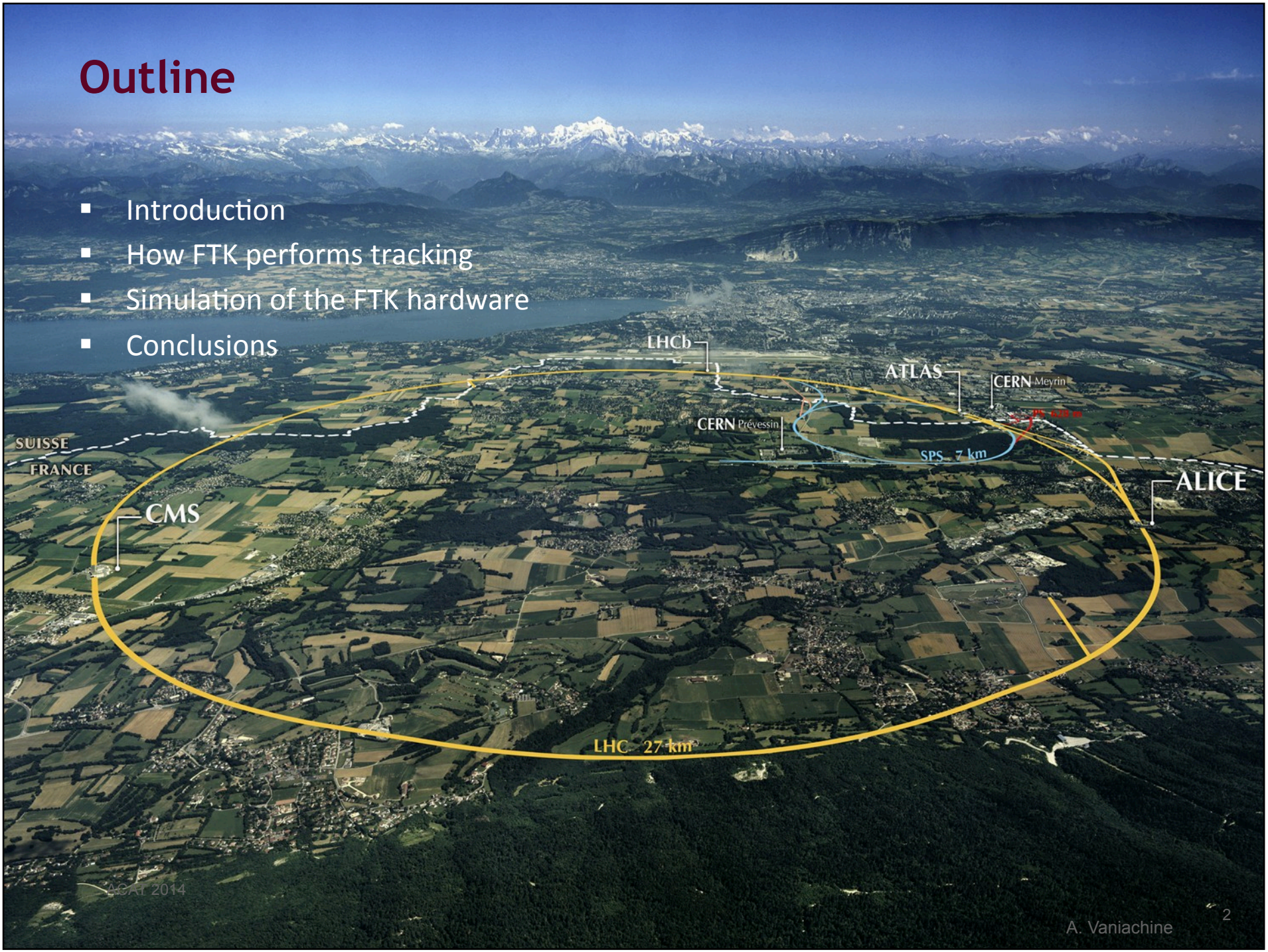
A. Vaniachine on behalf of the ATLAS Collaboration

16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT 2014)

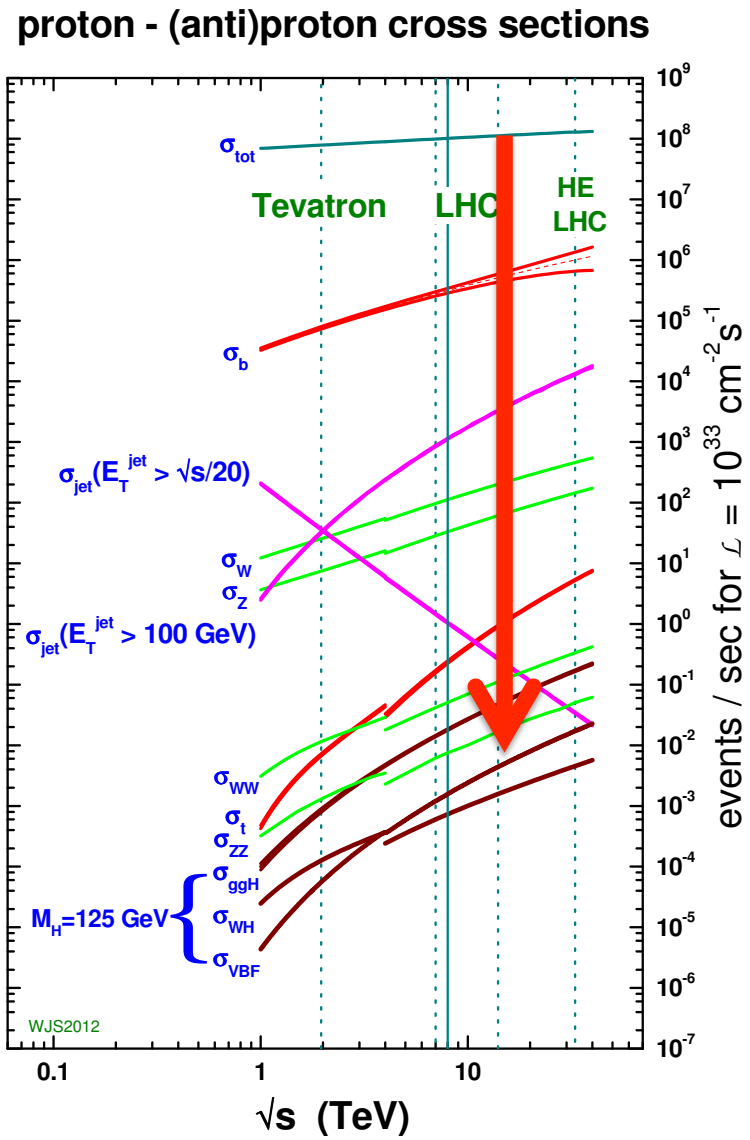
September 1-5, 2014, Prague, Czech Republic

Outline

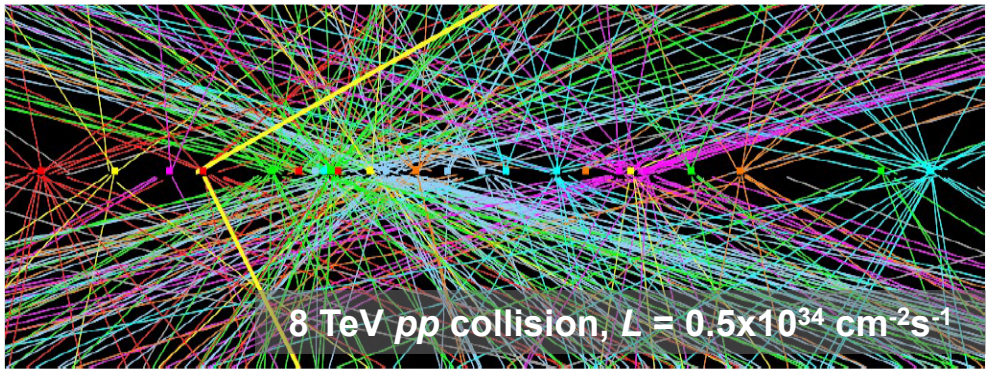
- Introduction
- How FTK performs tracking
- Simulation of the FTK hardware
- Conclusions



LHC Run 2: the Motivation and the Challenge



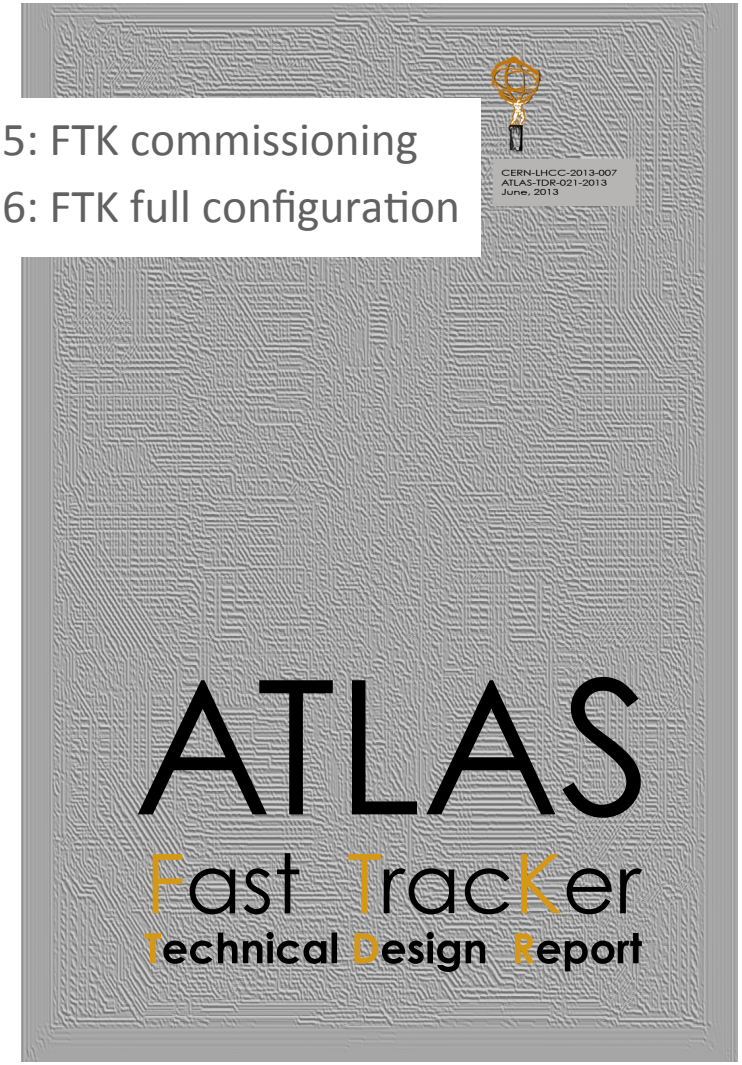
- In 2015 the LHC Run 2 will reach 13-14 TeV
- High precision tests of the Standard Model
  - Discoveries require rejection of "known" events by more than ten orders of magnitude
    - ATLAS will have two-tier trigger system: **L1** and **HLT**
- Run 2 *L* will reach 2×10^{34} cm^{-2}s^{-1} and beyond
  - Trigger rejection complicated by the presence of a large number of minimum-bias events - pileup
    - **Full tracking is a powerful tool to limit its effects**
- Run 1 approach - High Level Trigger (HLT) tracking will be difficult and time consuming
  - Particles from interesting processes (such as $Z\rightarrow\mu\mu$) are buried in a high particle density:



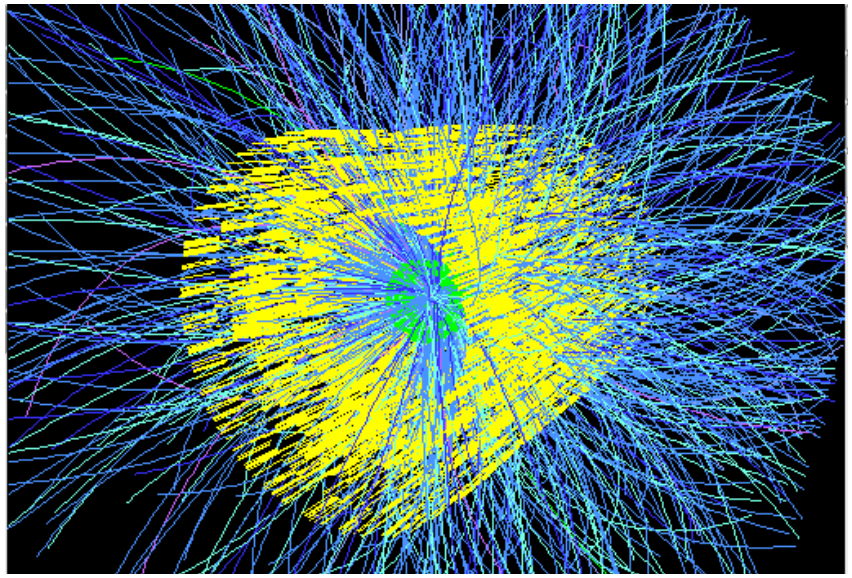
ATLAS Trigger in Run 2 Will Use FTK: Fast TracKer

2015: FTK commissioning

2016: FTK full configuration



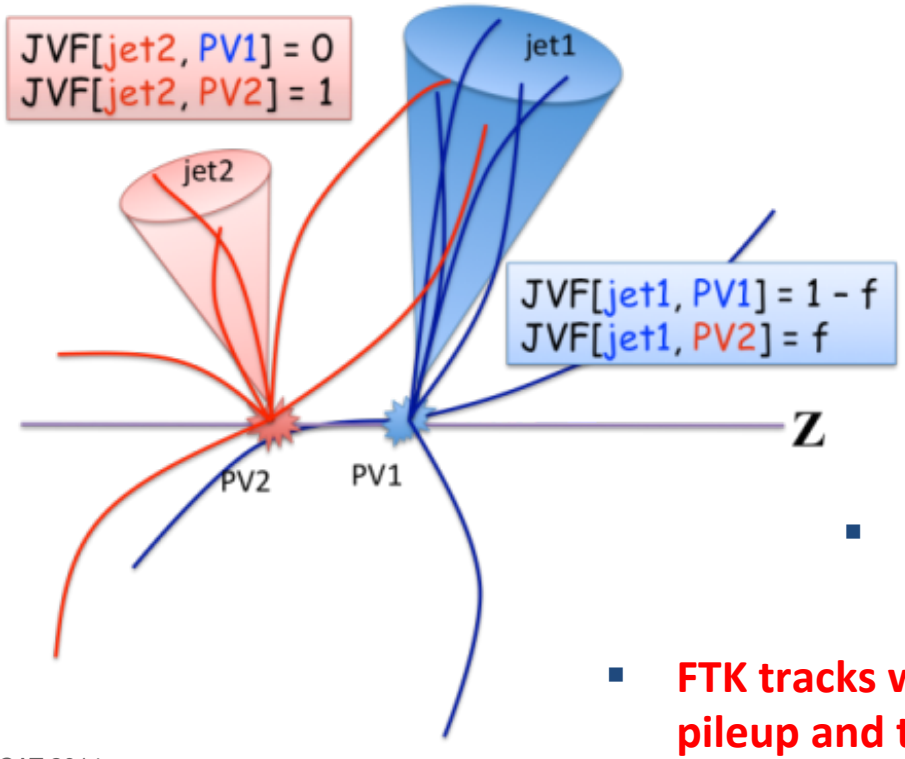
- Fast tracking is implemented in hardware with custom electronics
  - Between the L1 trigger and HLT
    - **Provides full tracking for all events passing L1**

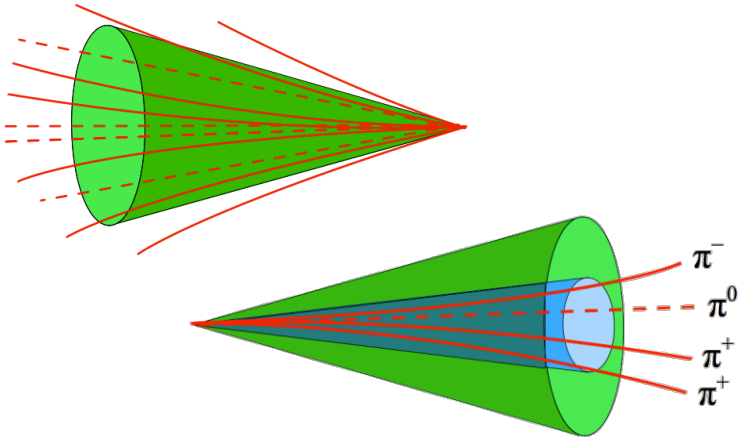


  - Improves trigger performance in many areas
    - e.g. *b*-jet or τ triggers

Role of Fast Tracking

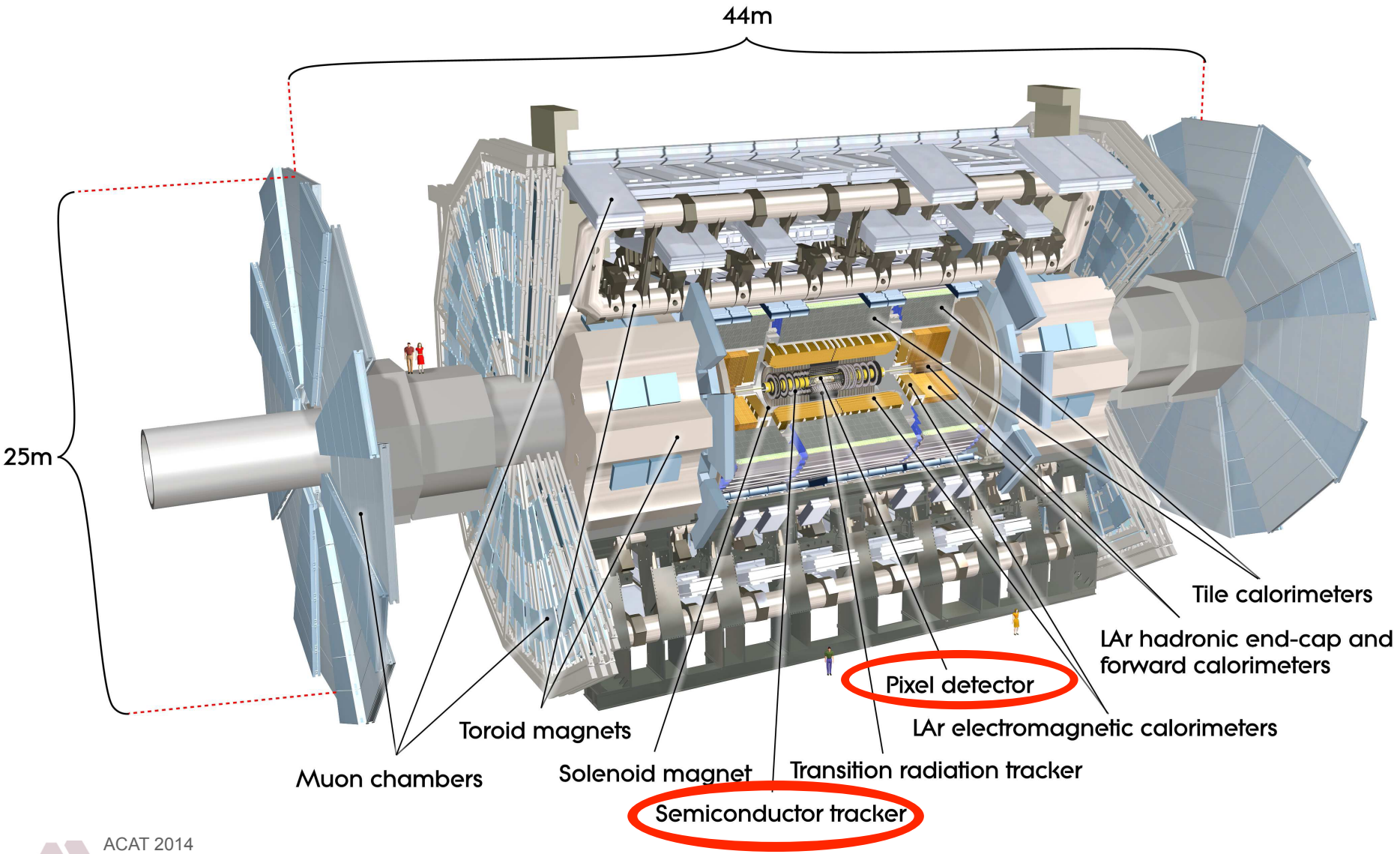
- Tracking allows to separate different collisions and the originated objects
  - Specific topologies (i.e. *b*-jet or τ) can be efficiently identified with high quality track reconstruction



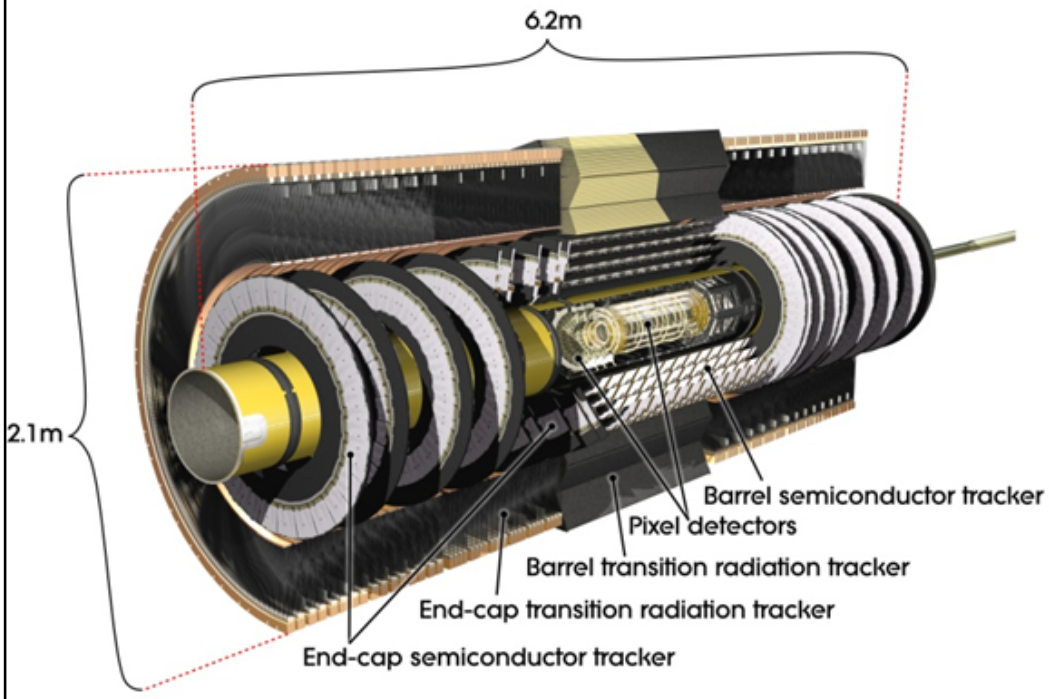


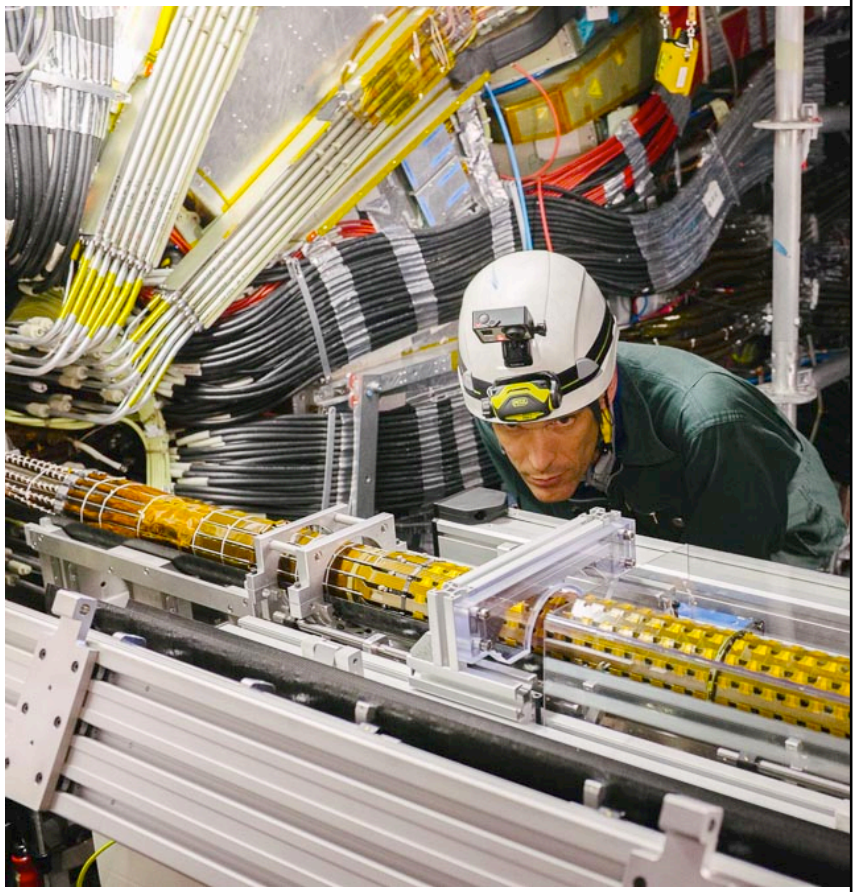
- Beyond these examples, FTK tracks will be widely used in trigger event selection
- **FTK tracks will be one of trigger's best tools against pileup and the prohibitive time of full scan tracking**

For Fast Tracking ATLAS Uses Silicon Detectors



Pixel Detector is Upgraded for LHC Run 2

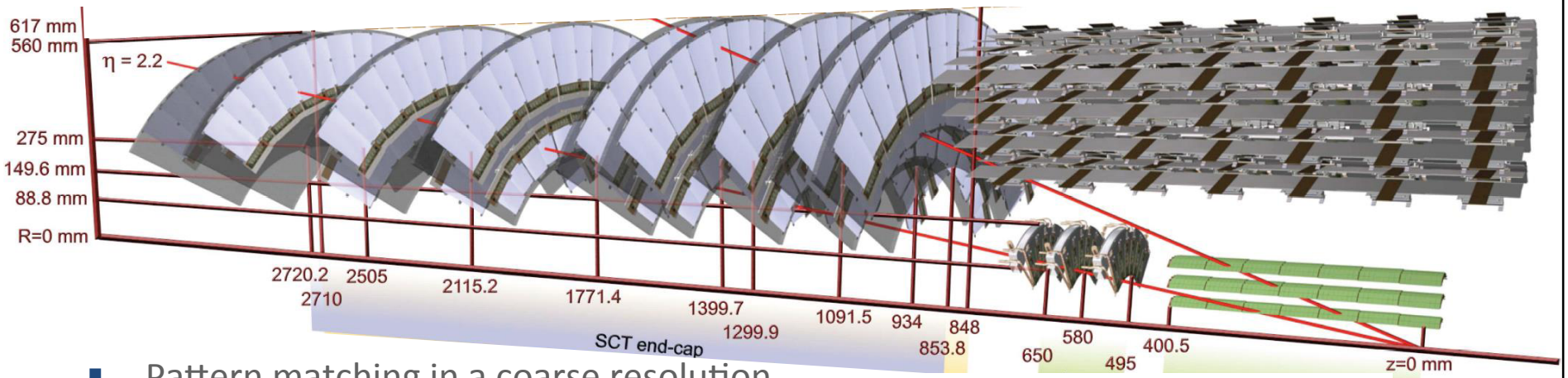




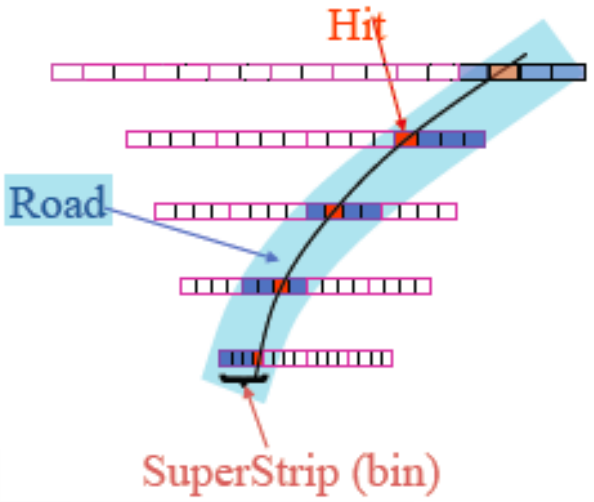
- Another solution to tracking problems in the high-luminosity environment is the Insertable B-Layer
  - That brings the total number of layers in silicon detectors to twelve

FTK Finds Tracks via Match and Fit

- FTK hardware performs global tracking in two steps: pattern matching and track fitting
  - Pattern matching and initial track fitting use 3 pixel, 4 SCT axial and 1 SCT stereo layers

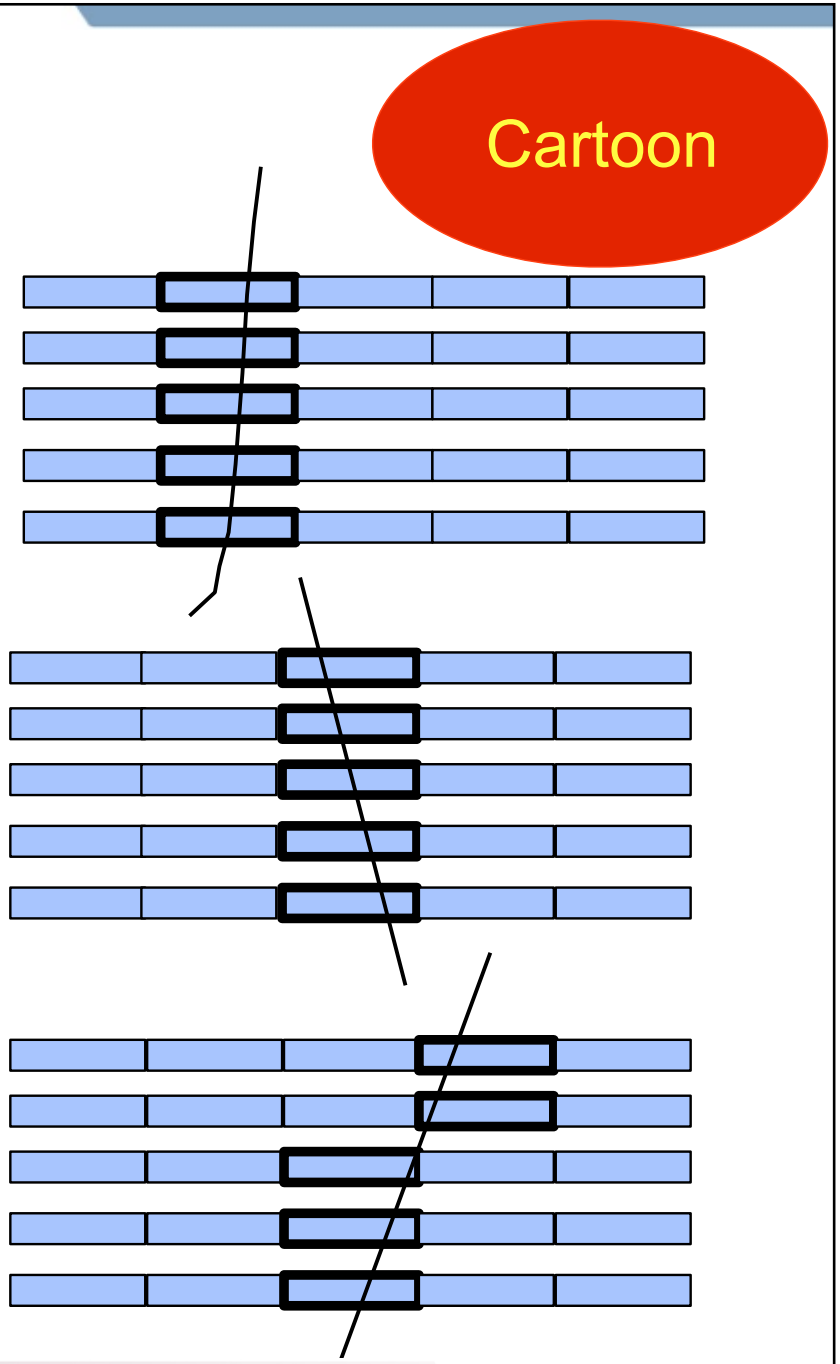


- Pattern matching in a coarse resolution
  - Joining adjacent pixels or SCT strips in coarser bins - SuperStrips
- Track fit in full resolution
  - Hits in a "road" - pattern of SuperStrips
- Road size optimized to balance workload between the two steps
  - pattern bank size vs. fit combinatorials



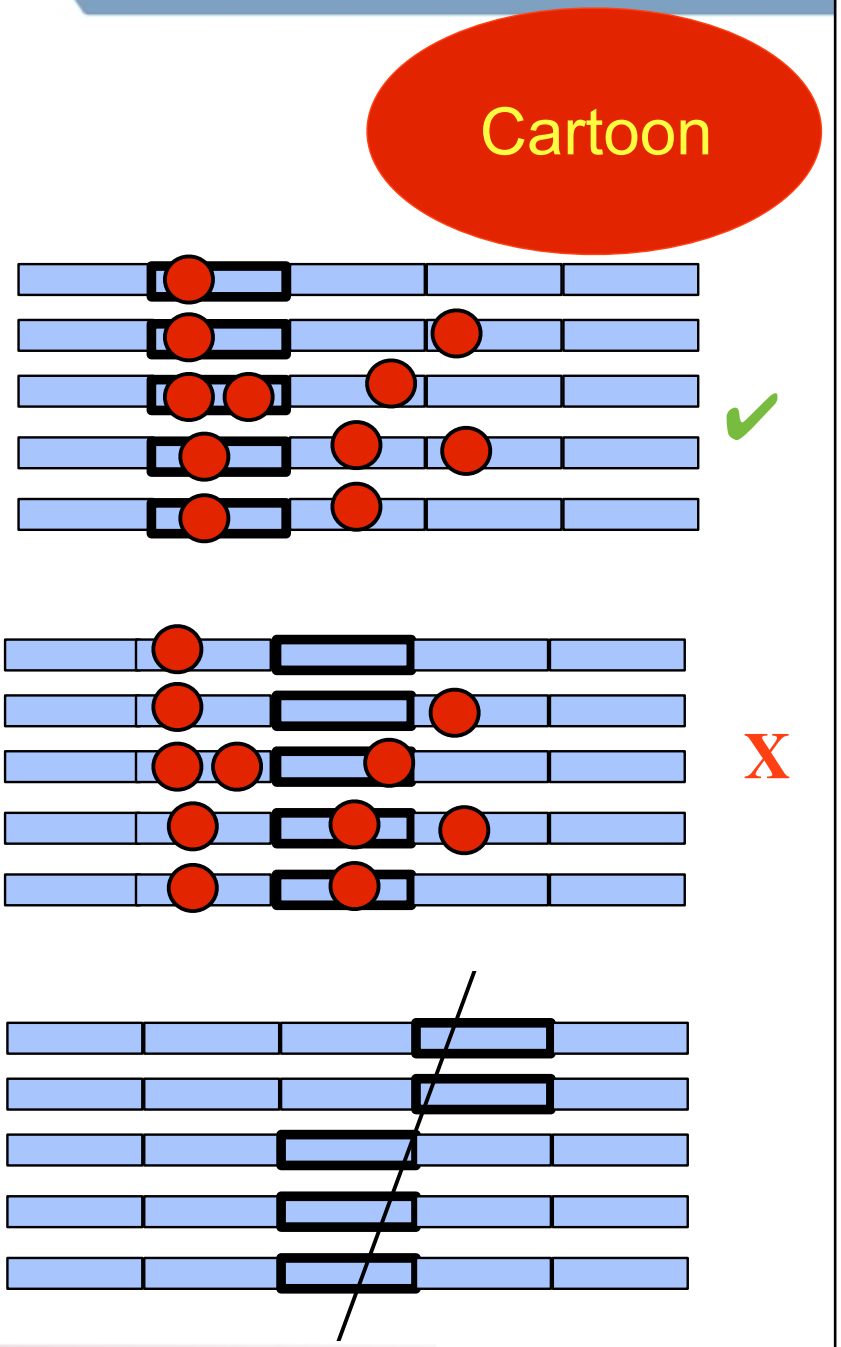
Pattern Matching

- The pattern matching uses **custom** Associative Memory (AM) ASICs pre-loaded with 1B patterns
  - All 1B patterns are matched in parallel

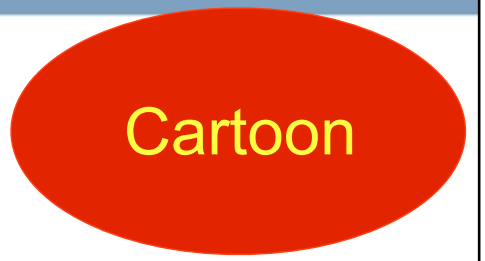


Pattern Matching

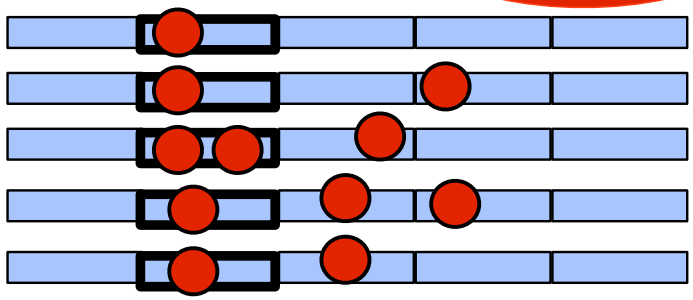
- The pattern matching uses **custom** Associative Memory (AM) ASICs pre-loaded with 1B patterns
  - All 1B patterns are matched in parallel
- Matching pattern IDs and the hits sent to the next stage
- Hits not matching a pattern are not passed to the next stage



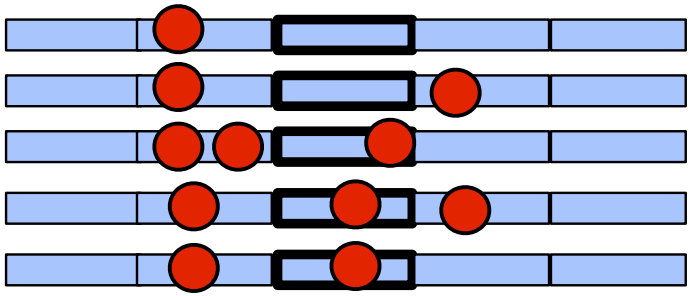
Pattern Matching



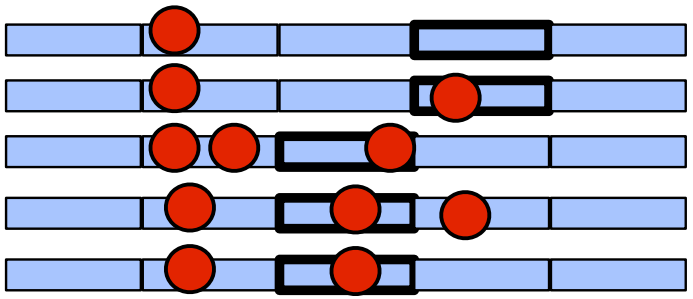
- The pattern matching uses **custom** Associative Memory (AM) ASICs pre-loaded with 1B patterns
  - All 1B patterns are matched in parallel



- Matching pattern IDs and the hits sent to the next stage
- Hits not matching a pattern are not passed to the next stage



- Patterns matching all but one layer can be flagged
  - Called the Don't Care bit
    - This feature balances the high-resolution vs. fake tracks rejection (see extra slide)

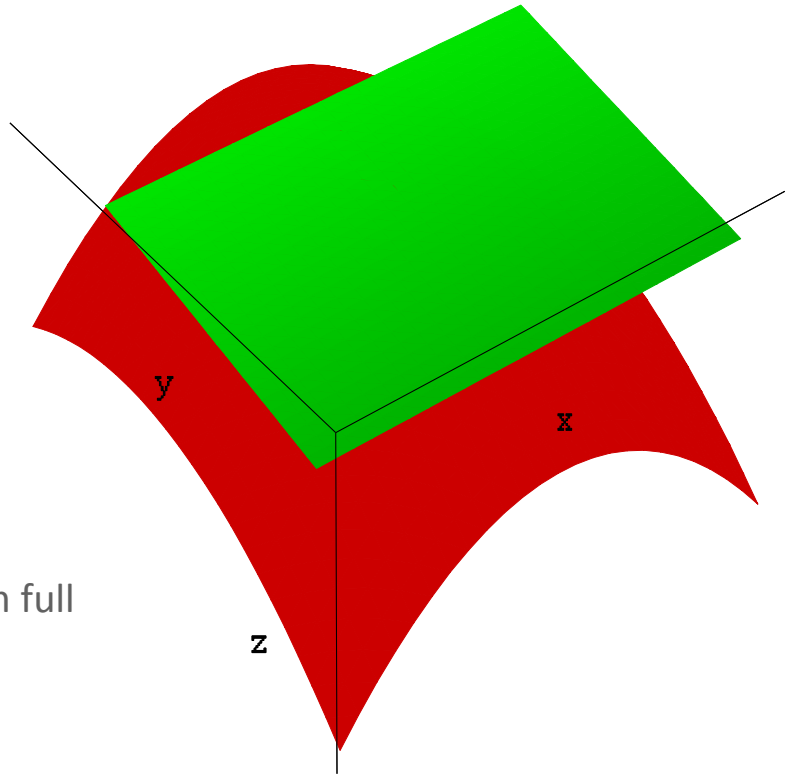


Track Fitting

- Determine the helix parameters and χ^2
- Linear fit one module in each layer

$$p_i = \sum_{j=1}^{14} a_{ij} x_j + b_i$$

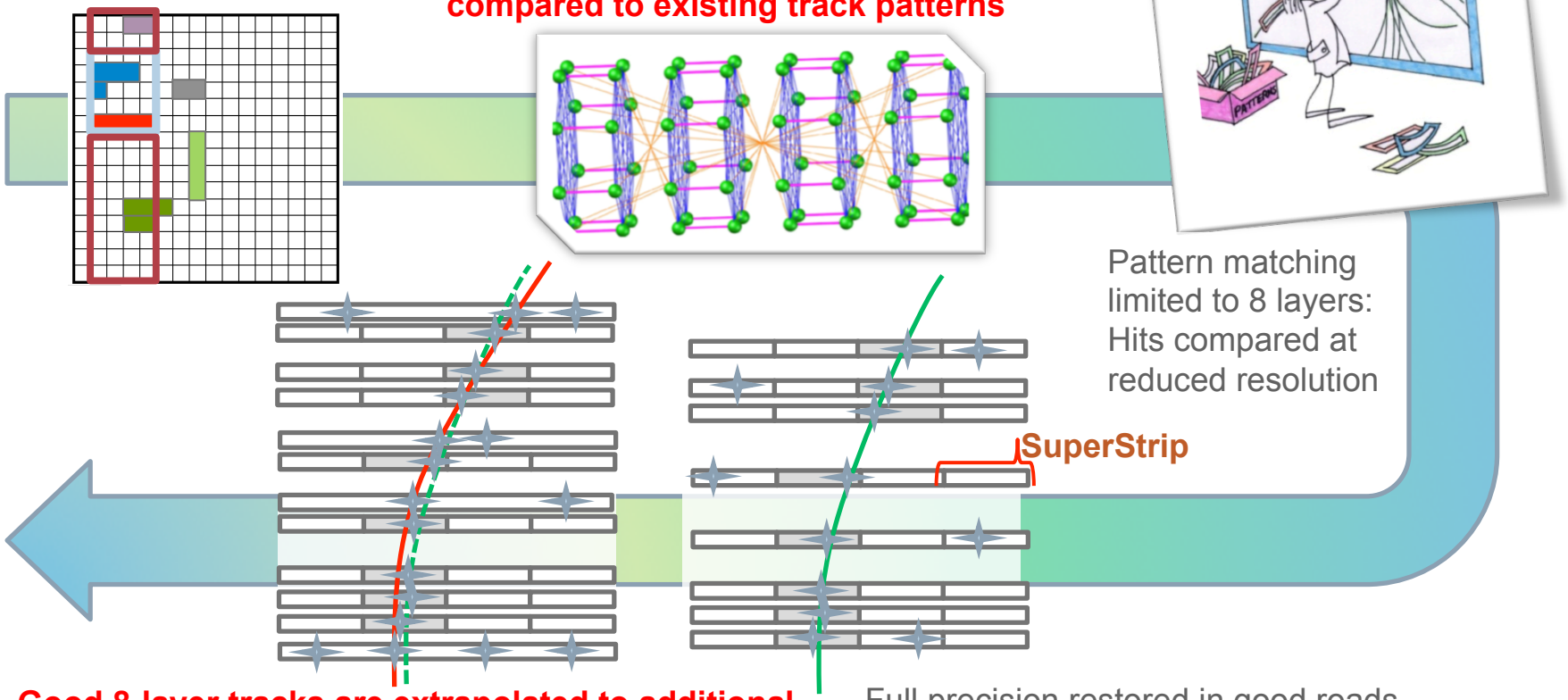
  - P_i: the helix parameters and χ^2 components
  - x_j: the hit coordinates in the silicon layers
  - a_{ij} and b_i: prestored constants determined from full simulation or real data track
- Very fast in FPGAs (~1 ns per track)



In Addition to Match and Fit, the Entire FTK Data Processing Pipeline Has a Few More Steps

FTK has a custom clustering algorithm, running on input FPGAs

The data is geometrically distributed to the processing units to be compared to existing track patterns



Pattern matching limited to 8 layers: Hits compared at reduced resolution

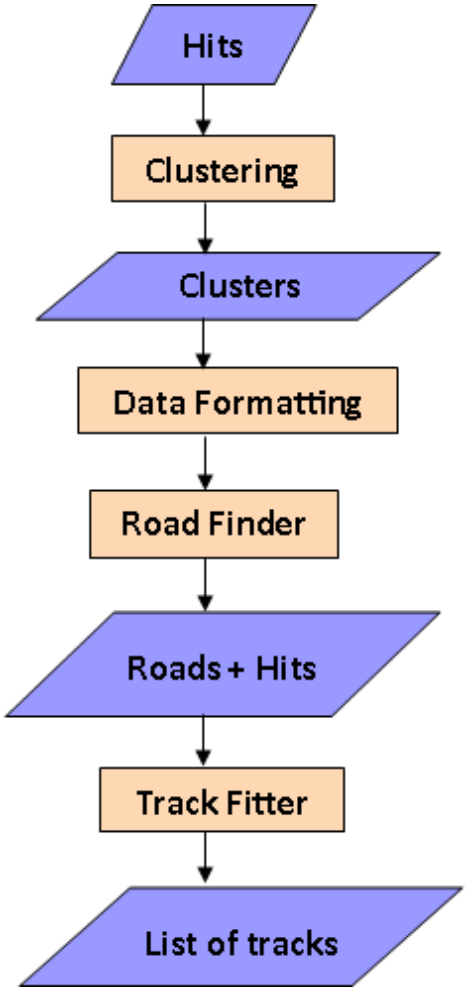
SuperStrip

Good 8-layer tracks are extrapolated to additional layers, improving the fit and removing duplicates

Full precision restored in good roads
Fits reduced to scalar products

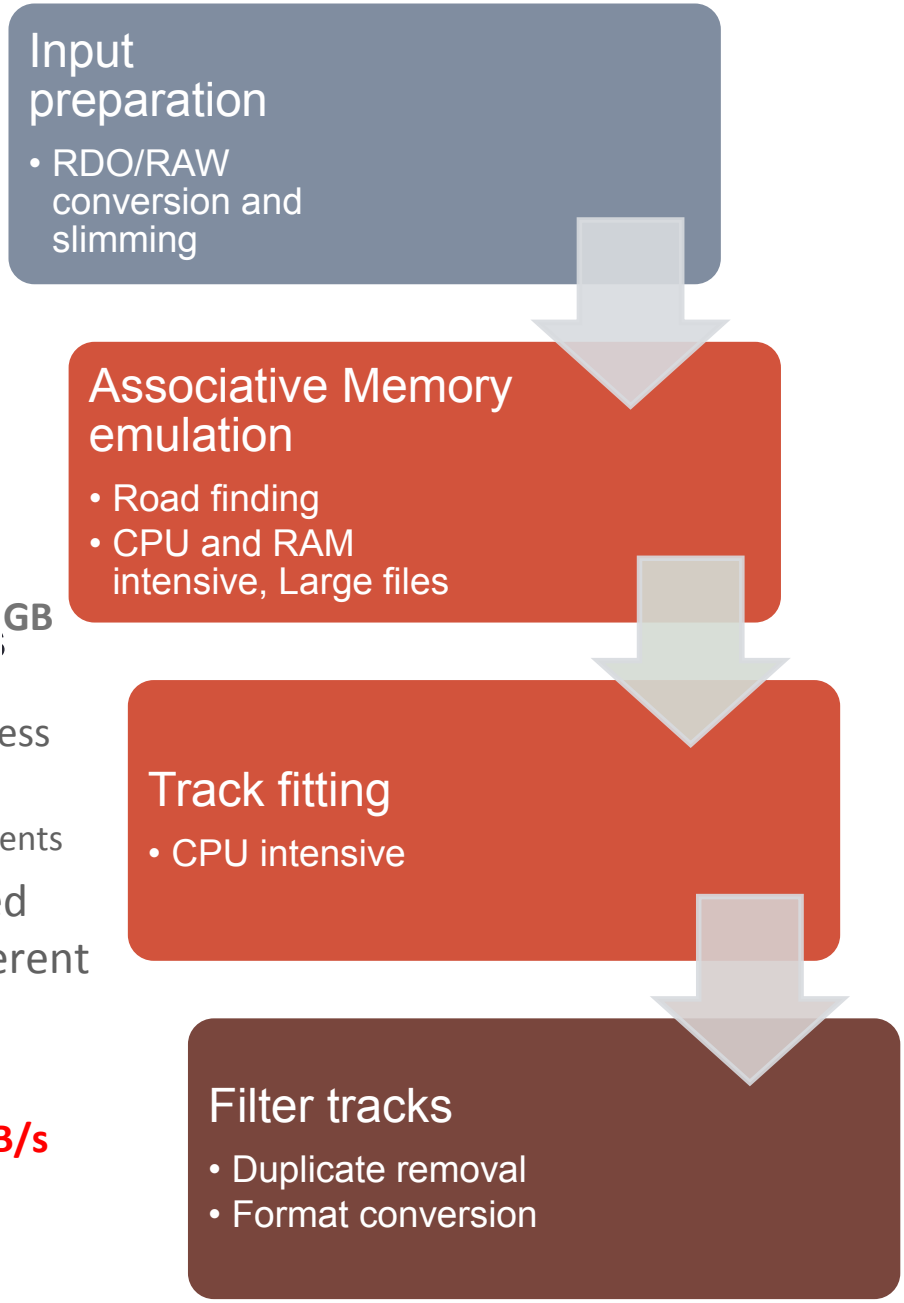
FTK Simulation Goals and Structure

- The entire FTK data processing pipeline has to be simulated in preparation for LHC upgrades, as we need to:
  - Develop trigger strategies at high luminosity
    - Calculate trigger efficiencies for precision physics studies
  - Support hardware design/finalization
- These two goals provide complementary requirements:
  - Physics performance studies require many millions of events
    - Efficient use of CPU is important
  - Design requires lesser data sample, but accurate emulation
- To achieve both, the FTK simulation mimics the hardware:
  - All the internal steps are reproduced
    - with the possibility to change all internal parameters to evaluate carefully the impact of firmware or boards' designs
  - Detector hits can be read from MC events or from real data

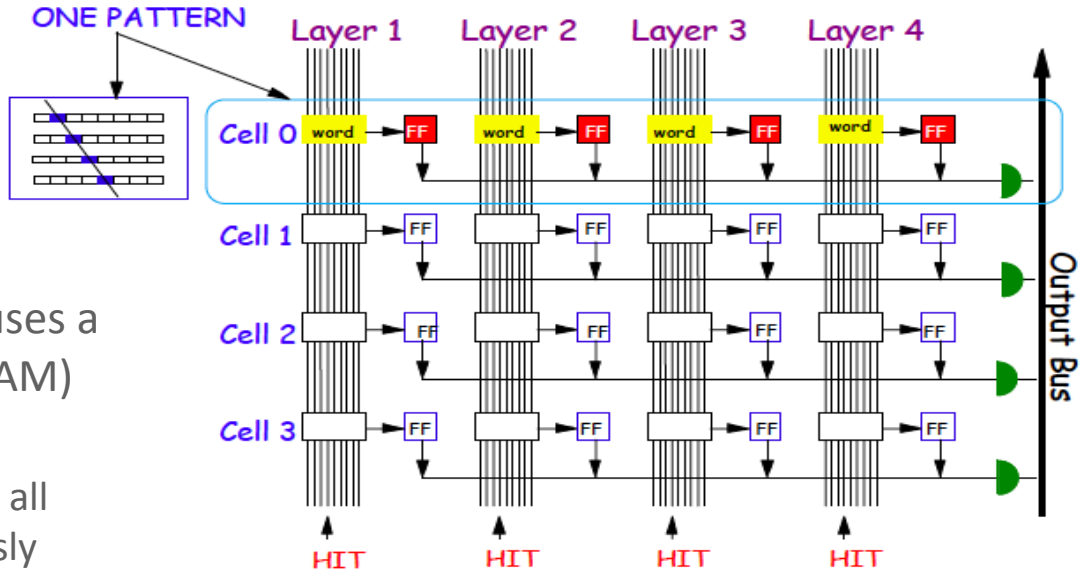


FTK Simulation Challenge

- The use of algorithms tailored for a large electronics system is resource intensive
- Memory size and access are the main challenges
  - FTK has **one billion patterns** that require about **35 GB** of memory
  - **Two million fit constants** require about **2 GB**
    - Plus smaller configuration data
  - Commissioning FTK configuration will be less demanding (0.25B patterns)
    - But still with significant memory requirements
- The simulation of such a highly parallelized system on general-purpose CPUs has inherent performance bottlenecks:
  - Limited parallelism
  - Low bandwidth access to memory: ~25 **GB/s**
    - In comparison, the AM system has the I/O bandwidth of about 25 **TB/s**



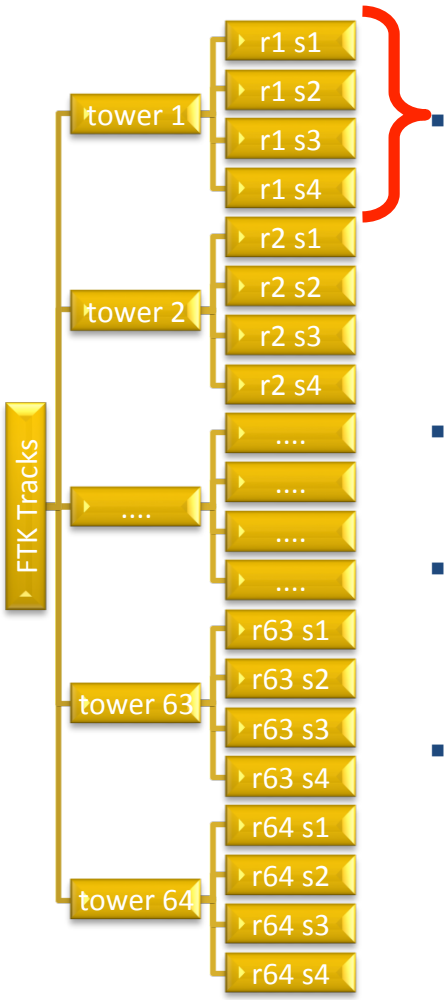
FTK Hardware Implements a Billion-Fold Parallelism



- In contrast with simulations, the Associative Memory (AM) ASIC uses a content addressable memory (CAM) architecture:
  - Any data inquiry is broadcast to all memory elements simultaneously
    - Data retrieval time is independent of patterns size
    - System performs millions of comparison per ASIC per second
- Each incoming hit reaches all one billion patterns in the whole FTK AM system within the same clock cycle (10 ns)
  - **This is a feature that cannot be matched by a system with general-purpose CPUs**
- Similar contrast with simulations on general-purpose CPUs happens with the track fitter hardware implementation
  - **No CPU can reach the 1 GHz rate obtained in the FPGAs on the FTK fit boards**

Simulating a Massively Parallel Tracking Processor

- Practical constraints with Grid Computing
  - Single-threaded and/or single-process
  - Real memory limited to 2 GB per CPU-core
  - Job duration is limited to several hours per hundred of events
  - Not more than several GB of configuration data per job
- It is natural to split the FTK simulations into parallel jobs
  - Configuration files for each job split into 256 η-ϕ subregions
    - **Each event is processed 256 times, matching a different subregion of the event with corresponding configuration subregion**
  - Processing requires subregions merging, practically organized into two steps
- Single jobs can be sequentially executed on a single node or multiple nodes
  - Usually distributed concurrently to different Grid nodes

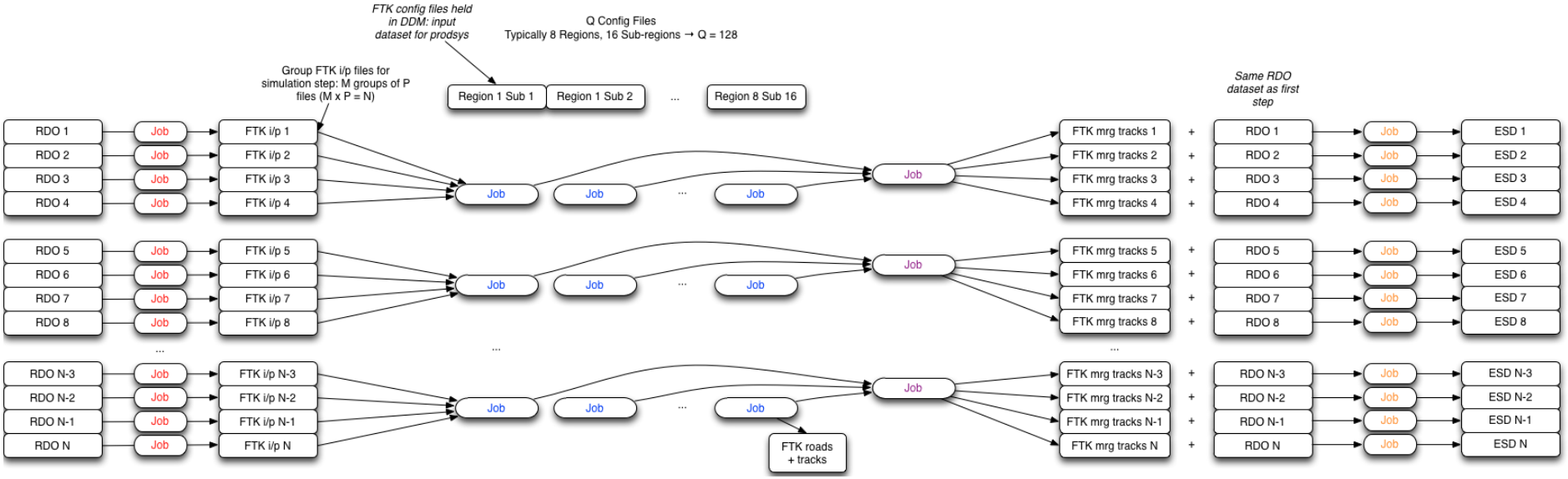


- **Each of 64 towers is split in 4 subregions**
  - Reducing the resources required
  - Increasing parallelism
- Memory required for the single core is reduced to 1 GB
- Input events' data is organized in towers reducing the time to retrieve the data
- **As a result, the FTK simulation workflow requires four steps**
  - See next slide

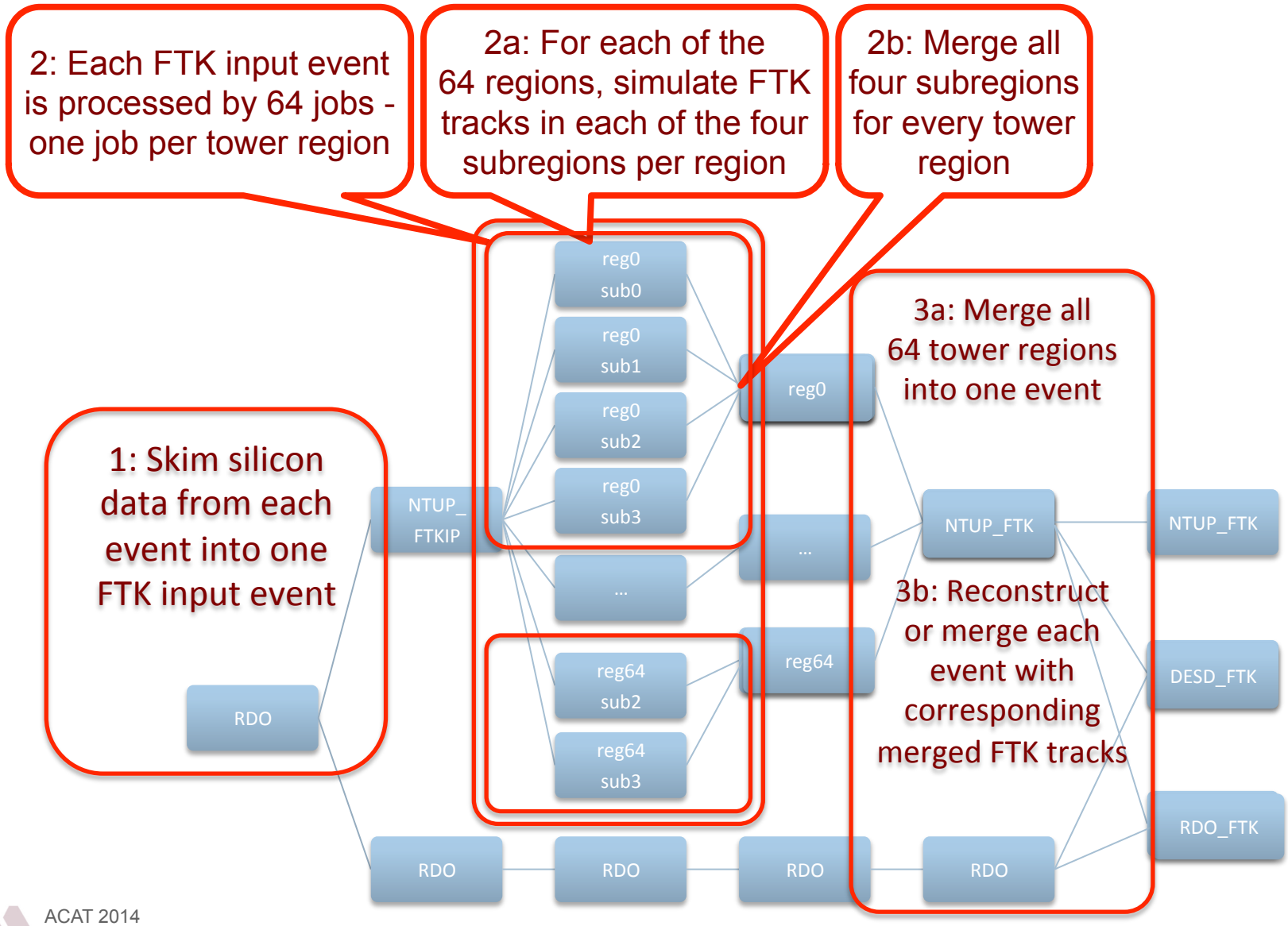
General FTK Simulation Workflow Has Four Steps

1. Input data preparation (extract silicon detector hits)
2. Parallel pattern matching and track fitting in each of the event subregions
3. Merging FTK tracks found in each of the subregions into the whole event
4. Trigger simulation (with FTK tracks) and reconstruction

- Corresponding jobs are shown below in matching colors
  - No need to read other ATLAS-specific details of the FTK simulations workflow

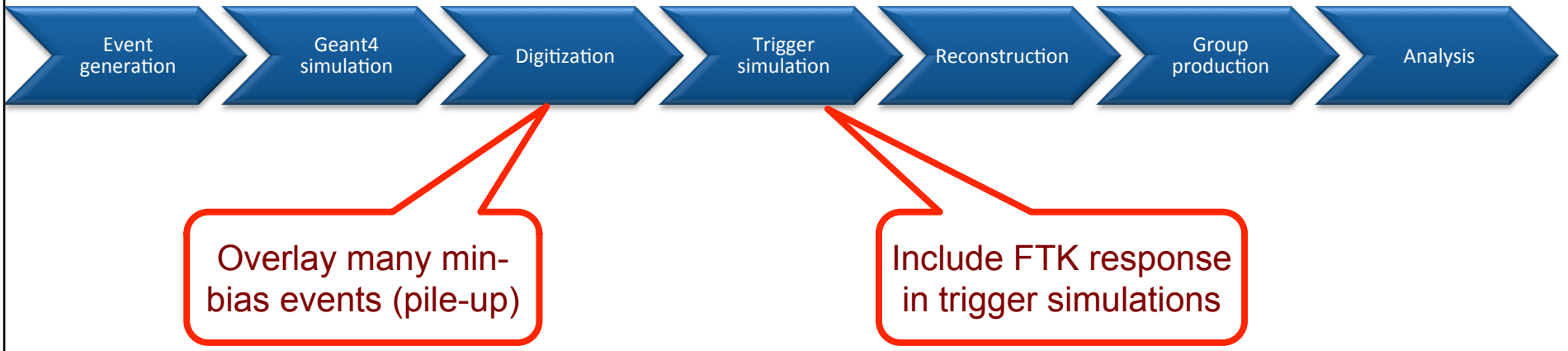


In Practice Workflow Was Reduced to Three Steps



Production System Workflow: Simulations

- The FTK workflow is integrated with Monte Carlo simulations at the Production System level above the ATLAS workload management system – PanDA



- A. Klimentov presents PanDA overview in the plenary talk on Tuesday:

  Next Generation Workload Management System for Big Data on Heterogeneous Distributed Computing

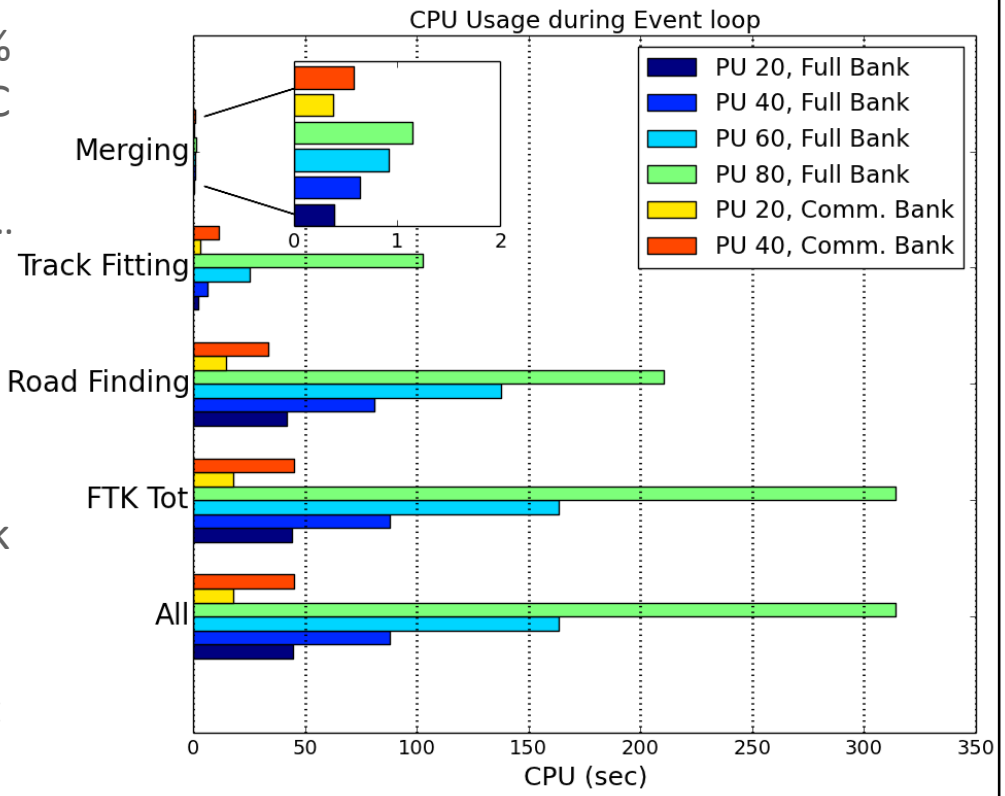
- Further details are presented in the posters:

  M. Borodin et al. ***Multilevel Workflow System in the ATLAS Experiment***, poster board 115

  J. Chapman et al. ***Challenges of the ATLAS Monte Carlo Production during Run-I***, board 108

FTK Simulation Performance

- FTK simulation adds from 15% to 30% of the CPU time for the ATLAS full MC simulation chain:
  - Geant4, digitization, reconstruction,…
- The required CPU resources change according to the FTK configuration
  - Full pattern bank: 1024M patterns
  - Commissioning bank: 256M patterns
- For the SLC6 VM with 1800 SPECint2k
  - Total execution time is up to 50 s/ev for the commissioning configuration
  - Full FTK configuration requires about 300 s/ev
- The execution time increases with increasing LHC luminosity
  - The dependence is close to linear, thanks to efficient pattern matching



- Pattern matching (Road Finding) time is dominated by AM simulation: ~2/3
  - The AM simulation time largely depends on the pattern bank size

Conclusions

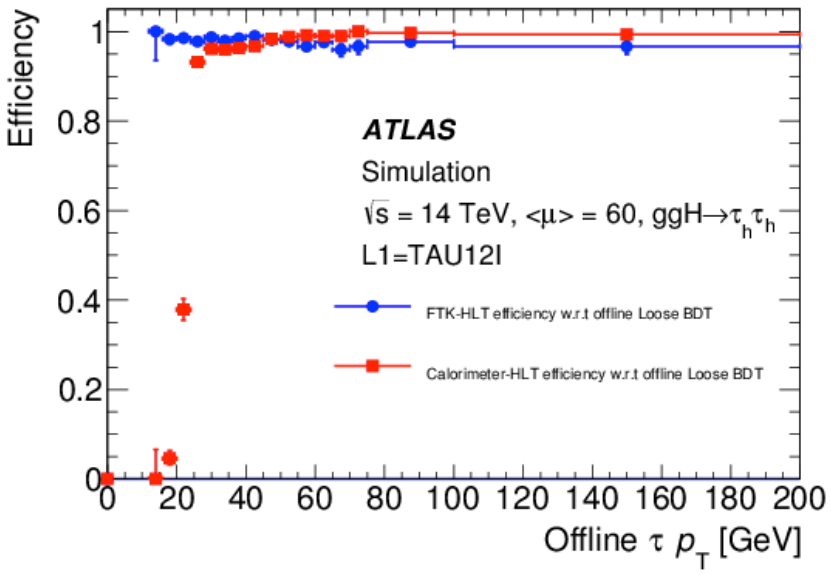
- **FTK is a fundamental upgrade of the ATLAS trigger and data acquisition system**
- The use of custom hardware based on AM ASICs and FPGAs provides unprecedented computing power
  - **Achieving a billion-fold parallelism - the "Holy Grail" of exascale computing quest**
- FTK simulation emerged as a foundation to support the use of FTK hardware
  - Continued FTK hardware integration with ATLAS HLT system are matched by proper changes in FTK performance studies
    - Demonstrating flexibility of the ATLAS Production System
- Current FTK simulations run quite smoothly in the ATLAS Production System
  - Dozens of samples totaling 9.6 M of fully simulated events were produced
- The next production of a multi-million event sample is under preparation

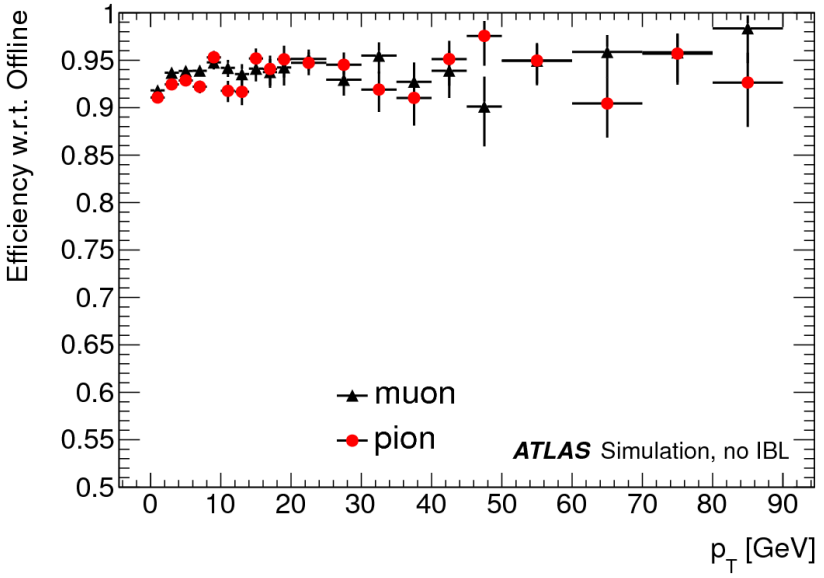
Argonne NATIONAL LABORATORY

Extra Slides

U.S. DEPARTMENT OF ENERGY

FTK Simulation Results: Expected Performance

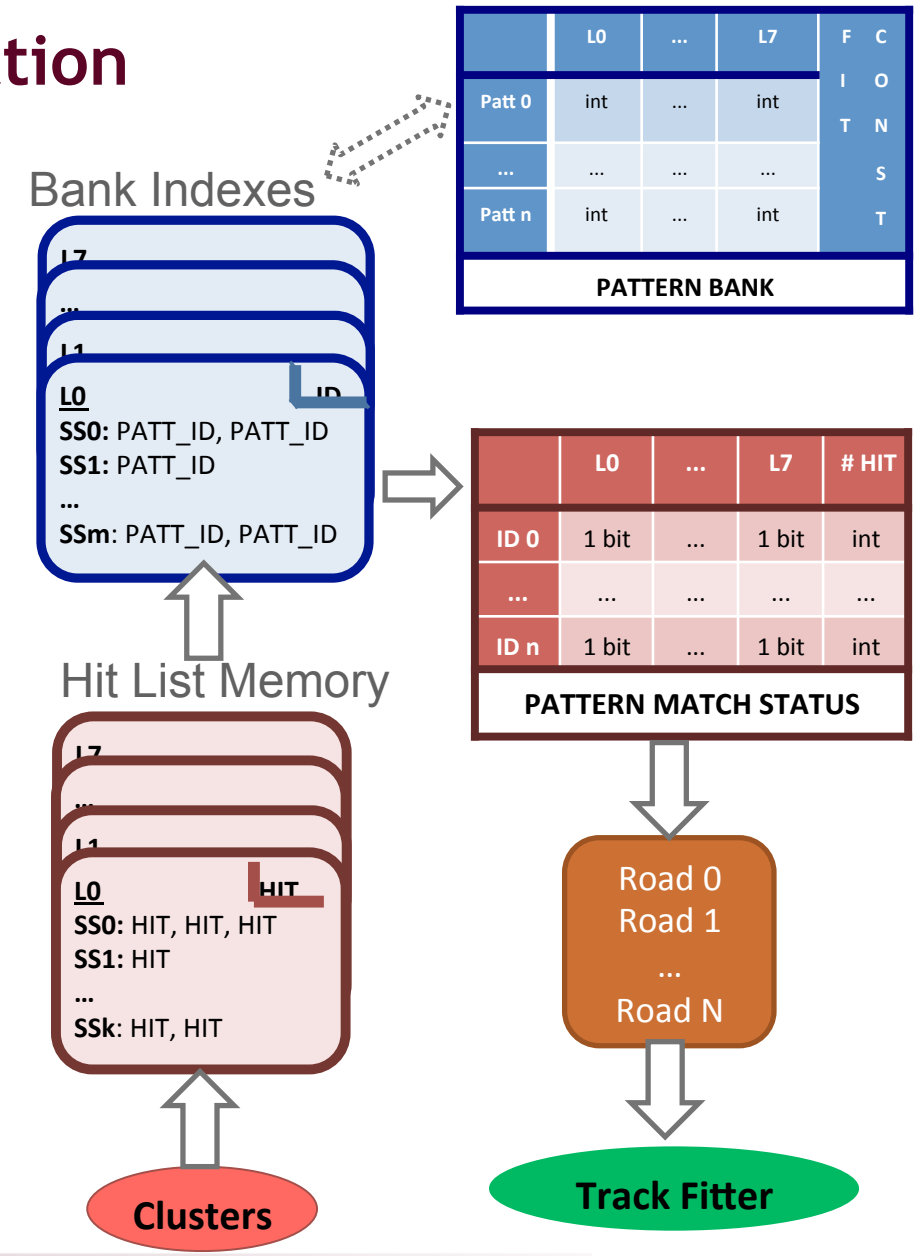




- The simulation allowed to evaluate the quality and efficiency of the tracks obtained from FTK
  - Track quality and efficiency is close to state-of-the-art algorithms, based on CPU and much larger latencies
    - The overall quality is verified in the use of tracks found by FTK in complex identification algorithms, such as b-tagging

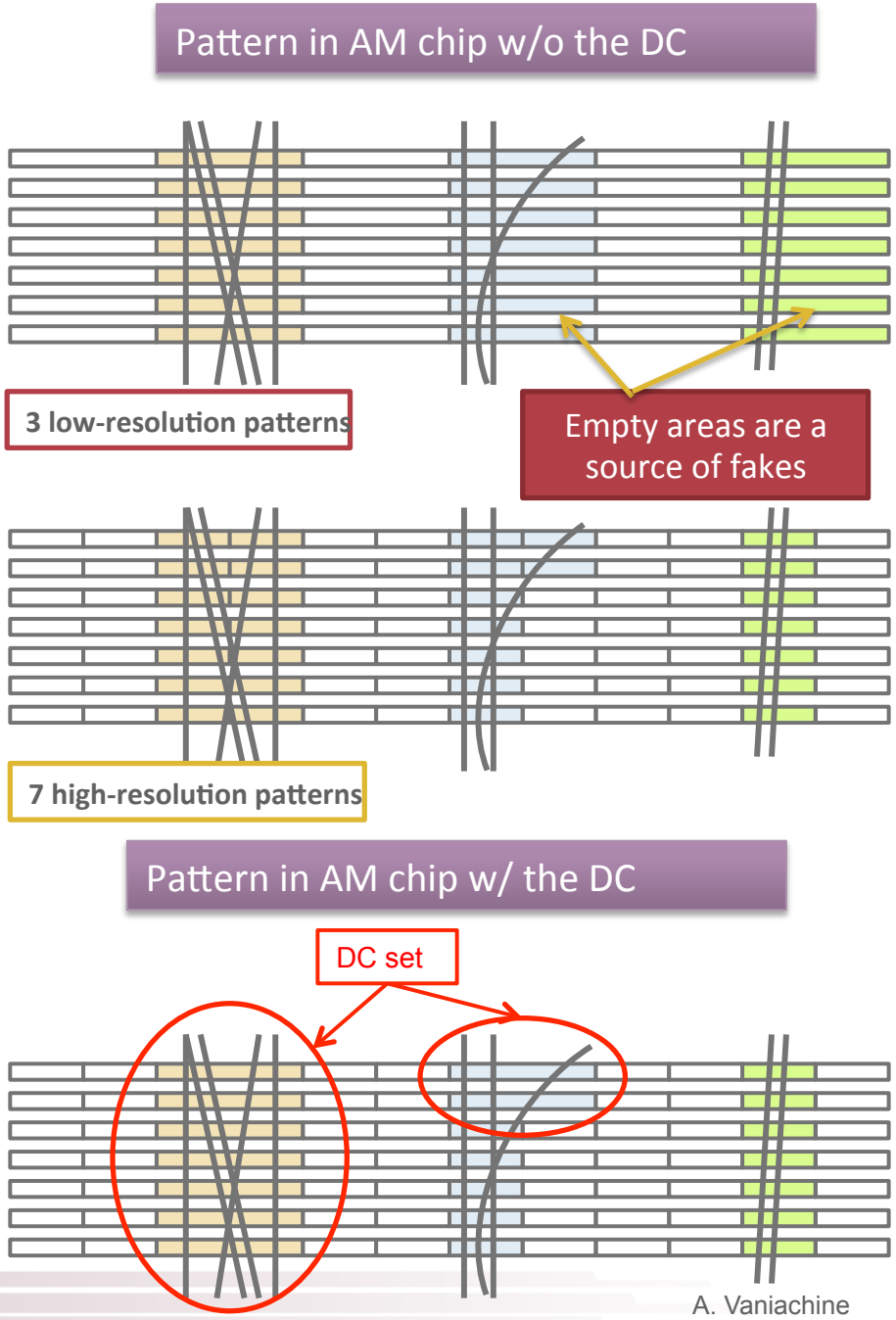
AM Simulation Implementation

- The AM emulation reproduces the matching criteria of the real chip
- Workflow differences improve the algorithm performance and allow tests
  - Pattern bank scan uses a large indexing structure optimizing the access time
  - The Don't Care bit feature is implemented as filter for the roads
- The pattern bank indexing structure has a large memory footprint
  - Indexing structure is large, almost as the pattern bank
  - Gain w.r.t. the full linear scan makes it unavoidable



Don't Care Bit Features

- The LSB bit in AM match lines can use up to 6 ternary bits
  - Each bit allow 0, 1 and X (don't care)
    - *K. Pagiamtzis and A. Sheikholeslami, Solid-State Circuits, IEEE Journal of, vol. 41, no. 3, 2006*
  - The DC bits allow to reduce the match precision where required
- The use of DC solves the problem in balancing the match precision
  - Low resolution patterns allow smaller pattern bank size (less chips, less cost), but the probability of random coincidences grows
  - High resolution fakes increase the filtering power at the price of larger banks
  - DC allows to merge similar pattern in favored configurations (less patterns) maintaining high-resolution and rejection power where convenient



FTK Simulation Details

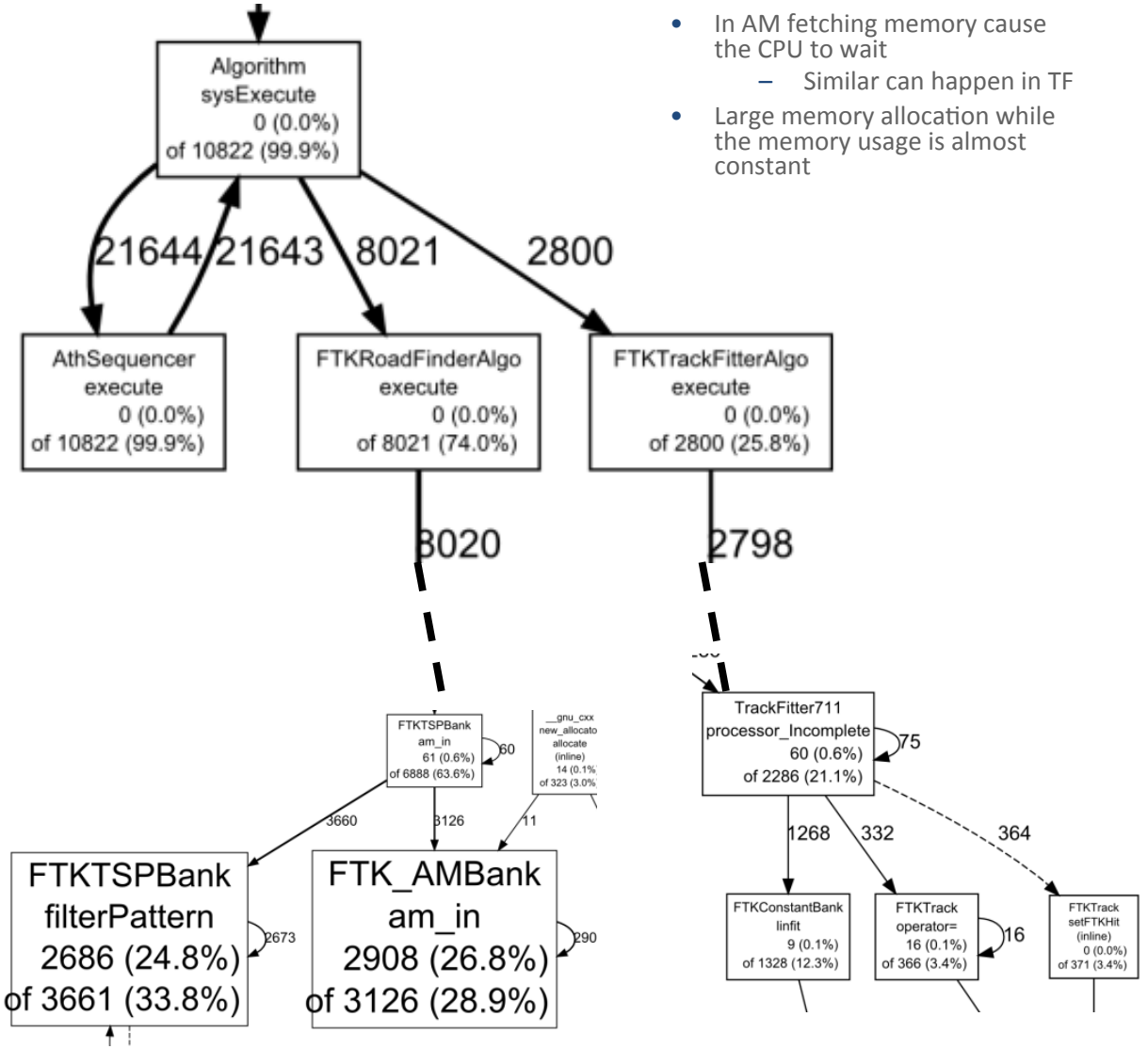
- FTK emulation is controlled by three algorithms
  - FTKRoadFinderAlgo: pattern matching emulation, reading the huge pattern bank and finding coincidences with the hits in an event
  - FTKTrackFitterAlgo: track fitter emulation, reads found patterns and search for good tracks among the sequence of hits selected by the previous stage
  - FTKMergerAlgo: tracks are found within FTK towers (now running in different jobs), they are collected in a single stream and duplicates are rejected
- Large use of resource in memory and disk to load the values that will be loaded in the AM and the FPGAs used for the tracks fitting
  - 10^9 patterns, 9 integers: 36 byte in memory, ~35 GB in memory
  - 10^6 1st stage fit constants, 11x12 floats, 500 MB in memory
  - 1.6×10^6 2nd stage fit constants, 16x17 floats, 1.7 GB in memory
  - Other auxiliary configuration data are within 100 MB
- Disk usage is about 50% less, due to compression
  - Datasets distribution on the Grid is currently used
    - Exploring a long term solution to store references in the ATLAS Conditions DB

Ongoing Code Optimization

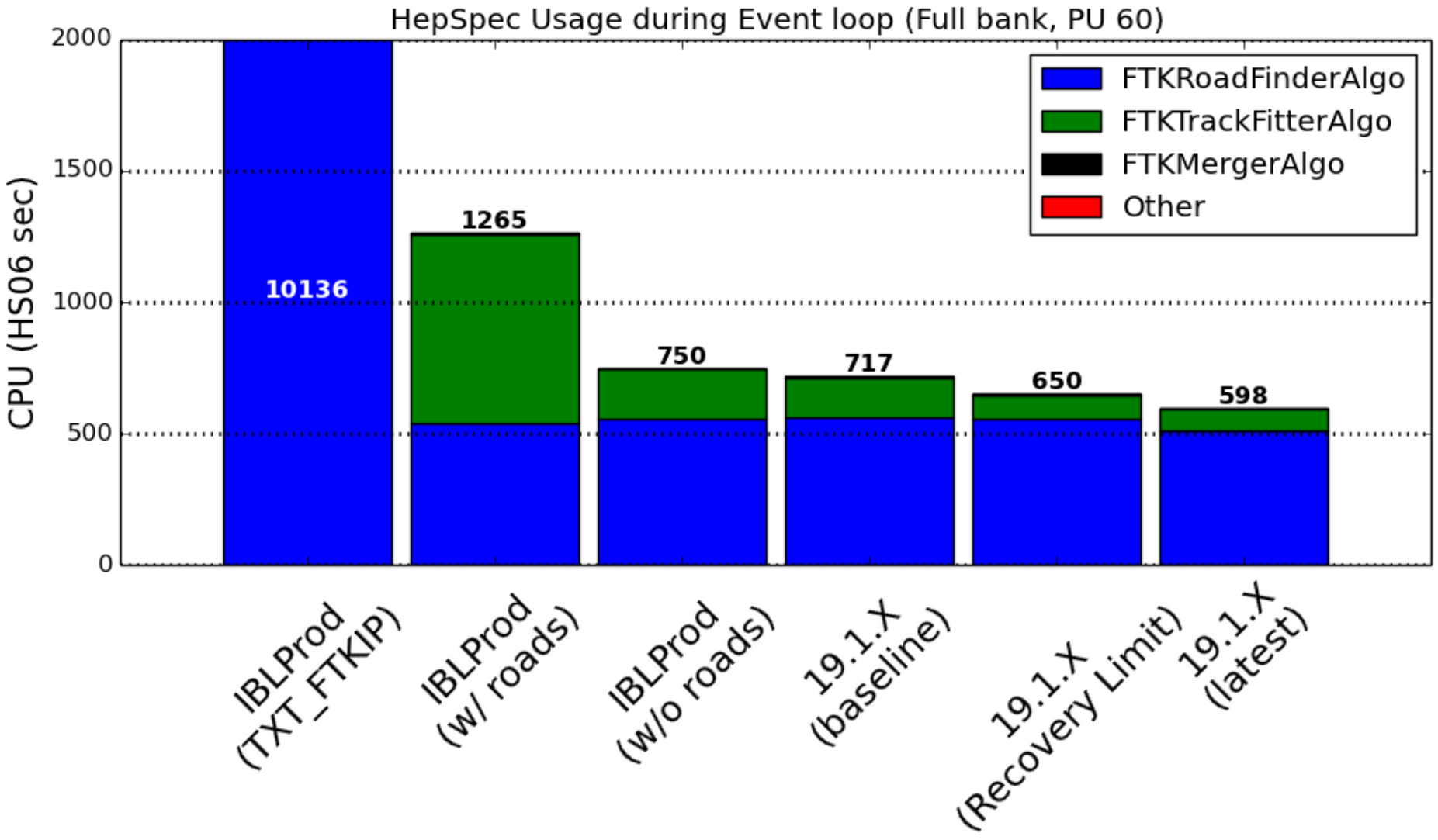
- Full simulation optimization continue
- Memory access is the main bottleneck
  - In AM fetching memory cause the CPU to wait
    - Similar can happen in TF
  - Large memory allocation while the memory usage is almost constant

- FTK emulation has clear hot spots in the algorithms workflow
  - Increasing the similarities with the hardware is expected to improve the CPU performance
    - Further hardware compatibility improvements require additional steps

- Profiling of the single sub-region, with ttbar events having pile-up of 80 and full pattern bank



Measured Performance Speed Up



FTK Fast Emulation

- Improvements in the FTK full emulation implementation has allowed to produce several million of events but alternatives paths are required
- To reduce the CPU consumption further and allow a better integration in ATLAS Integrated Simulation Framework the FTK fast emulation is under development
  - Parameterized response, not simulate the FTK response but use HLT objects' properties to evaluate the efficiency and fake rate of FTK based selections
  - Truth-seeded fast emulation, similar with truth-seeded offline emulation proposal
  - Offline-seeded FTK tracking, similar to previous with the possibility to reproduce fakes if they can be correlated to low quality offline tracks
  - Hybrid approach, combining truth seeding and regional full FTK reconstruction based on the RoI

What does the FTK add to the trigger toolbox?

- Without FTK, ATLAS TDAQ in Run 1 relied on identifying regions of interest (ROI)
  - These ROIs were then reconstructed at Level 2
    - Tracking in an ROI is fast
    - But limited to a narrow scope(s) of the detector, each ROI is about DR~0.2
- The FTK is designed to provide the full track list @100 kHz
  - For all tracks with $p_T > 1.0$ GeV, $|\eta| < 2.5$
  - At phase 1 luminosity
- Every event taken at Level 1 will have a full track list at beginning of Level 2
  - Supply all 5 track helix parameters, χ^2, and hits on detector
- Full list of tracks may be used for:
  - Primary vertex identification
  - Jet vertex fraction, Track-Based e/μ Isolation, b-tagging jets, improved τ selection