Real-time reconstruction of tracks with Retina algorithm at LHCb

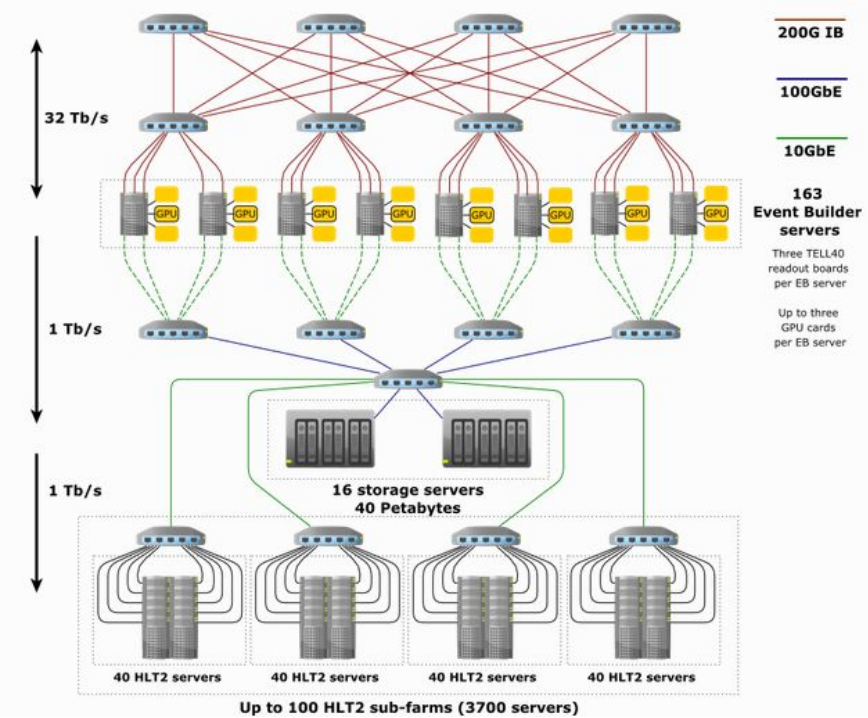
Fabio Novissimo

LHCb experiment

- LHCb performs high precision measurements in the field of flavour physics.
- Large samples are required.
- Measurements are still limited by statistic ⇒ increase luminosity to collect more data.
- Process of selecting interesting events becomes harder.

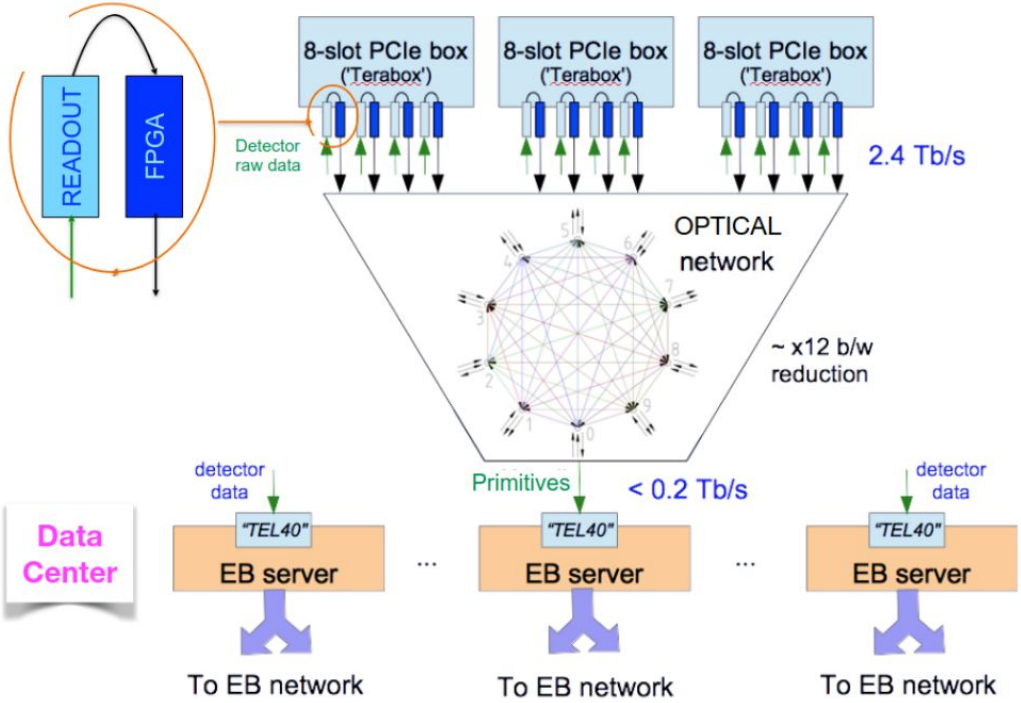
Introduction to the problem: the DAQ

- Data coming from the detector are sent to the Event Builder (EB) which assembles data coming from the same events: event building process.
- An array of GPU performs the first reconstruction and selection of events.
- This DAQ architecture does not scale well with the increase in luminosity.



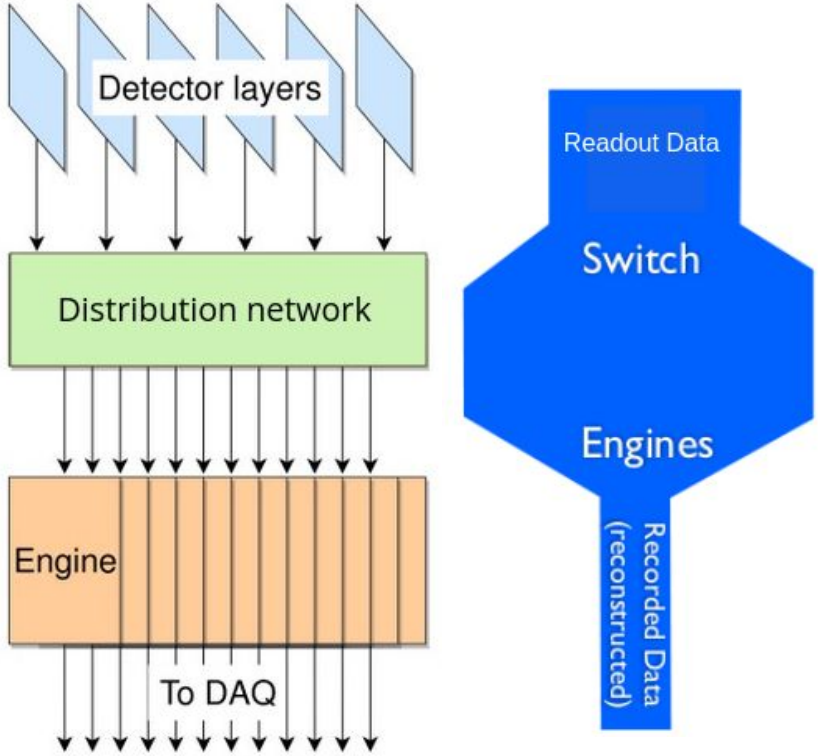
Solution: 2-stage reconstruction scheme

- Pre-evaluate primitives (tracks, calo-clusters) for input to the EB.
- Drop all raw-data not associated to the primitives ⇒ reduction of data flow.
- Reconstruction scheme based on FPGA connected through an optical network.



The “Artificial Retina”

- Architecture for fast reconstruction imitating some features of the biological vision.
- Main features:
  - Low latency
  - High degree of parallelization
  - Low power consumption
- First realistic application on the experiment: the VELO demonstrator.



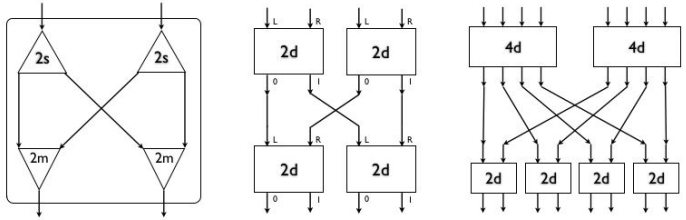
Goal of my thesis

Design and optimization of the communication network that connects all the FPGAs of the system.

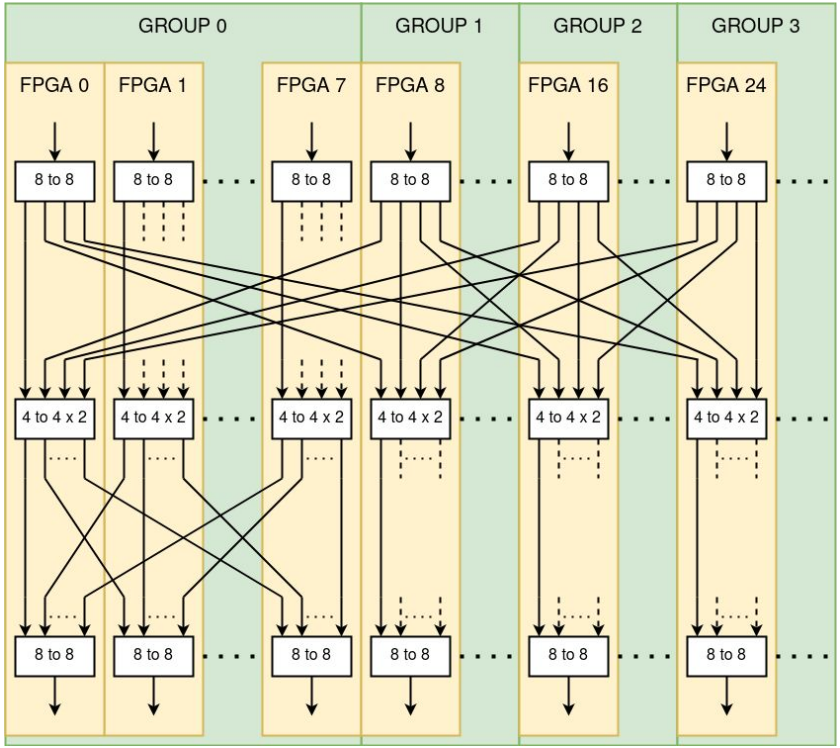
Main components of the communication network:

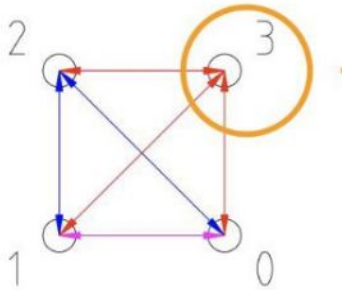
- Dispatcher: 2 inputs and 2 outputs
- Track Processing Unit (TPU): minimal functional unit to perform track reconstruction

Goal:
Choose the optimal way to connect the TPUs to the Dispatchers (nodes) of the network.

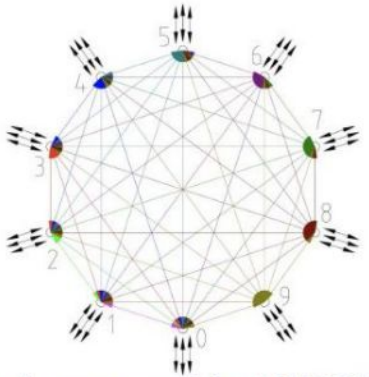


Network topology





4 interconnected networks



each composed by 10 FPGA

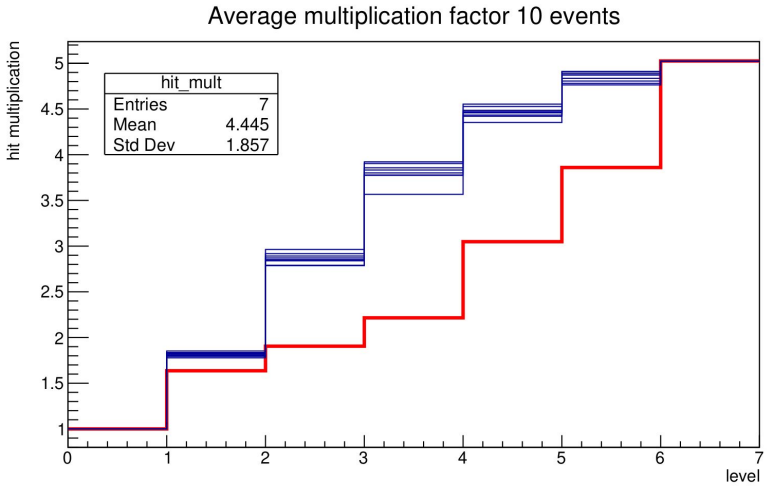
Heuristic optimization algorithm

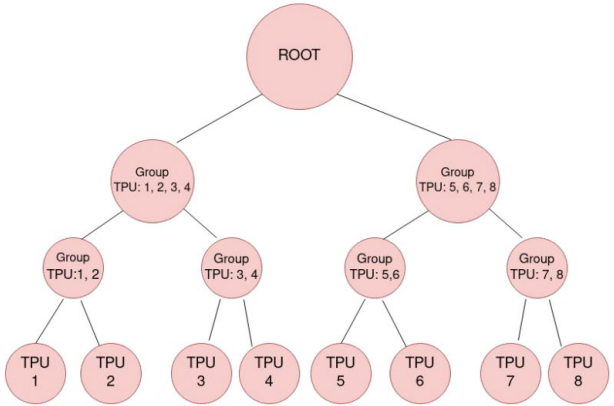
- Main idea: group the TPUs in the nodes in a way such as to delay the bandwidth increase to the last stage of the Switch.
- Group the TPUs having more hits in common.

Below the gain in bandwidth in the case where we consider the switch as a binary tree.

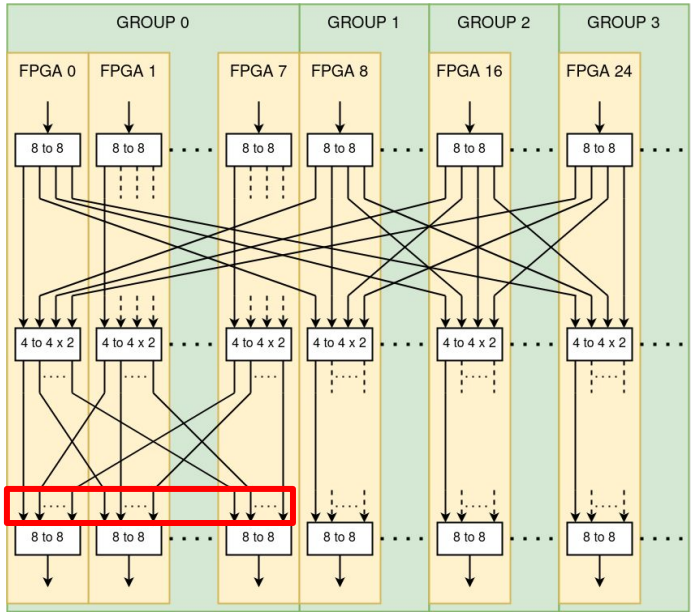
Red: optimized Switch (binary tree)

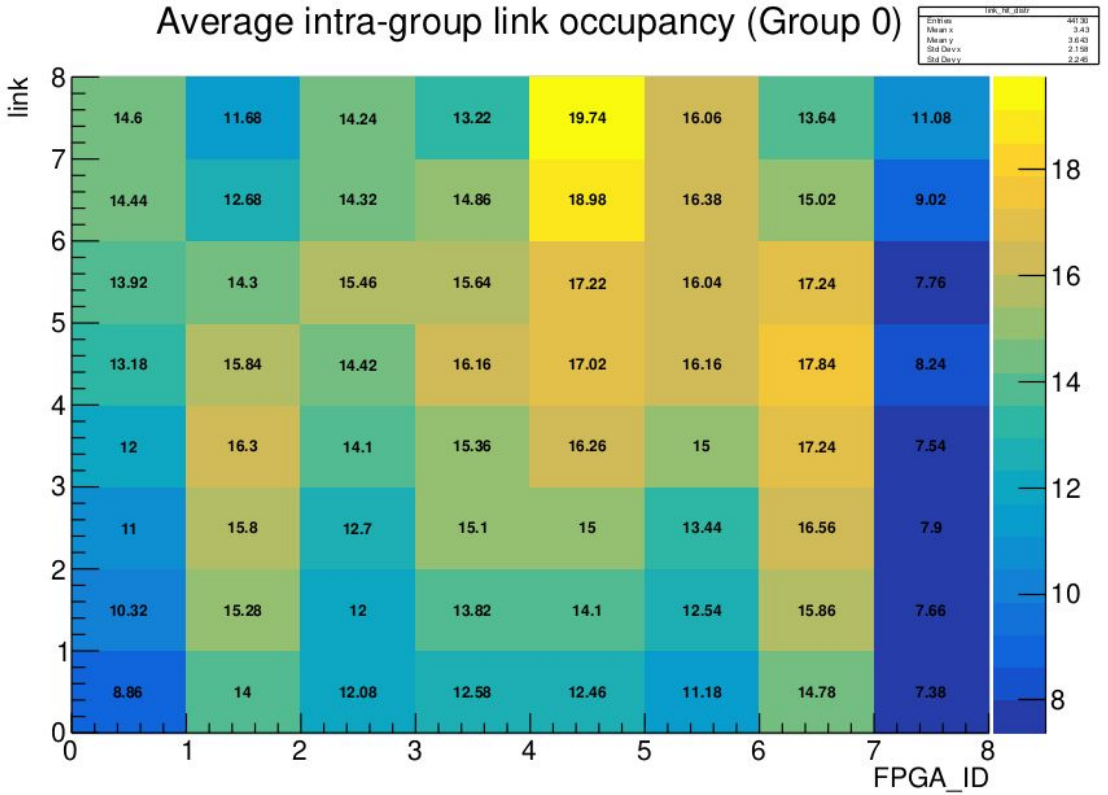
Blue: random Switch (binary tree)





Optical link occupancy





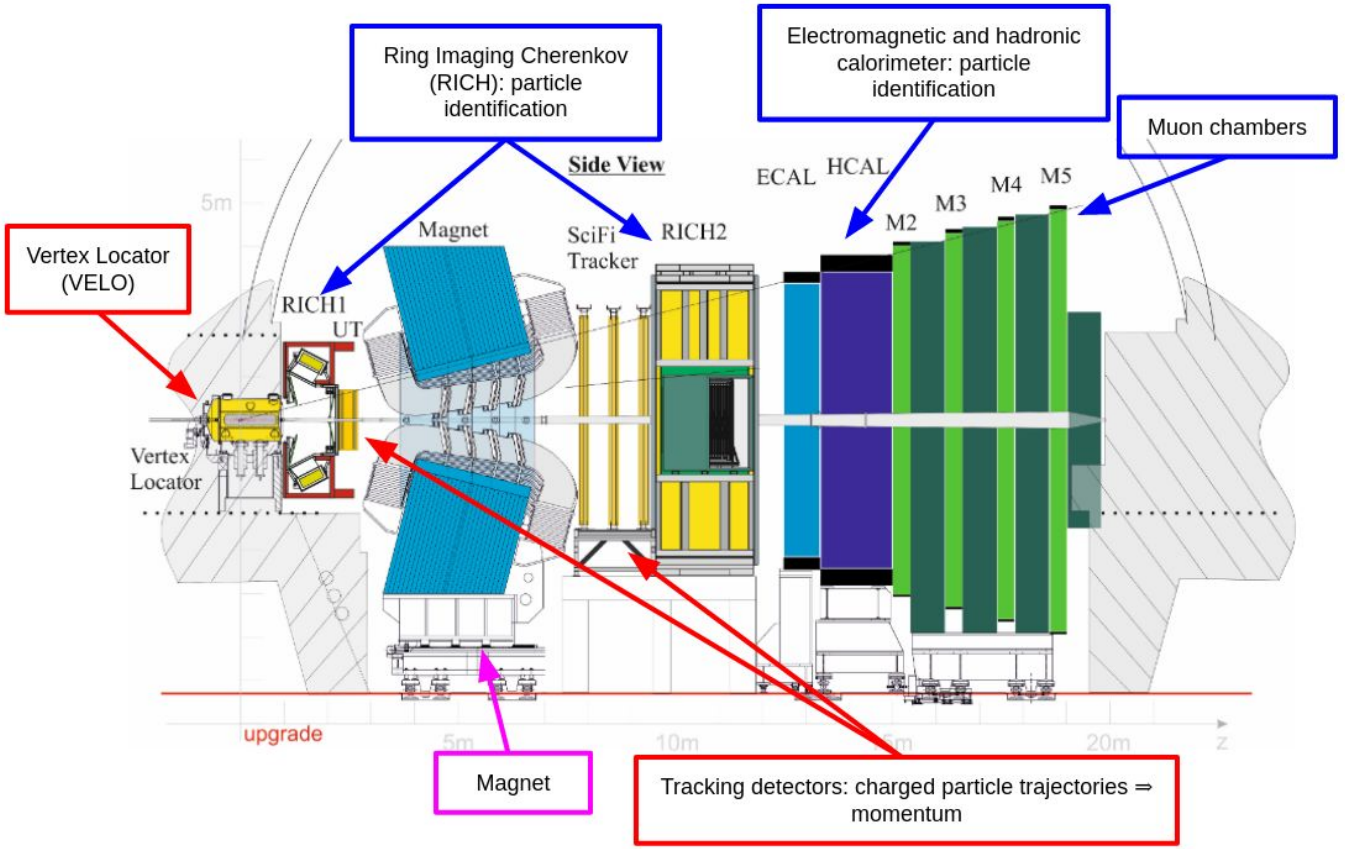
Conclusions

- The LHCb experiment is developing a new Data Acquisition system, to further increase the size of its data samples and the statistical precision of its measurements.
- The project is based on fast FPGA cards, connected by a high-speed optical network.
- I designed, simulated and optimized the connections in this network for maximum capacity.
- My results show that this new device is feasible with the existing optical link technology.
- My optimization tool is now being used for the existing hardware prototype, and will be extended to design the actual system that will be installed in LHCb in the next run.

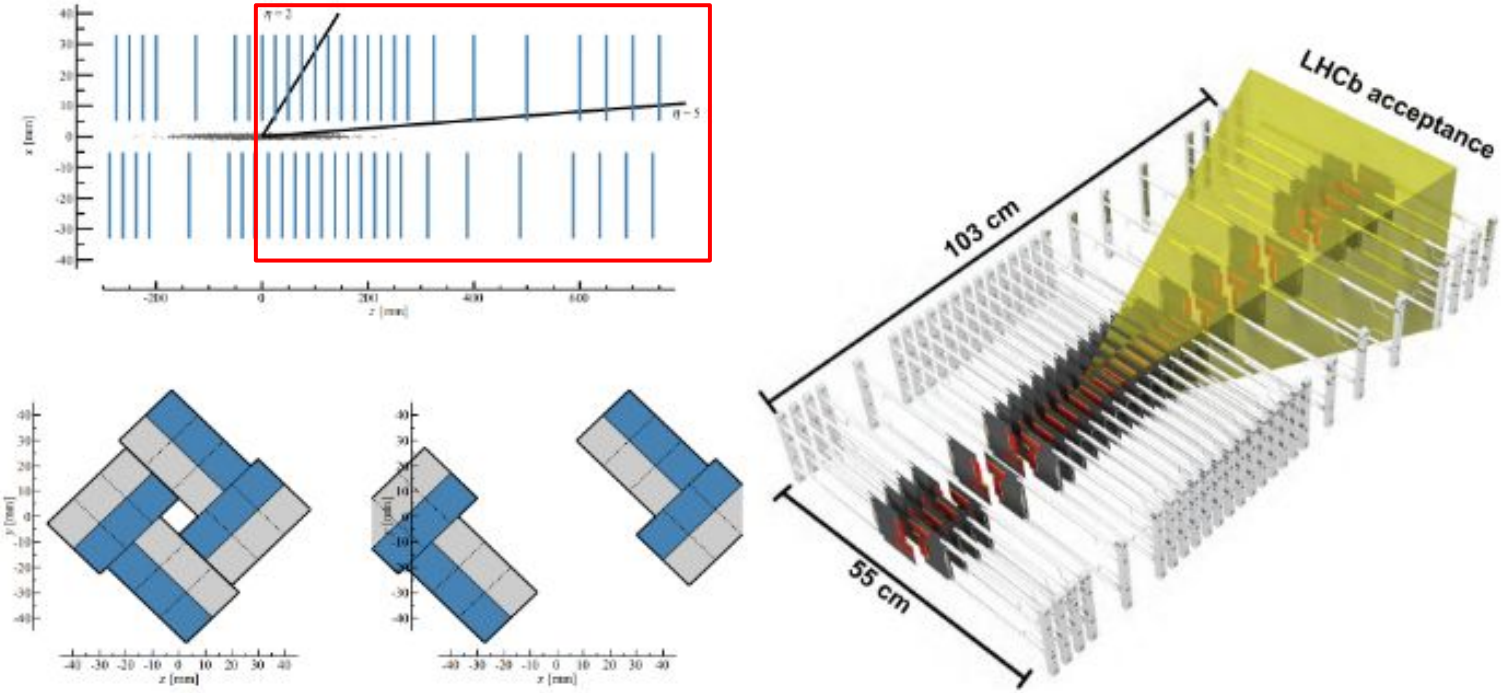
Thanks for your attention

Backup slides

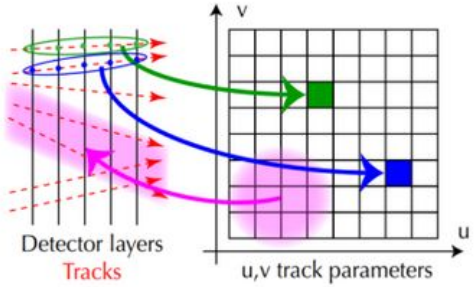
LHCb detector



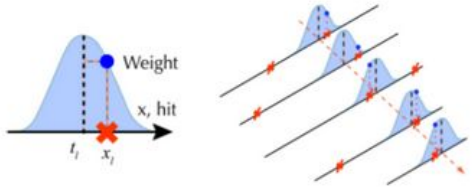
VErtex LOcator (VELO)



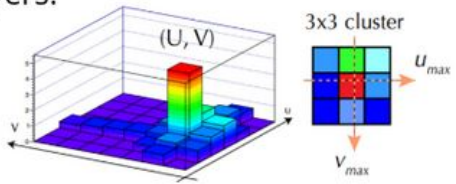
Artificial Retina algorithm



Parameter space divided into cells.
Each cells correspond to a pattern.
Receptors: interception of pattern tracks with detector layers.



Distance hit- receptor is calculated.
A weight is assigned to each distance and for each cell all the weights are summed (excitation level)



Search of local maxima over a threshold → reconstructed tracks

Everything is executed in parallel!

Heuristic algorithm: binary tree

- For every pair of TPUs (i, j) compute $|H_i \cap H_j|$ where H_i = {hits in TPU i}.
- Build a matrix M where:
  - $M_{ij} = |H_i \cap H_j|$
  - $M_{ij} = |H_i \cap H_j| / |H_i \cup H_j|$
  - $M_{ij} = |H_i \cap H_j| / (|H_i \cup H_j| + |H_i \cap H_j|)$
- Find the max of M and its position recursively.
- Pair the corresponding TPUs.
- Move to the next level and repeat the previous steps with the newly created nodes instead of the TPUs.

Heuristic algorithm: real Switch topology

Different FPGAs each taking hits from a specific VELO module. The algorithm generalizes as follows:

- Make a copy of the newly created nodes.
- For each possible way to divide the modules between the two group of nodes, choose the combination for which the difference of the number of hits going into the two groups is minimal.
- Repeat the steps of the simplified algorithm, creating the frequency matrix and pairing the nodes together.
- Iterate until there are no more modules to divide.