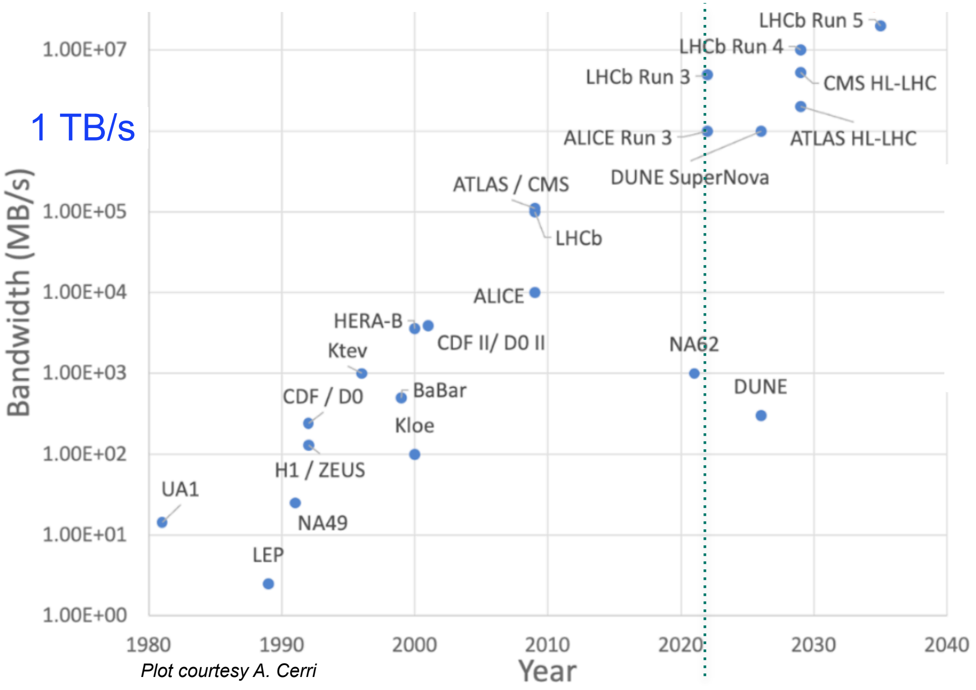
# FPGA-Based Real-Time Reconstruction for HEP Experiments

Giovanni Punzi - Università di Pisa & INFN

CEPC International Workshop - Nov 6-10, 2025 - Guangzhou

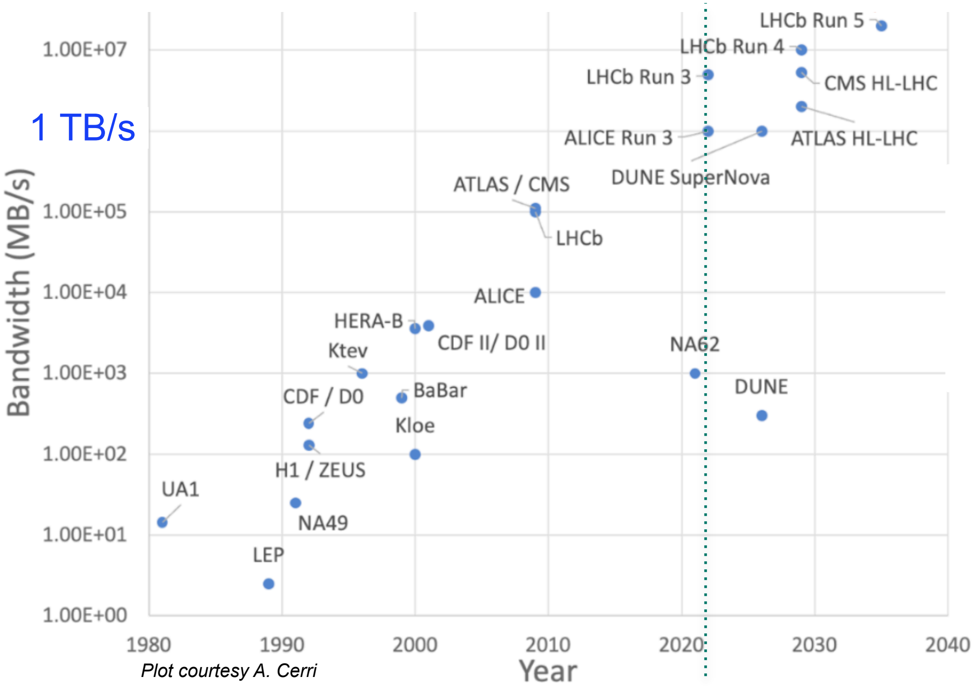
Istituto Nazionale di Fisica Nucleare

# HEP data processing rate vs time




*Plot courtesy A. Cerri*

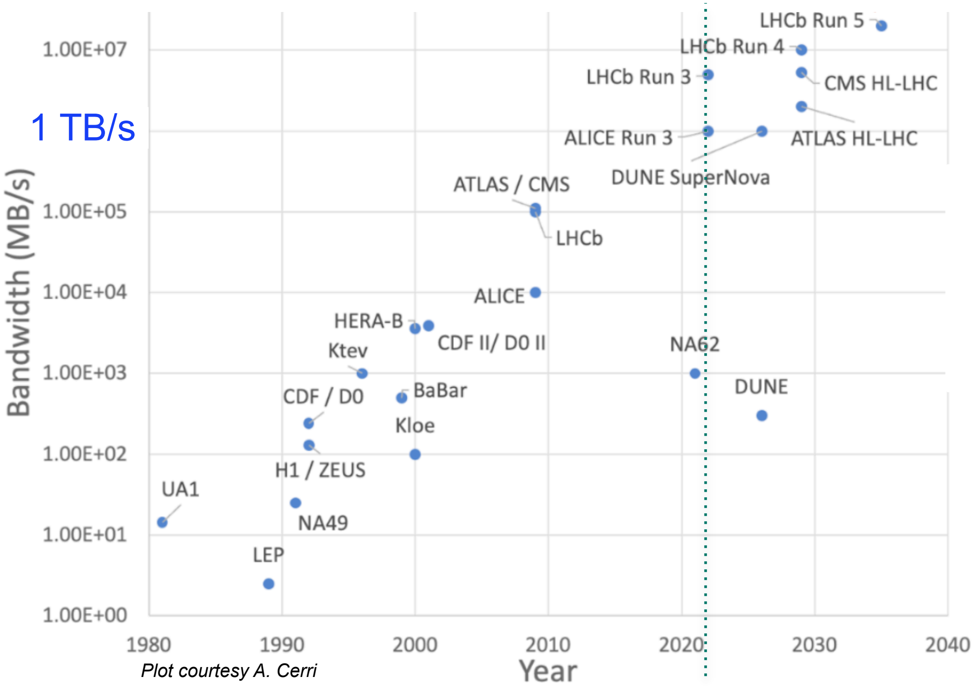
# HEP data processing rate vs time




*Plot courtesy A. Cerri*

- Progress of experiments goes together with increasing data processing rate.

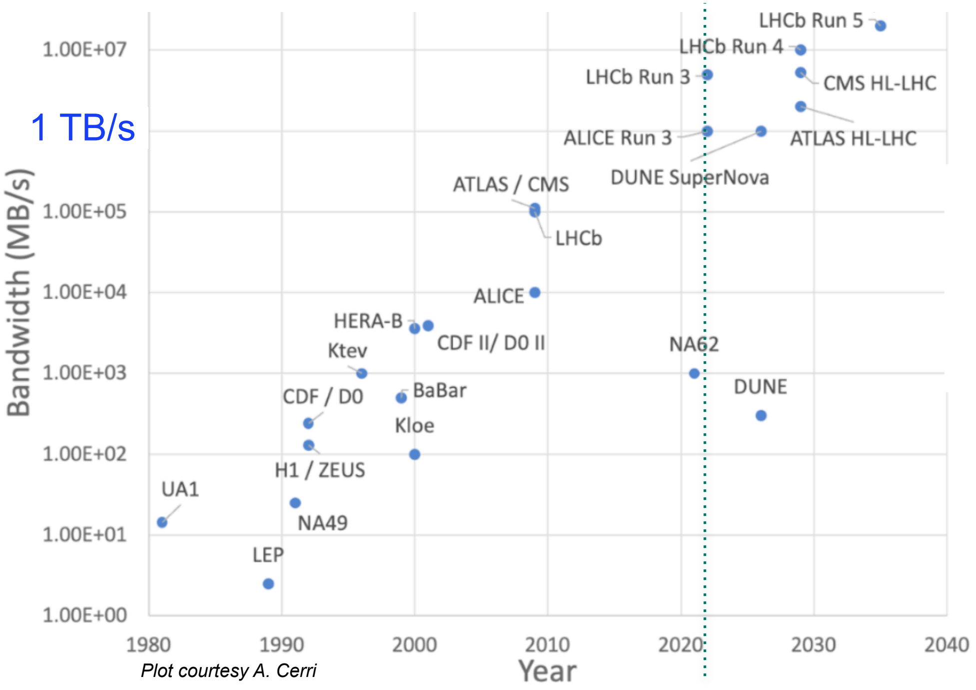
# HEP data processing rate vs time




*Plot courtesy A. Cerri*

- Progress of experiments goes together with increasing data processing rate.
- **I describe an approach to accelerate data reconstruction, using FPGAs closely integrated with detector readout ("RETINA").**

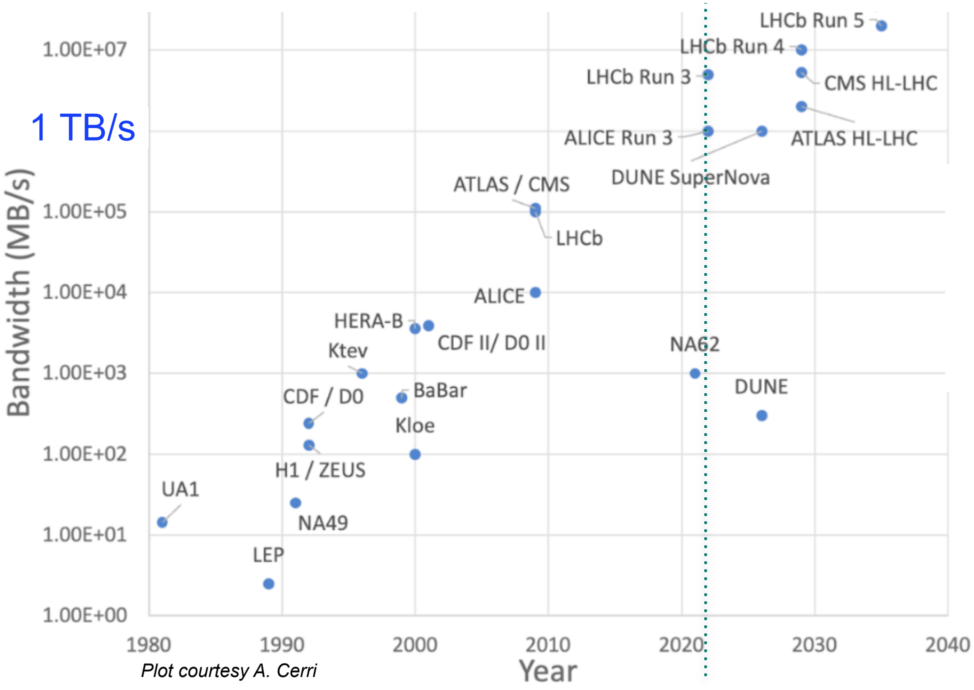
# HEP data processing rate vs time



*Plot courtesy A. Cerri*

- Progress of experiments goes together with increasing data processing rate.
- **I describe an approach to accelerate data reconstruction, using FPGAs closely integrated with detector readout ("RETINA").**
- Low-pt Flavor physics the most demanding: => LHCb a prime target of this R&D when it started in 2014@INFN [idea dating back to 2000]

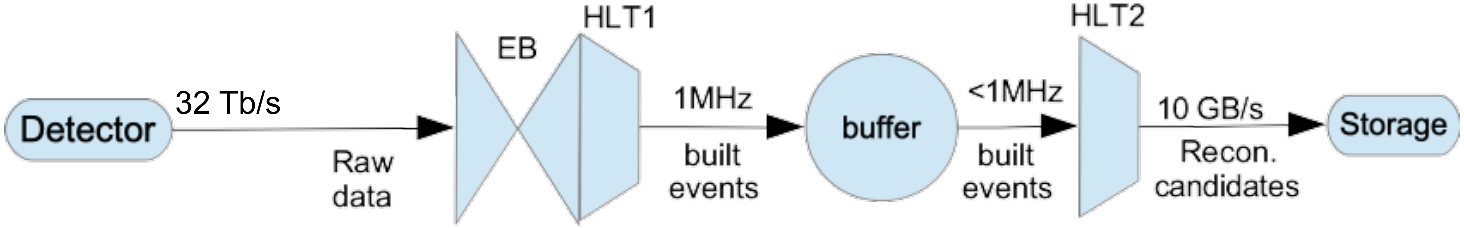
# HEP data processing rate vs time




*Plot courtesy A. Cerri*

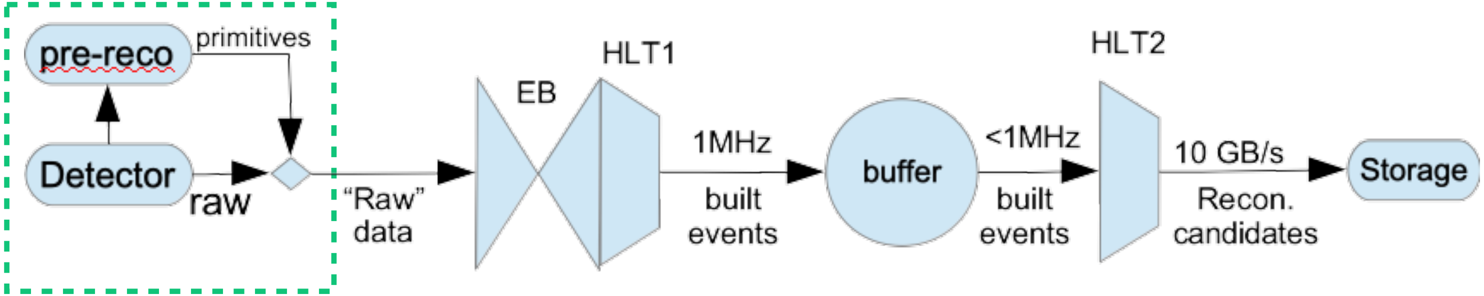
- Progress of experiments goes together with increasing data processing rate.
- **I describe an approach to accelerate data reconstruction, using FPGAs closely integrated with detector readout ("RETINA").**
- Low-pt Flavor physics the most demanding: => LHCb a prime target of this R&D when it started in 2014@INFN [idea dating back to 2000]
- But the approach is generally applicable and its benefits go beyond high data rates

# Data Processing model @LHCb

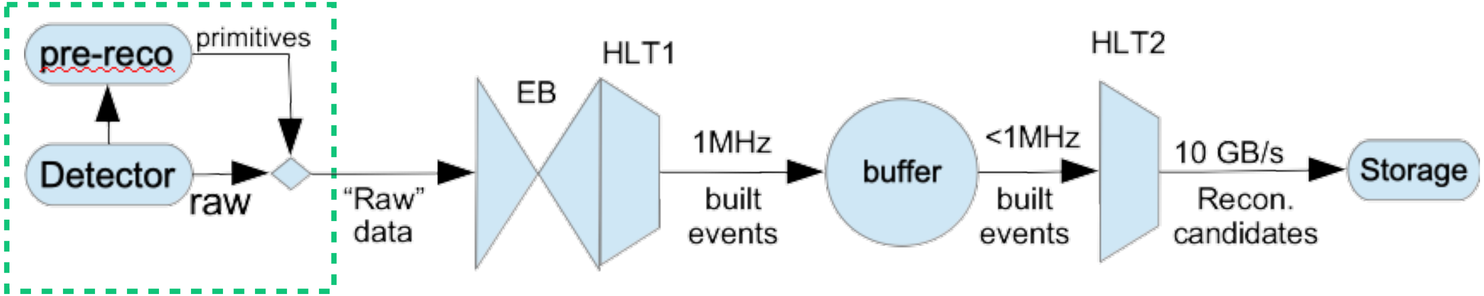



- Triggerless readout of whole detector + ~full event reconstruction before first trigger decision is made
  (often referred to as 'trigger less', or 'full-software trigger')
- Two-level DAQ: HLT1 (extensive reco for trigger purpose), HLT2 (physics reconstruction + final selection).
  - Alignments between HLT1 and HLT2, to make sure HLT2 reco is final.
    (Large disk buffer in the middle)
  - In Run3, HLT1 moved physically inside the Event Builder to save on data transport, and turned to GPUs for better efficiency, cost.
- What could still be improved ?

# Adding primitive-based reconstruction

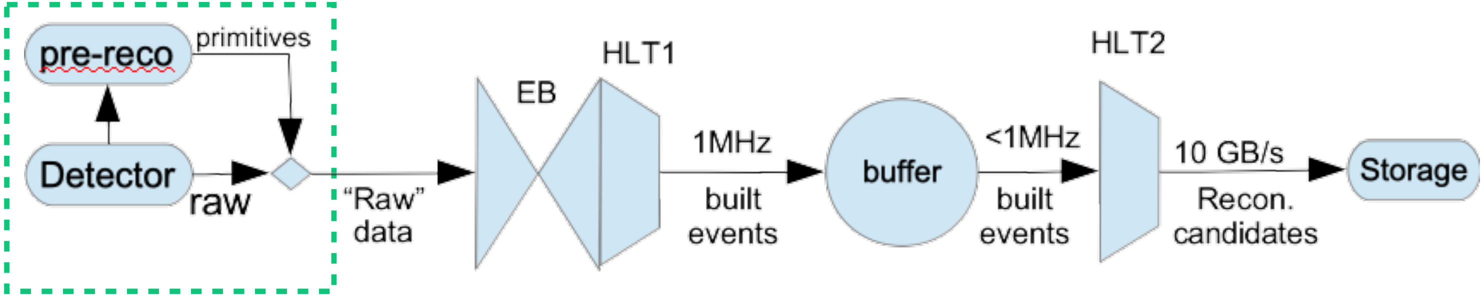


# Adding primitive-based reconstruction



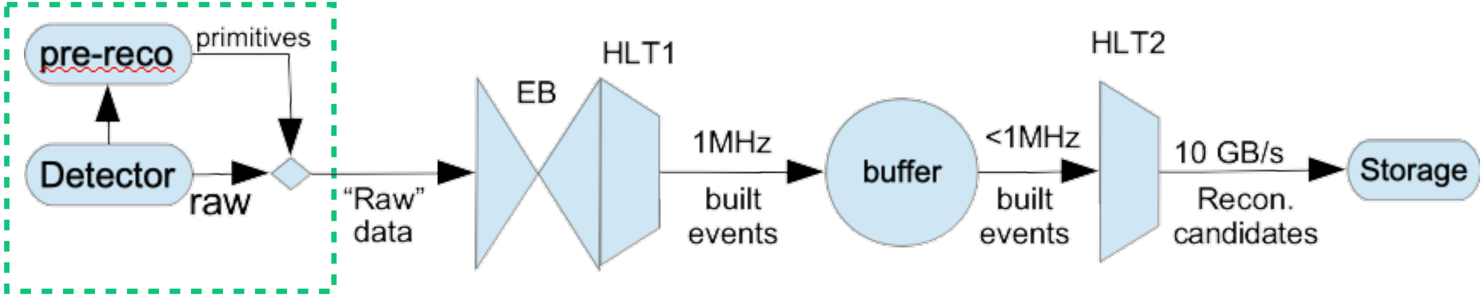

- Push processing *before* EB: reconstruct intermediate data structures ("primitives") using ~local info.
  - Ex. Track segments, muon stubs...
  - <u>Logically embed in the detector block:</u> make primitives look like "Raw Data" to the DAQ.

# Adding primitive-based reconstruction



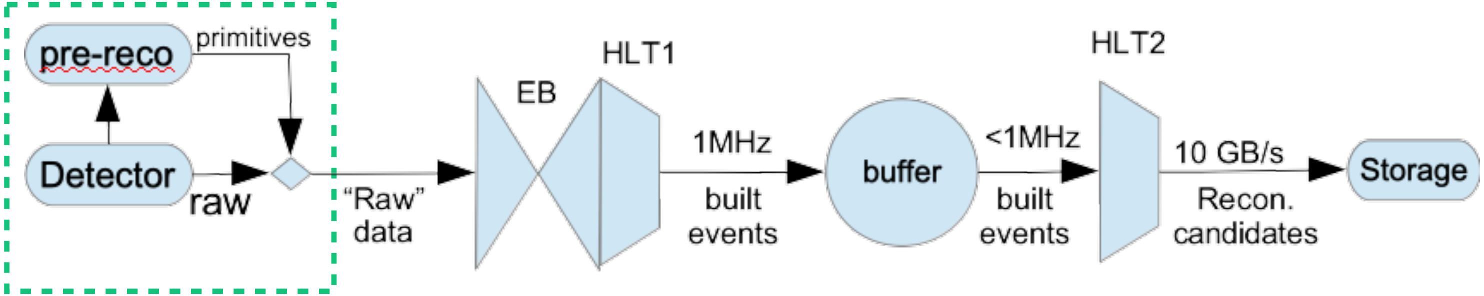

- Push processing *before* EB: reconstruct intermediate data structures ("primitives") using ~local info.
  - Ex. Track segments, muon stubs...
  - <u>Logically embed in the detector block:</u> make primitives look like "Raw Data" to the DAQ.
- Advantages:
  - Accelerate HLT reconstruction. This helps whether the readout is triggerless or not
  - Reduce data flow at the source (e.g. drop hits not on a track)

# Adding primitive-based reconstruction



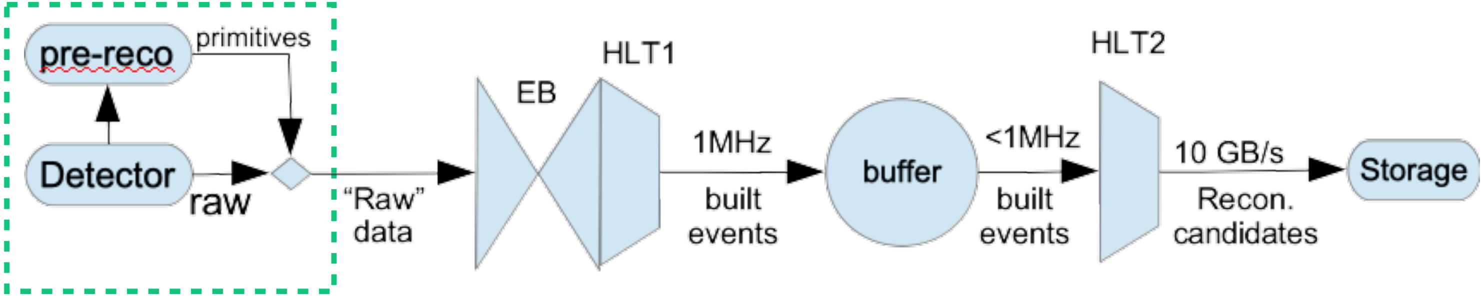

- Push processing *before* EB: reconstruct intermediate data structures ("primitives") using ~local info.
  - Ex. Track segments, muon stubs...
  - <u>Logically embed in the detector block:</u> make primitives look like "Raw Data" to the DAQ.
- Advantages:
  - Accelerate HLT reconstruction. This helps whether the readout is triggerless or not
  - Reduce data flow at the source (e.g. drop hits not on a track)
- Drawback: it is hard !
  - Can't use time-multiplexing 'a la GPU': (split rate over many processors). Need to *actually* process a new event every 25ns.
  - Large b/w, little buffering, constrained latency.

# Adding primitive-based reconstruction




- Push processing *before* EB: reconstruct intermediate data structures ("primitives") using ~local info.
  - Ex. Track segments, muon stubs...
  - Logically embed in the detector block: make primitives look like "Raw Data" to the DAQ.
- Advantages:
  - Accelerate HLT reconstruction. This helps whether the readout is triggerless or not
  - Reduce data flow at the source (e.g. drop hits not on a track)
- Drawback: it is hard !
  - Can't use time-multiplexing 'a la GPU': (split rate over many processors). Need to *actually* process a new event every 25ns.
  - Large b/w, little buffering, constrained latency.
- Example at LHC Phase-2: CMS' "hit doublets" simplify tracking. Use on-detector ASICs (2S modules).

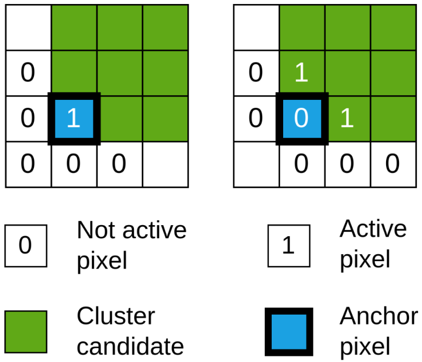
# Adding primitive-based reconstruction




- Push processing *before* EB: reconstruct intermediate data structures ("primitives") using ~local info.
  - Ex. Track segments, muon stubs...
  - <u>Logically embed in the detector block:</u> make primitives look like "Raw Data" to the DAQ.
- Advantages:
  - Accelerate HLT reconstruction. This helps whether the readout is triggerless or not
  - Reduce data flow at the source (e.g. drop hits not on a track)
- Drawback: it is hard !
  - Can't use time-multiplexing 'a la GPU': (split rate over many processors). Need to *actually* process a new event every 25ns.
  - Large b/w, little buffering, constrained latency.
- Example at LHC Phase-2:  CMS' "hit doublets" simplify tracking. Use on-detector ASICs (2S modules).
- For more complex primitives, can use off-detector FPGAs (in the back-end)

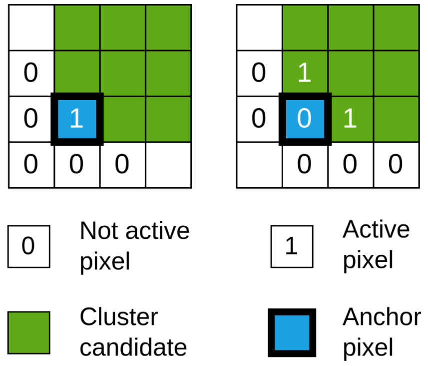
# Example in operation: hit finding in a pixel detector

- Hits in the LHCb's pixel vertex detector (VELO) appear as clusters of pixels
- Firmware deployed in Run3 in FPGA readout boards to find clusters on-the-fly (Arria 10)
- Pixels read out as 2*4 arrays (SuperPixels, SP). Clusters found by unpacking them into active matrices, where each pixel actively checks if it belongs to a pattern. Centroids evaluated by LUT. Fast solution in principle, but cannot handle large-area detectors.

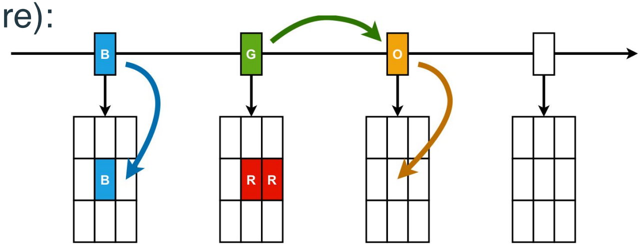


# Example in operation: hit finding in a pixel detector

- Hits in the LHCb's pixel vertex detector (VELO) appear as clusters of pixels
- Firmware deployed in Run3 in FPGA readout boards to find clusters on-the-fly (Arria 10)
- Pixels read out as 2*4 arrays (SuperPixels, SP). Clusters found by unpacking them into active matrices, where each pixel actively checks if it belongs to a pattern. Centroids evaluated by LUT. Fast solution in principle, but cannot handle large-area detectors.




- Solution: dynamically allocate matrices where active pixels are found.
- Input data travel along a chain of empty matrices (dataflow architecture):
  - When a SP hits an empty matrix, it allocates it to its position
  - It a SP hits a matrix it belongs to, it fills the matrix
  - Cluster finding happens in parallel in all matrices (no buffering)

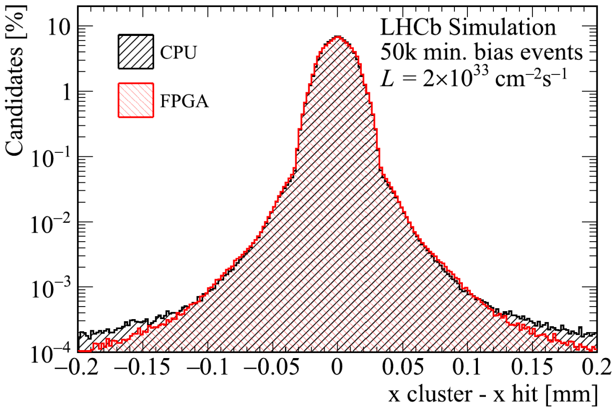



-> works at **~30 MHz@Run3 Lumi ($2\times10^{33}$ cm$^{-2}$s$^{-1}$)** [IEEE TNS 70, 6 (2023)]

# Benefits of embedded Cluster finding

- Quality of real-time cluster reconstruction as good as CPU algorithm
- Cluster data is 15% more compact
- FPGA implementation saved 12% of initial HLT1 computing power, using 1/50th of the electrical power [IEEE TNS 70, 6 (2023)]

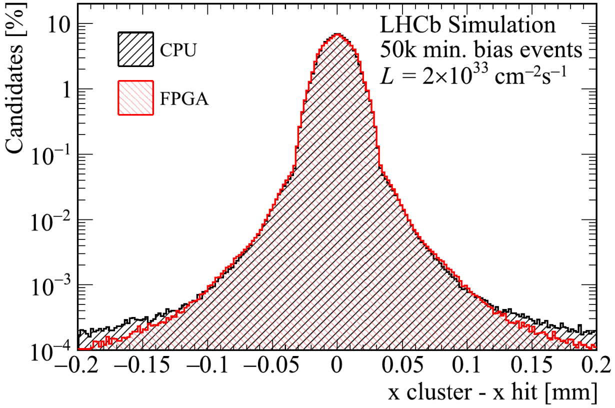
-> Now running in LHCb by default: **hit positions replace pixels as raw data**



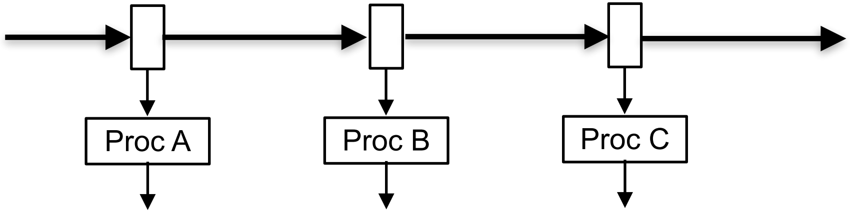
# Benefits of embedded Cluster finding

- Quality of real-time cluster reconstruction as good as CPU algorithm
- Cluster data is 15% more compact
- FPGA implementation saved 12% of initial HLT1 computing power, using 1/50th of the electrical power [IEEE TNS 70, 6 (2023)]

-> Now running in LHCb by default: **hit positions replace pixels as raw data**



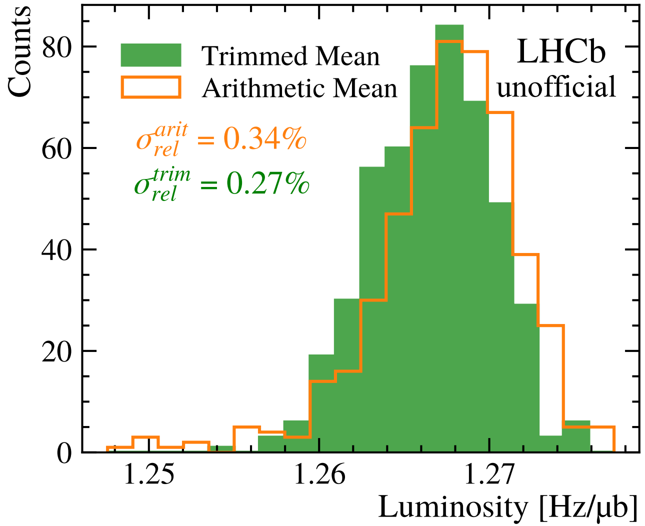

- But real-time availability of $10^{11}$ hits/s **in accessible way** enables further applications
- Data in a FPGA is not buried in some buffer, but swims through a "sea of gates"
- You can attach further processing modules to the chain without stopping the data flow



# Applications of real-time hit finding

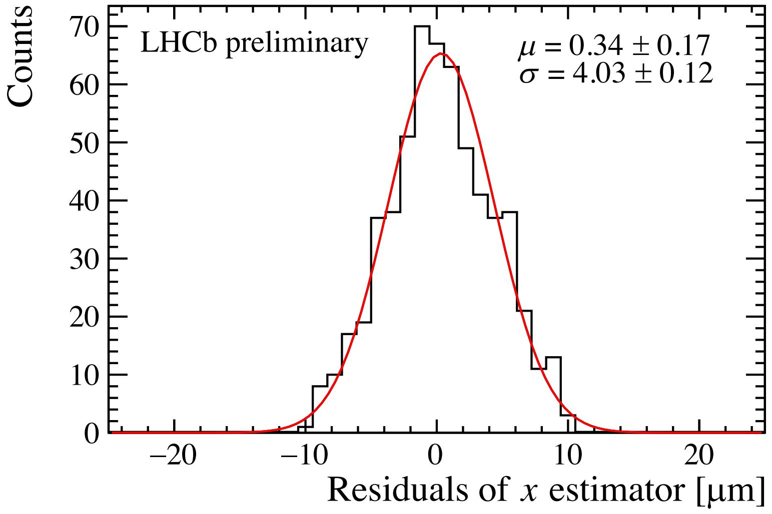
- Current application: **fast monitoring of LHC beam vs time**
- High-rate accelerator luminosity monitor (paper in writing)
  - Simple hit counting yields sensitive measurements fast
    - Hits are cleaner than raw pixel data
  - Can count many different regions, correct for defects...
- New applications in development:
  - Fast beam position monitor
  - Detector alignment monitor and real-time correction
  - Automated detection+suppression of defective channels

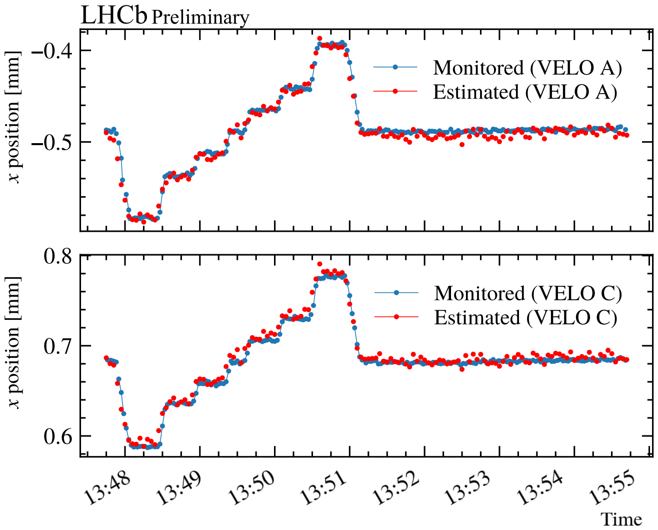
**-> cleaner data -> lower systematics -> higher-precision**




LHCb-DP-2025-006, in preparation

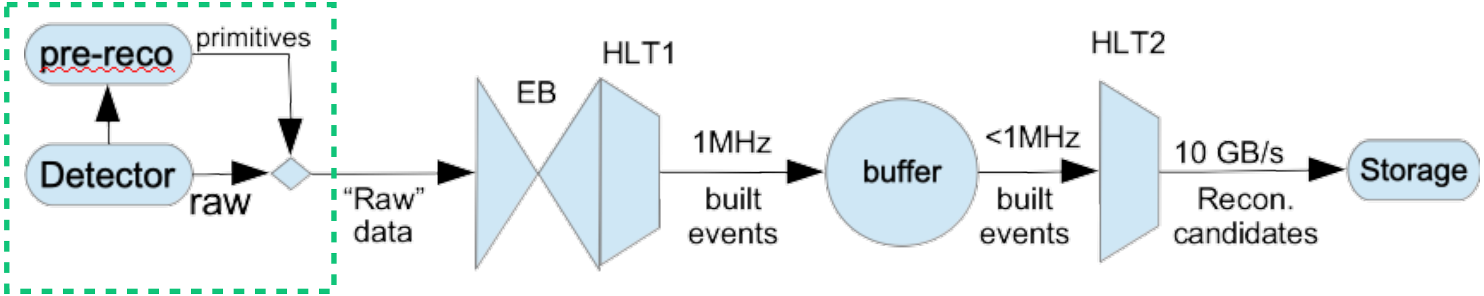
# Real-time beam/detector position measurements




- Large hit rate require **no track reconstruction** ('trackless') to find beam position
  - simply exploits cylindrical symmetry of hit distribution ($\Longrightarrow$**No track alignment needed)**
- **O(μm)** precision, continuous monitoring. Can measure different parts of detector independently

# Not-so-local reco: from clusters to tracks




- Moving track reconstruction to the primitive-reco level gets more difficult
- More complex that clustering, because tracking is not a 'local' computation. Need to converge data.
- At this level, cannot afford much buffering (low latency) neither multiplexing to 500 parallel streams
  - Looking at event tracking in ~25 ns ...
  - Is this even possible ?

# Past experience in ultra-fast tracking

| Name | Tech. | Exp. | Year | Event rate | clock | cycles/event | latency |
|---|---|---|---|---|---|---|---|
| SVT | AM | CDF-L2 | 2000 | 0.03 MHz | 40 MHz | ~1600 | <20μs |
| FTK | AM | ATLAS-L2 | 2014 | 0.1 MHz | ~200 MHz | ~2000 | O(10μs) |
| *?* | *?* | *LHC-L0* | *~2030* | *40MHz* | *~1GHz* | *~25* | *~μs* |

- Fastest tracking devices required O($10^3$) clock cycles/event
  - All based on pattern matching with stored track patterns
  - Extremely fast - but LHC at Level 0 **needs O(100) better** even with evolution of clock speed
- **Impossible ?**
-

# Past experience in ultra-fast tracking

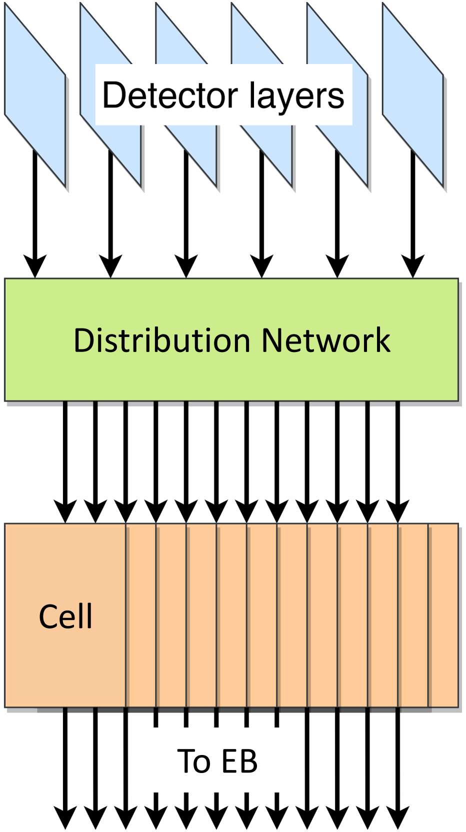
| Name | Tech. | Exp. | Year | Event rate | clock | cycles/event | latency |
|---|---|---|---|---|---|---|---|
| SVT | AM | CDF-L2 | 2000 | 0.03 MHz | 40 MHz | ~1600 | <20μs |
| FTK | AM | ATLAS-L2 | 2014 | 0.1 MHz | ~200 MHz | ~2000 | O(10μs) |
| *?* | *?* | *LHC-L0* | *~2030* | *40MHz* | *~1GHz* | *~25* | *~μs* |
| *Vision* | *(neural)* | *(Brain)* | *old* | *~40 Hz* | *~1kHz* | *~25* | *<100ms* |

- Fastest tracking devices required $O(10^3)$ clock cycles/event
  - All based on pattern matching with stored track patterns
  - Extremely fast - but LHC at Level 0 **needs O(100) better** even with evolution of clock speed
- A **bold conceptual analogy**: natural vision in the brain (retina -> area V1) as a tracking system
  - Bit of a stretch, but not as much as it seems. Finds lines and edges in an image.
- Interestingly, the neural system has the required speed, if you replace the slow biological clock with the silicon clock, faster by $10^6$ .
- It's the neural architecture: data-flow, no buffers, no memory, no processors: **compute-while-moving-data**

  => try and implement the same principles in a artificial architecture: FPGA + fast serial links

  => The hit-finding method described before came from this. But tracks are more difficult

# 'Retina' Architecture




Input from detector over multiple lines

Distribution network:

- Custom Switch: routes hits only to appropriate cells.

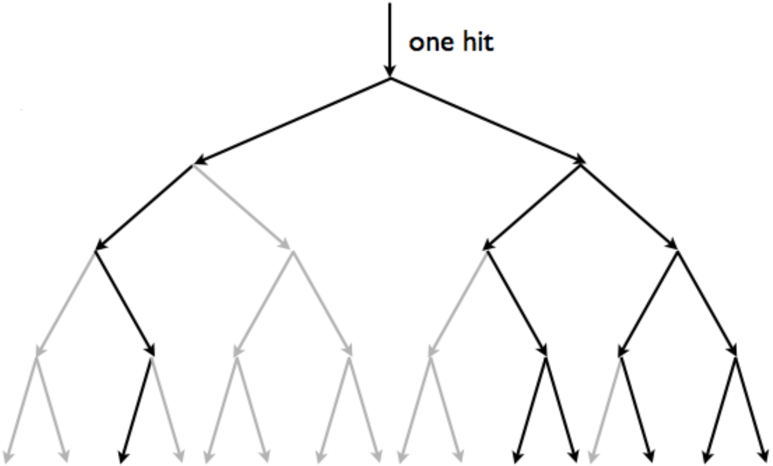
Cell:

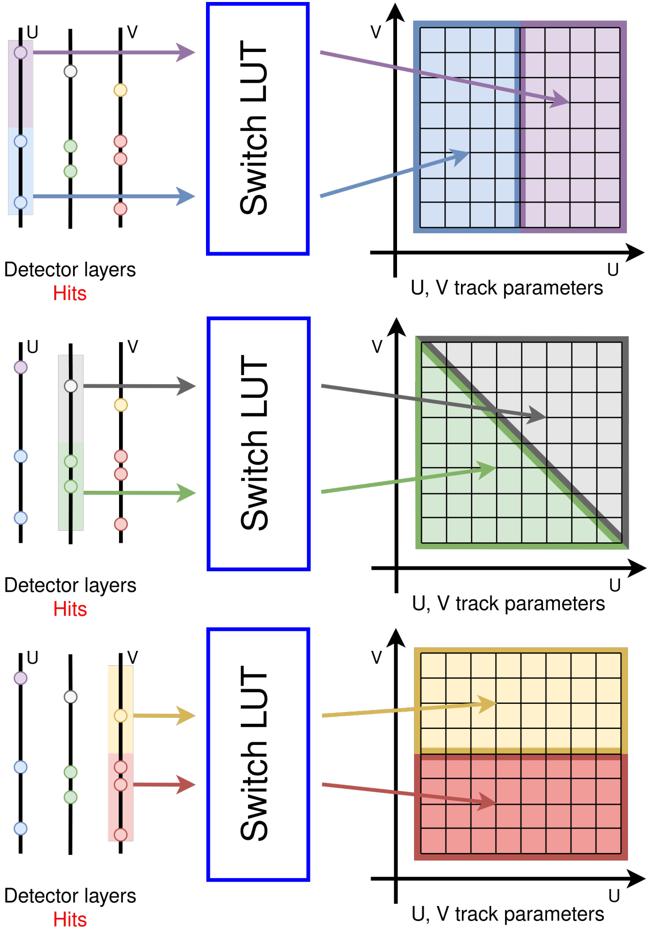
- Evaluates presence of primitive in specific region

Primitives are sent to the Event Builder.

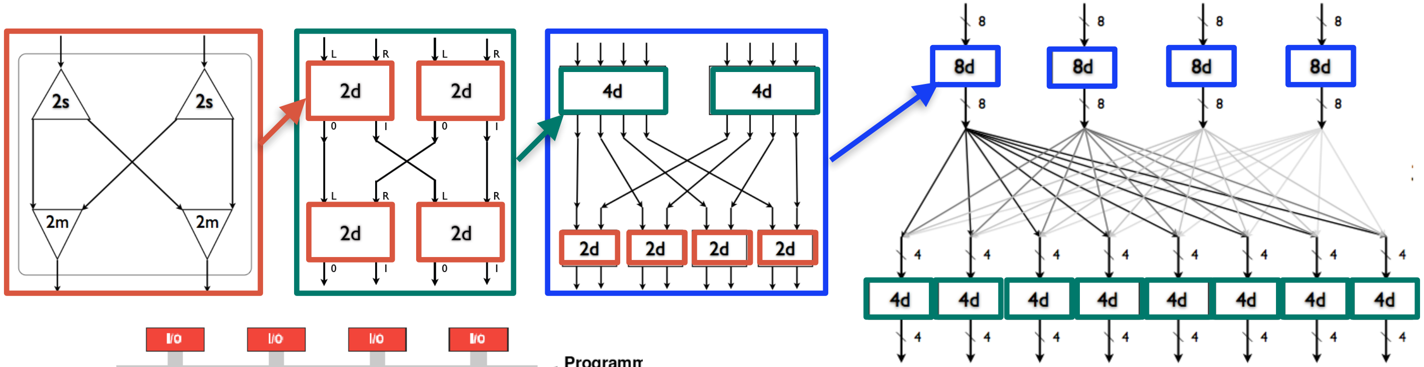
# The Distribution Network

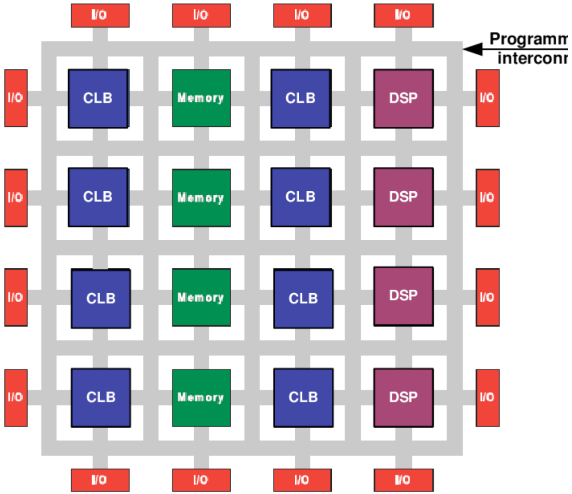
- Hits from different detector layers delivered to different zones of the cell matrix according to layer/coordinates (similar to a “change of reference system”/Hough transform)
- No-stopping process, duplicate hits as necessary



# Building a large custom switching network recursively from uniform elementary blocks implemented in FPGA fabric




- Fast switching network implementation exploits the freedom allowed by the fast programmable FPGA interconnections
- Requires the **large bandwidth** of modern FPGAs
- Cannot do in single chip: breakdown in different board, connected by modern fast optical links

# Industry Analogy: Microsoft's CATAPULT

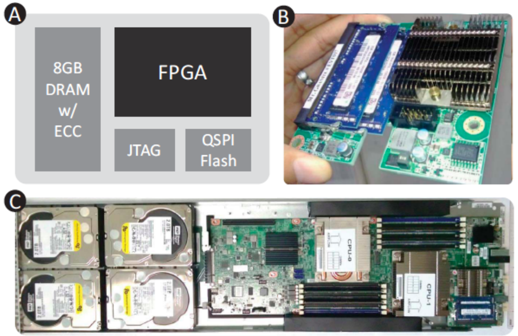
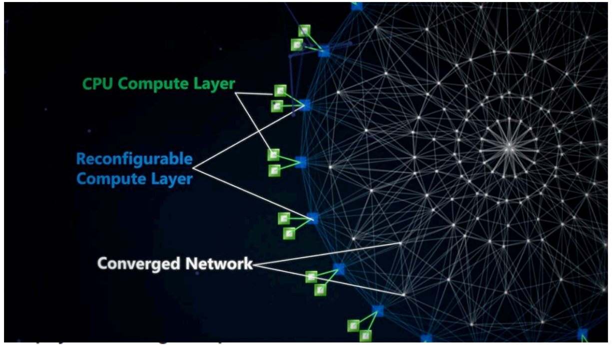



Figure 1: (a) A block diagram of the FPGA board. (b) A picture of the manufactured board. (c) A diagram of the 1 U, half-width server that hosts the FPGA board. The air flows from the left to the right, leaving the FPGA in the exhaust of both CPUs.




Proceeding of the 41st ISCA - ISBN 978-1-4799-4394-4 p. 13-24

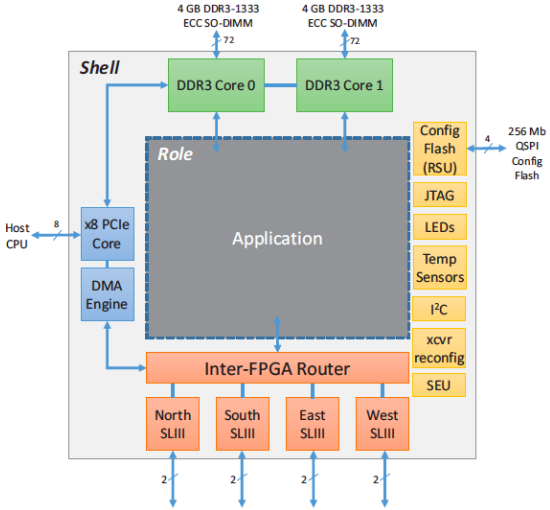
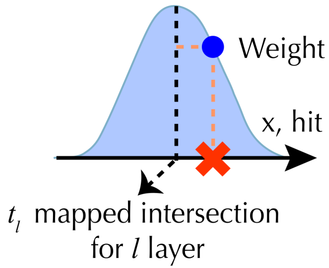
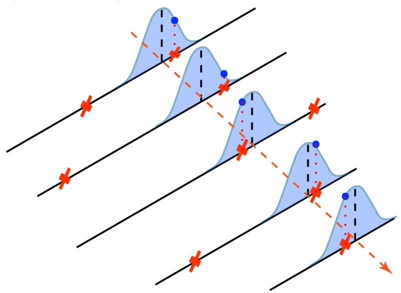



Figure 3: Components of the Shell Architecture.

- No time to describe board connectivity, just mention a structural analog in independently-developed CATAPULT architecture
- Distributed, inter-connected FPGA boards (powering Bing)
- Larger latencies, but similar issues

# Processing done by individual cells



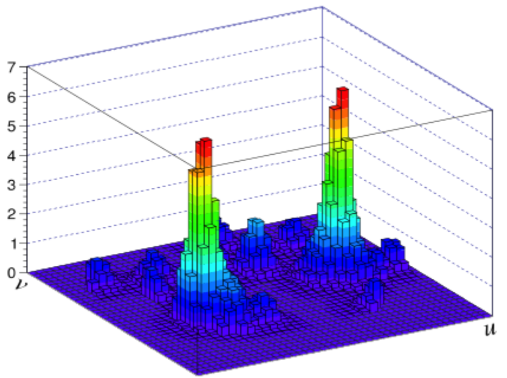



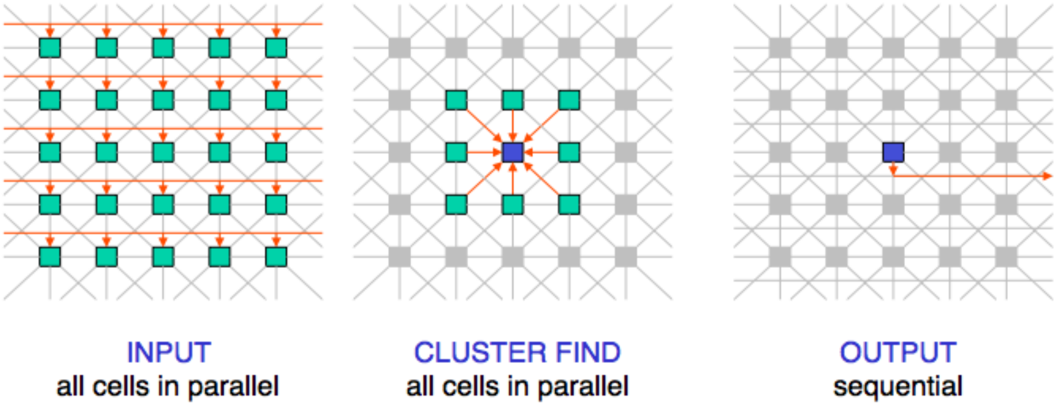
$$R = \sum^{\text{hits}} e^{-\frac{(x_l - t_l)^2}{2\sigma}}$$

R is close to N (# of layers) only if we have a set of hits near the mapped track

- Each cell computes its response (*R*) as the weighted sum of inputs
  - For tracking, hits closer to the reference track get larger weight (Gaussian in the example)
- Digital analogue of "receptive fields" in vision processing in the natural brain
  - Hence the historical name 'retina architecture'
  - More specific than a generic 'neural net'
  - Calculation must still happen "in zero-time" for the system to work

# Processing within the cell matrix



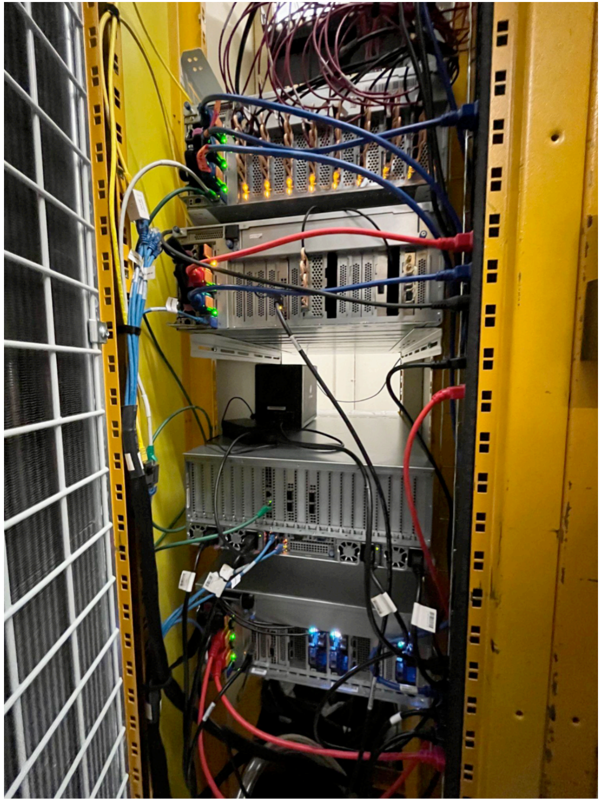

- 3 steps happen simultaneously (pipelined) while input is coming, in order not to stop the flow:
    1. All cells are filled in parallel
    2. <u>Clusters</u> found by local negotiations between neighbors (similar to pixel clustering)
    3. Output of cluster centers are queued to output
- A final pipeline stage may be added to perform application-dependent processing (e.g. track fitting)

# Hardware demonstration

- A complete Retina demonstrator was installed and tested at the LHCb TestBed facility. Culmination of a decade-long effort.

- Reconstruct a VELO quadrant using 8 PCIe-hosted FPGA cards (Stratix-10, 2.8 MLE). (Takes VELO clusters as input)

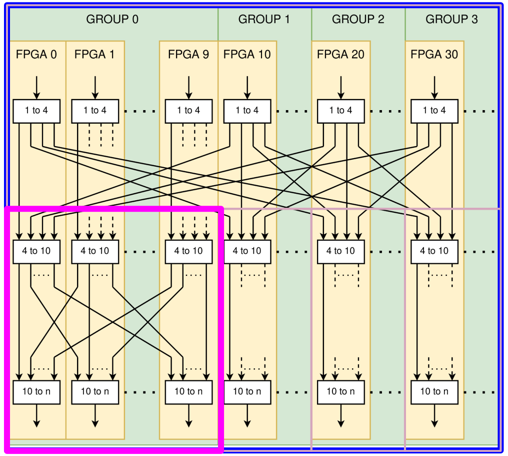
- Test on LHCb MC @Run3 luminosity ($2\times10^{33}$ $cm^{-2}s^{-1}$).
  - Achieved **20 MHz event rate** (LHCb rate ~27 MHz)
    - That is < 20 clock cycles/event
    - **O($\mu$s) latency** (no internal buffering)
  - Low power consumption (~500 W)
- Steady reliable running for weeks

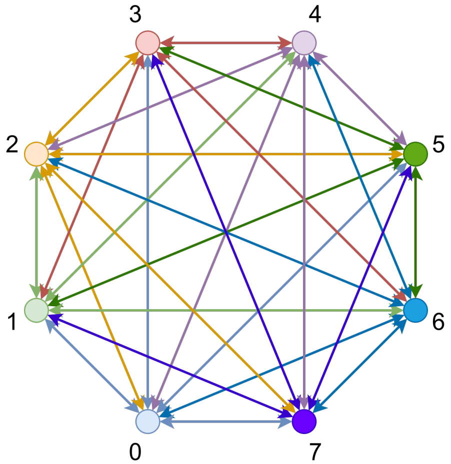


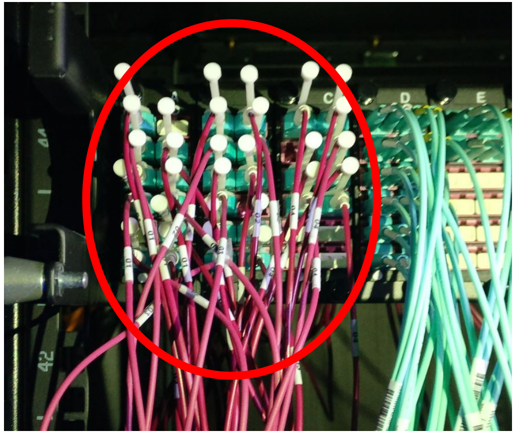
# Detail of the switching network

- Topology: 8-nodes full-mesh network.
- 28 full-duplex optical links at 25.8 Gbps, total bandwidth 1.4 Tb/s.
- Open source protocol Intel SuperLite II v4
- Traffic managed by LUTs - dedicated optimization code for load-balancing

- Implemented via optical patch panel, allows for easy reconfiguration

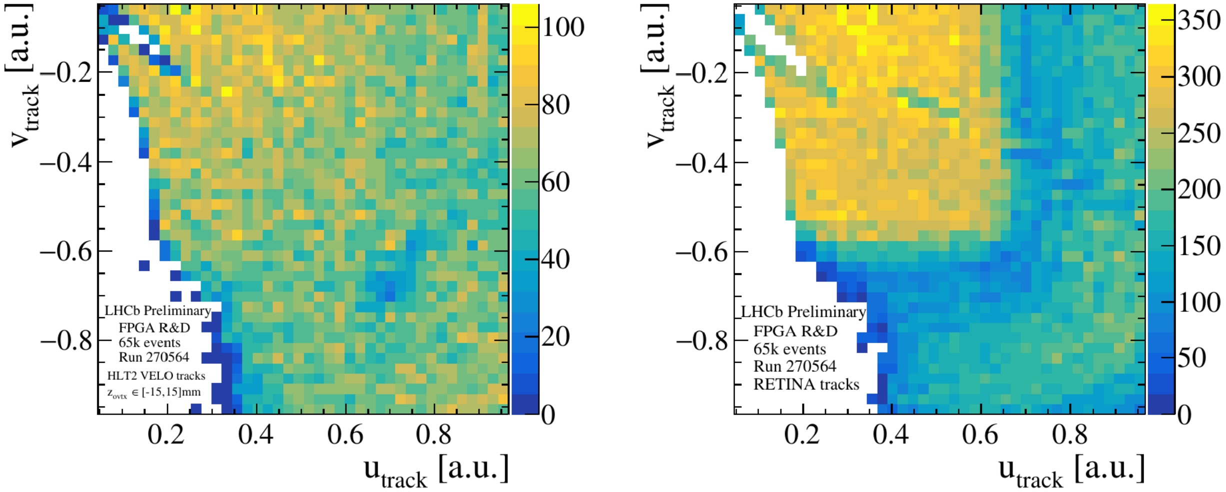


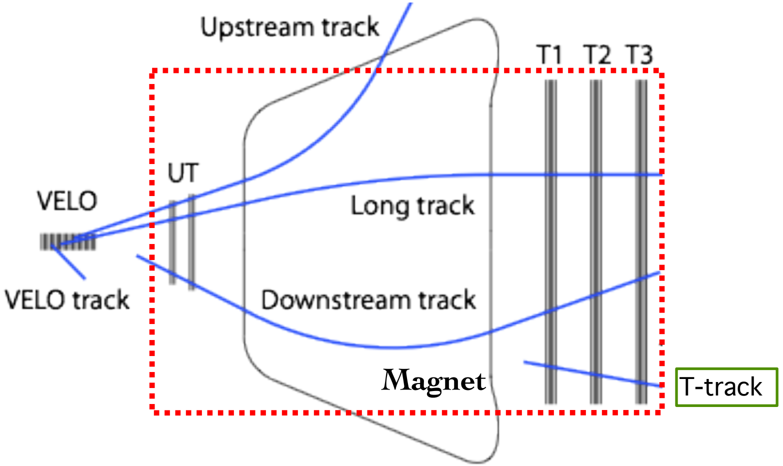



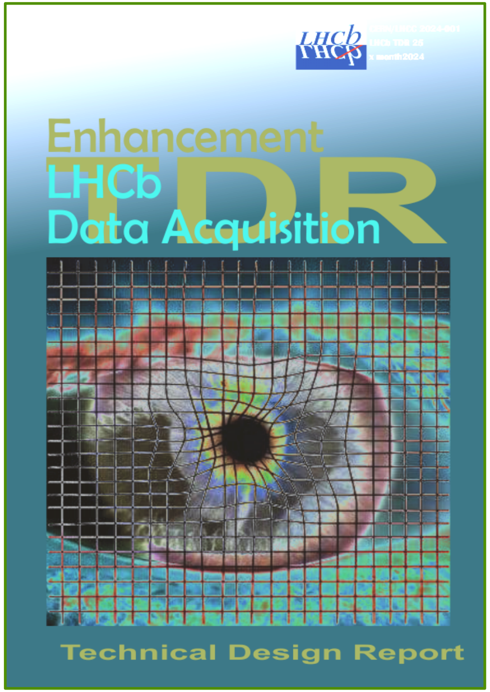
# Results on live LHCb data

- Running parasitically on real LHCb data during Run 3 physics data taking (at reduced rate)
- Online alignment constant applied on the fly.
- Tracks distribution from demonstrator (right) very similar to HLT2 output (left).
- Slight difference in distribution due to lack of the final fitting stage



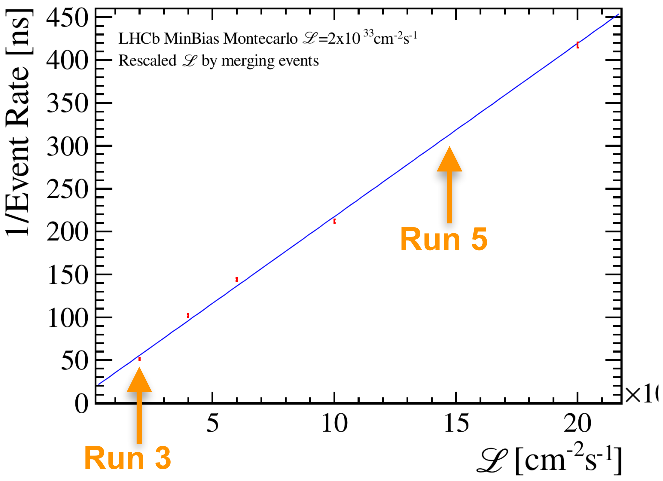
# DWT project: SciFi track primitives for LHCb

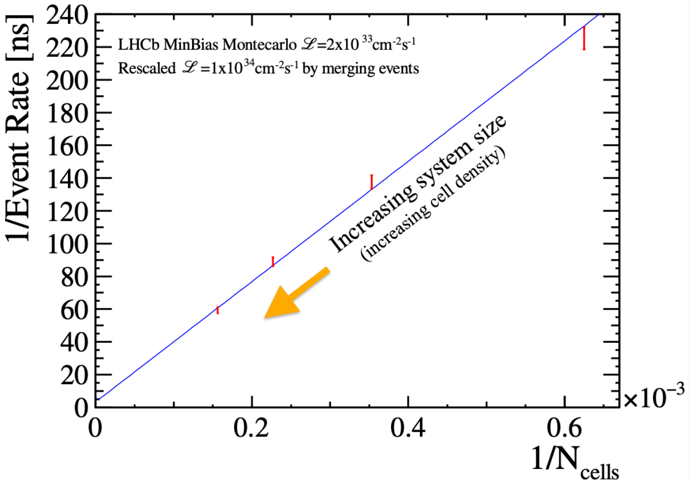



- Track segments in the SciFi detector play important role in LHCb
    - Currently used as 'seeds' for HLT1 tracking sequence
    - Heavy to compute (before GPUs only possible at HLT2)
- Implementation as 2-step retina device (axial layers, then stereo)
- Requires ~100 FPGA boards (new LHCb readout boards)
- Described in LHCb DAQ enhancement TDR for Run 4 (approved by LHCC)
- Fully detailed technical description available as a LHCb public note

# Future performance




- Characterized on **actual existing hardware** and data
  - Emulate higher luminosities by event overlapping
- Perfect **LINEAR** scalability with occupancy and size, up to highest densities
- Unique feature of this particular type of heterogeneous architecture
  - 'Standard' architectures $\propto N^n$

Ideal for high-rate applications - increasingly affordable with technological evolution

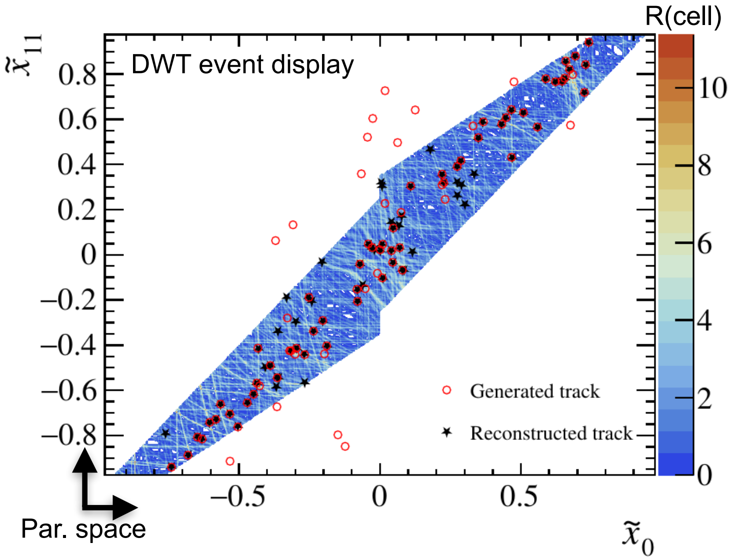
# Summary

- An architecture has been developed for more efficient pattern recognition, relying on strong connectivity and dataflow
- FPGA-based prototypes proved viability, and linear scalability
- Small application already operating in LHC Run 3
    - Extensive tracking project under construction for Run 4
- While initially thought for large-volume data, the prompt availability of pre-reconstructed primitives also enables precision applications

# Backup

# Emulation study of DWT performance

- Studies performed with realistic device Emulator, running on official LHCb MC productions.
- Tracking quality of primitives is at a level close to HLT1 - will be refined to tracks in HLT1 processing
  - <u>TDR initial:</u> Efficiencies ~90% , Ghost rates ~15%. **Now improved by further studies.**




| Track type | MinBias | $D^0 \to K^0_S\pi^+\pi^-$ | $B^0_s \to \phi\phi$ |
|---|---|---|---|
| Long, $p > 3\,\mathrm{GeV}/c$ | 85 (86) | 83 (84) | 84 (85) |
| Long, $p > 5\,\mathrm{GeV}/c$ | 90 (91) | 89 (90) | 89 (89) |
| Long from $B$ not $e^\pm$, $p > 3\,\mathrm{GeV}/c$ | - | - | 88 (87) |
| Long from $B$ not $e^\pm$, $p > 5\,\mathrm{GeV}/c$ | - | - | 90 (90) |
| Down, $p > 3\,\mathrm{GeV}/c$ | 84 (85) | 83 (84) | 83 (84) |
| Down, $p > 5\,\mathrm{GeV}/c$ | 89 (91) | 88 (89) | 88 (89) |
| Down from strange not $e^\pm$, $p > 3\,\mathrm{GeV}/c$ | - | 83 (83) | - |
| Down from strange not $e^\pm$, $p > 5\,\mathrm{GeV}/c$ | - | 88 (88) | - |
| Down from strange not long not $e^\pm$, $p > 3\,\mathrm{GeV}/c$ | - | 83 (83) | - |
| Down from strange not long not $e^\pm$, $p > 5\,\mathrm{GeV}/c$ | - | 88 (89) | - |
| ghost rate | 16 (10) | 17 (12) | 17 (13) |
| ghost rate / (1 - ghost rate) | 0.2 (0.1) | 0.2 (0.1) | 0.2 (0.1) |