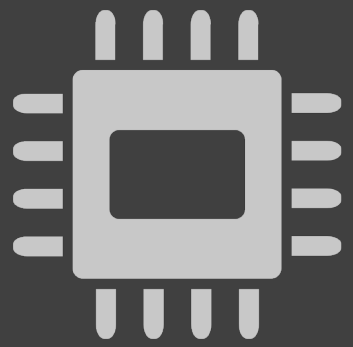
The FPGA design for the L0 trigger of the RICH detector of the NA62 Experiment at CERN SPS

FDF



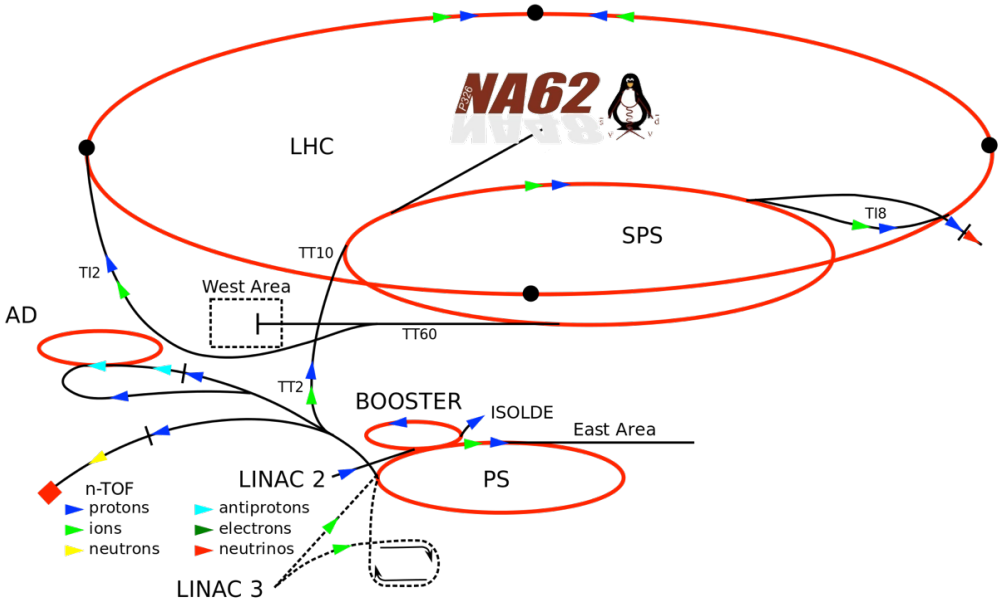
Mattia Barbanera (INFN Perugia)
mattia.barbanera@pg.infn.it

FDF Seminar

26.11.2024

Table of Contents

Some Context: The NA62 Experiment



NA62 Detector Layout and Principles

Measure the $K^+ \to \pi^+ \nu\bar{\nu}$ BR to 10% precision collecting O(100) events

- Decay occurring with 10^{-12} probability

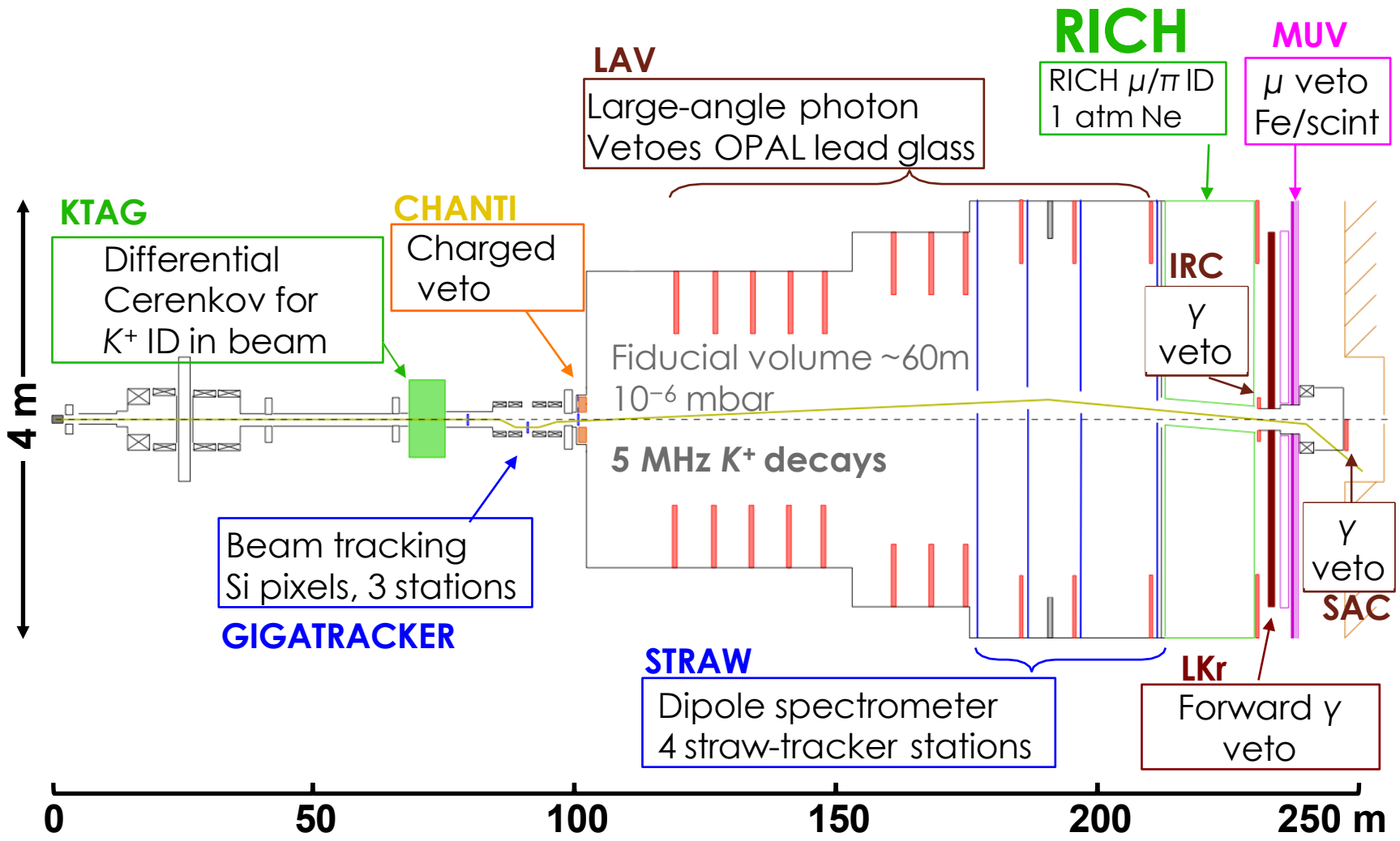
Identify particles

- High-performance EM calorimeter
- Redundant particle ID for e/μ/π
- Hermetic photon vetoes

Measure crossing time

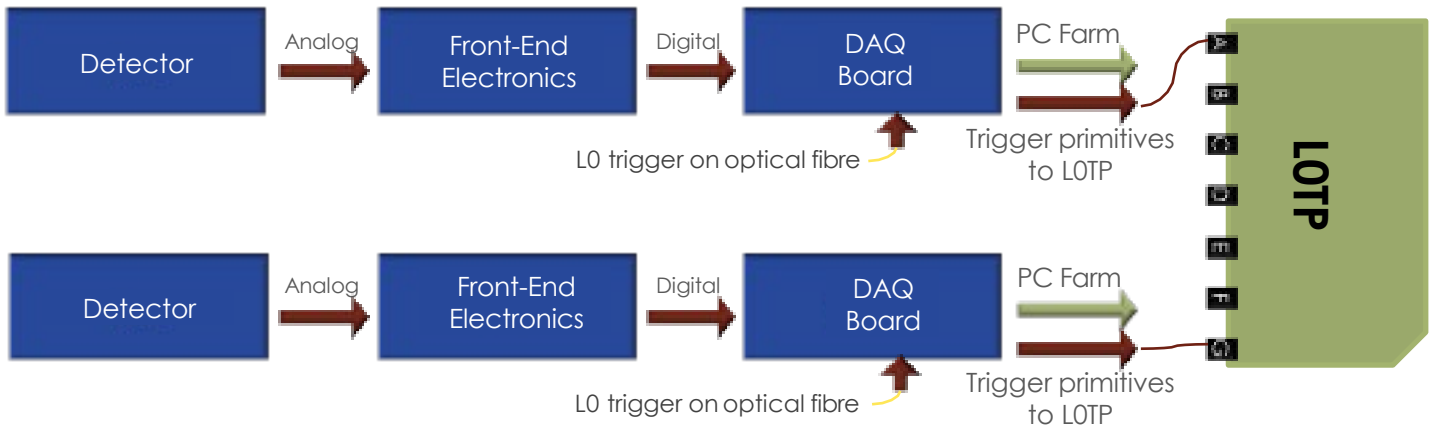
- Resolution better than 100 ps
- High-rate 10 MHz events

Precision tracking



Trigger and Data Acquisition

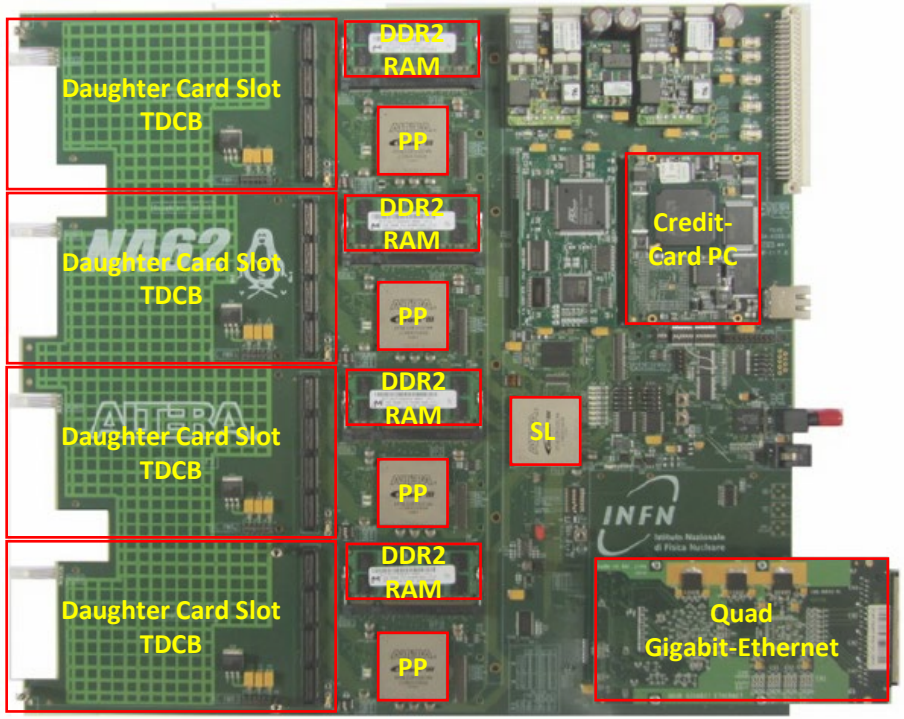
- Three-level trigger to reduce the amount of stored data
  - 10 MHz events → 10 kHz
  - Level-0 (L0): hardware trigger; reduction factor 10
  - Level-1 and Level-2: software triggers; reduction factor 10 each
- L0: a subset of detectors produces fast and small-sized information
  - **Trigger primitives** to be sent to L0TP
    - Timestamp information: ~100 ps resolution
    - Primitive ID: type of event detected (specific for each detector)
- L0TP produces L0 triggers if predefined conditions are satisfied
  - Some detectors using TEL62 have dedicated GW creating trigger primitives



Data Acquisition: TEL62 Read-Out Board

Common read-out board shared by many detectors: **TEL62**

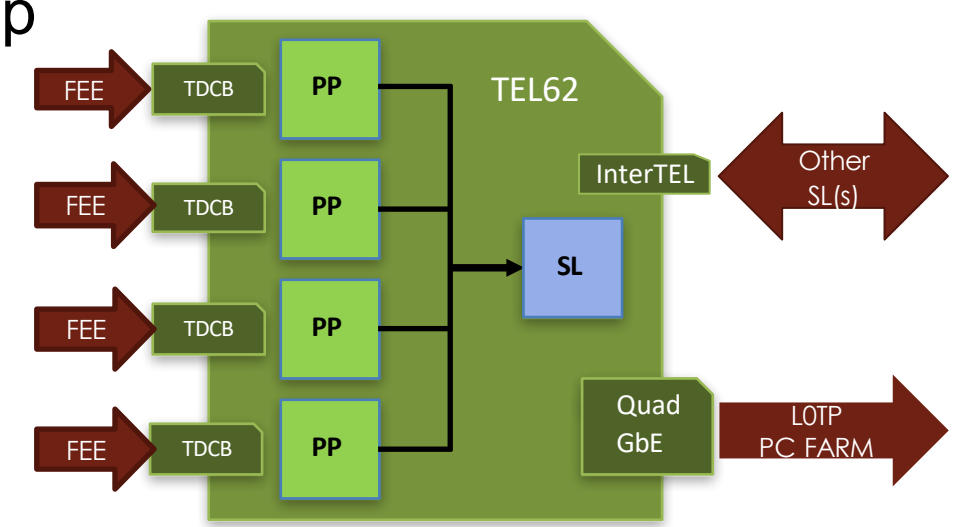
- 4 Daughter Card Slots
  - TDCB (Time-To-Digital Converter Board)
  - Dedicated FEE (Front-End Electronics) for each subdetector
- 4 Pre-Processing Stratix-III FPGAs
  - Receive data from the 4 TDCBs
  - 2 GB data buffer
- 1 Sync-Link Stratix-III FPGA
  - Combines the inputs from the 4 PPs
  - Sends data to PC farm and trigger processor
- 1 Credit-Card PC
  - Configuration and monitoring interface



RICH-LO GW: Working Principles

- TDCBs produce timing information of particles interacting in the RICH
  - 40-bit timestamp
    - ~100 ps LSb (resolution), ~60 s MSb

- GOAL: group hits and provide a precise time reference for trigger
  - Exploit the full ~100 ps resolution of the timestamp
  - Hits belong to the same Cherenkov circle



- PPs perform a preliminary clustering
- SL merges the clusters from the PPs
  - And from InterTEL, if applicable
- SL produces primitives to send to the L0 Trigger Processor

RICH-LO GW: Requirements and Working Principles

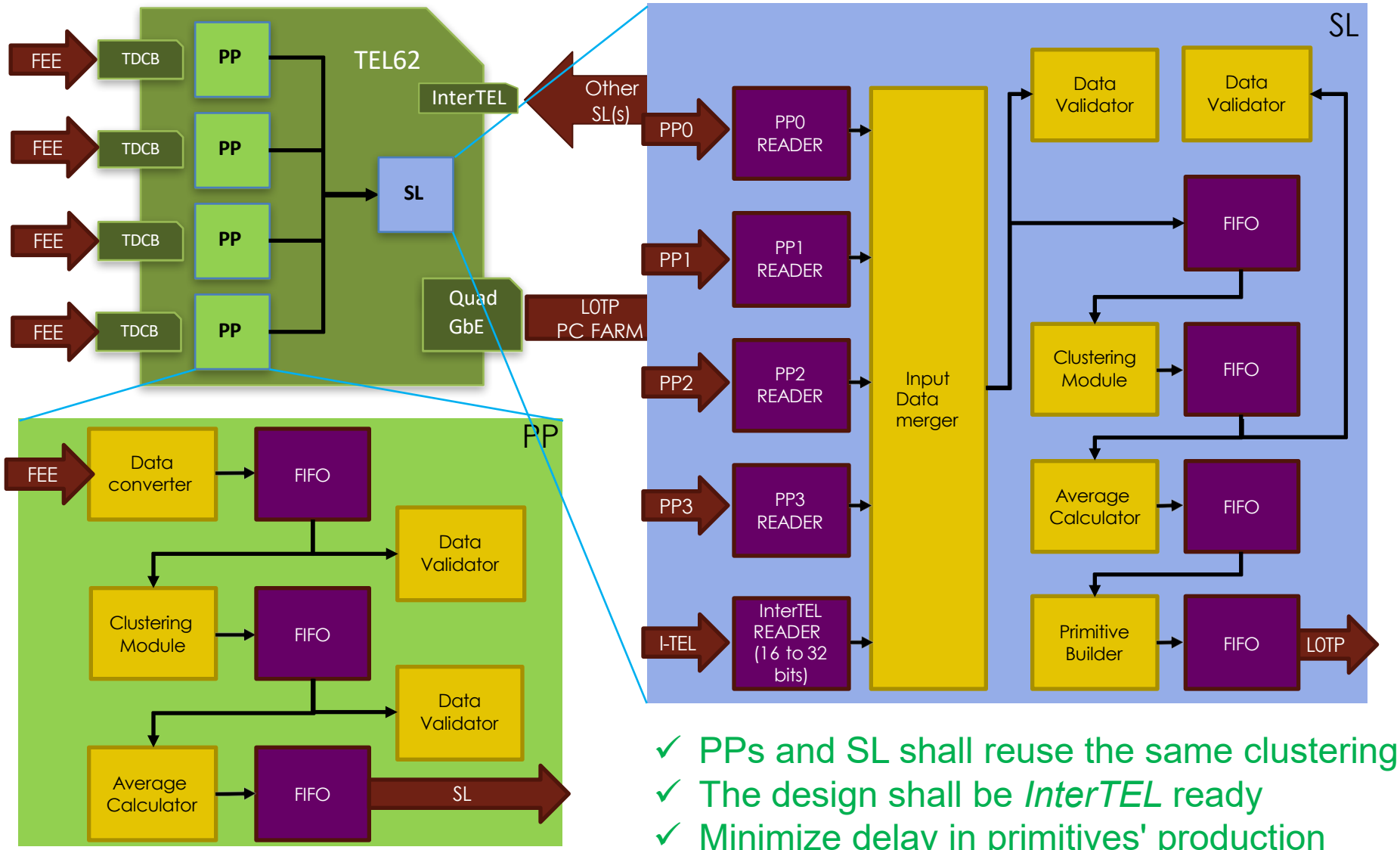
REQUIREMENTS	WORKING PRINCIPLES
• Primitives shall be ordered in time down to 100 ps LSb • TDCBs do not produce time-ordered data	• Exploit 25-ns ordering of read-out modules

• Minimize delay in primitives' production
  • Maximum 32 µs, or 5 time-frames of 6.4-µs
  • 6.4 µs: basic time-division of the experiment
  • Translates in: throughput of 1 word per clock cycle
	• All modules shall use a common data format as input and output
	  • Reusability and compatibility

• Minimize logic occupation
  • Already high due to read-out modules
	• PPs and SL shall reuse the same clustering module
	  • Reusability and compatibility

• The design shall be *InterTEL* ready
  • 16-bit bus
  • Daisy-chain to improve trigger accuracy and robustness	

RICH-LO GW: Block Diagram



- ✓ PPs and SL shall reuse the same clustering module
- ✓ The design shall be *InterTEL* ready
- ✓ Minimize delay in primitives' production
  - ✓ If modules process 1 word per clock cycle

Common RICH Format for Primitives and Clusters

- 2 types of 32-bit words
  - **Timestamp**: Count time in 28 bits
    - Resolution of 400 ns, up to ~60 minutes
  - **Data**: Describe cluster information
    - Fine-time timestamp: resolution of 100 ps
    - Number of hits
    - Cluster Time-Sum
- Separable 2x16-bit words for *InterTEL*
  - Each word identified by 2x2 bits

Data converter

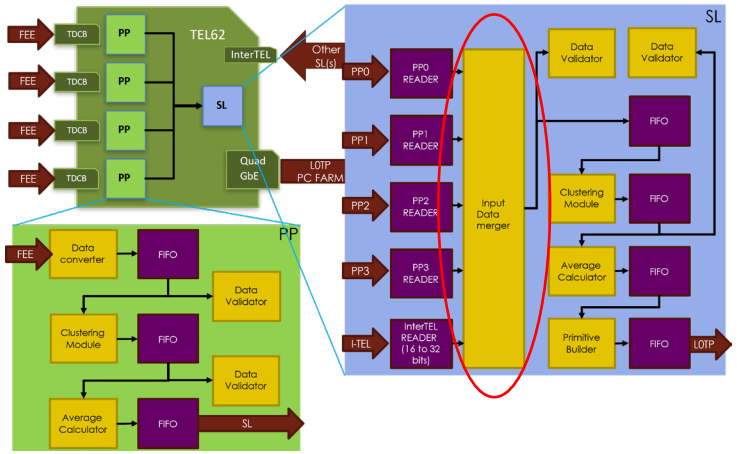
- Reads TDCBs data and converts them into *RICH format*
  - Interpreted as clusters
  - $N_{hits} = 1$, CTS = 0
- 16 timestamps per 6.4 μs time slot
  - If no data in a timestamp, produces fake clusters (speed-data)
    - $N_{hits} = 0$, CTS = 0

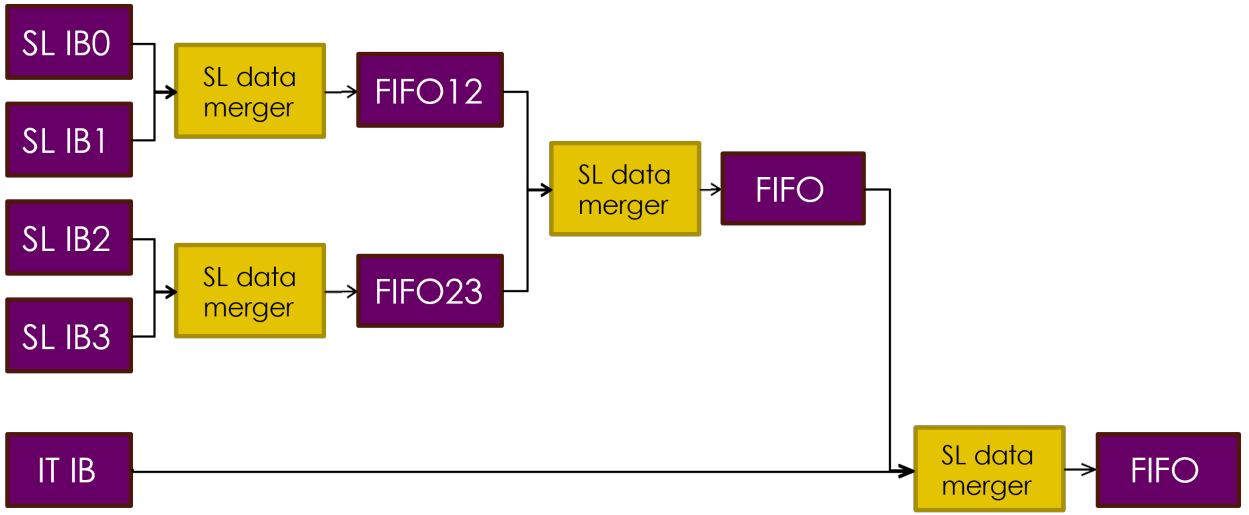
31 30	29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14	13 12	11 10 9 8	7 6 5 4 3 2 1 0
T.S. 1/2	Timestamp (27:14) MSb ~= 60 s, LSb =400 ns		**T.S. 2/2**	Timestamp (27:14) MSb ~= 60 s, LSb =400 ns		
Data 1 1/2	N_{hits} (7:2)	Cluster Time-Sum (7:0) (signed) LSb = 100 ps	**Data 1 2/2**	N_{hits} (1:0)	Fine Time (11:0) LSb = 100 ps (up to 400 ns)	
Data 2 1/2	N_{hits} (7:2)	Cluster Time-Sum (7:0) (signed) LSb = 100 ps	**Data 2 2/2**	N_{hits} (1:0)	Fine Time (11:0) LSb = 100 ps (up to 400 ns)	

✓ All modules shall use a common data format ✓ The design shall be *Inter-TEL* ready

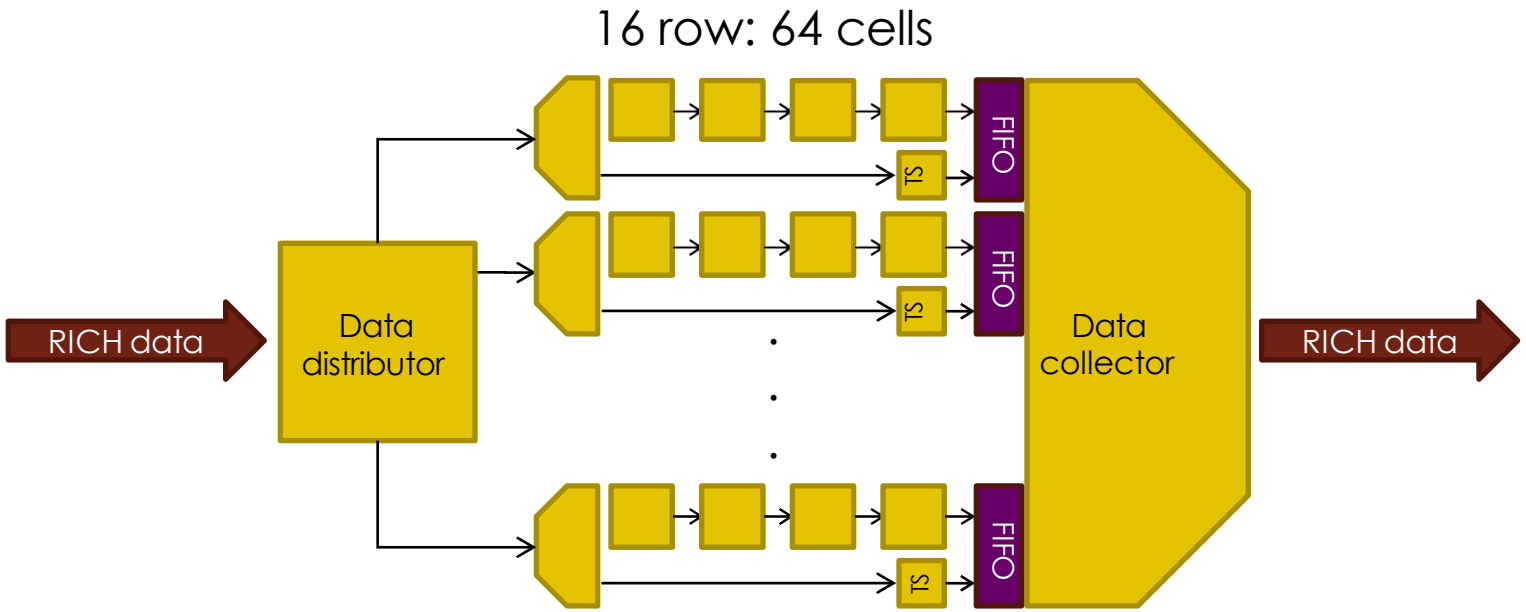
SL Input Data Merger

- `SL data merger` merges the clusters from two sources
  - Purely combinatorial
  - Clocked at the same frequency of the other modules (160 MHz)
- Waits that both its input FIFO are not-empty
  - Consistency in data ordering
  - **WARNING:** Starvation of data in FIFOs
- Produces data sorted in frames of 25 ns
  - Same as PPs input
  - No replication of TS or speed-data





Clustering Module: Working Principles



16 rows

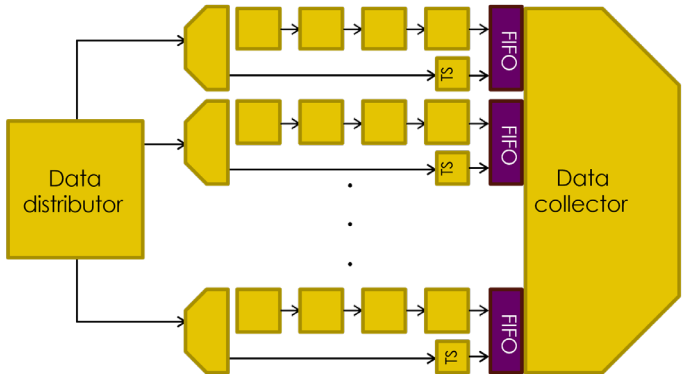
- Each row takes in input hits belonging to a specific 25-ns frame
  - Enough rows to avoid waiting for an empty slot

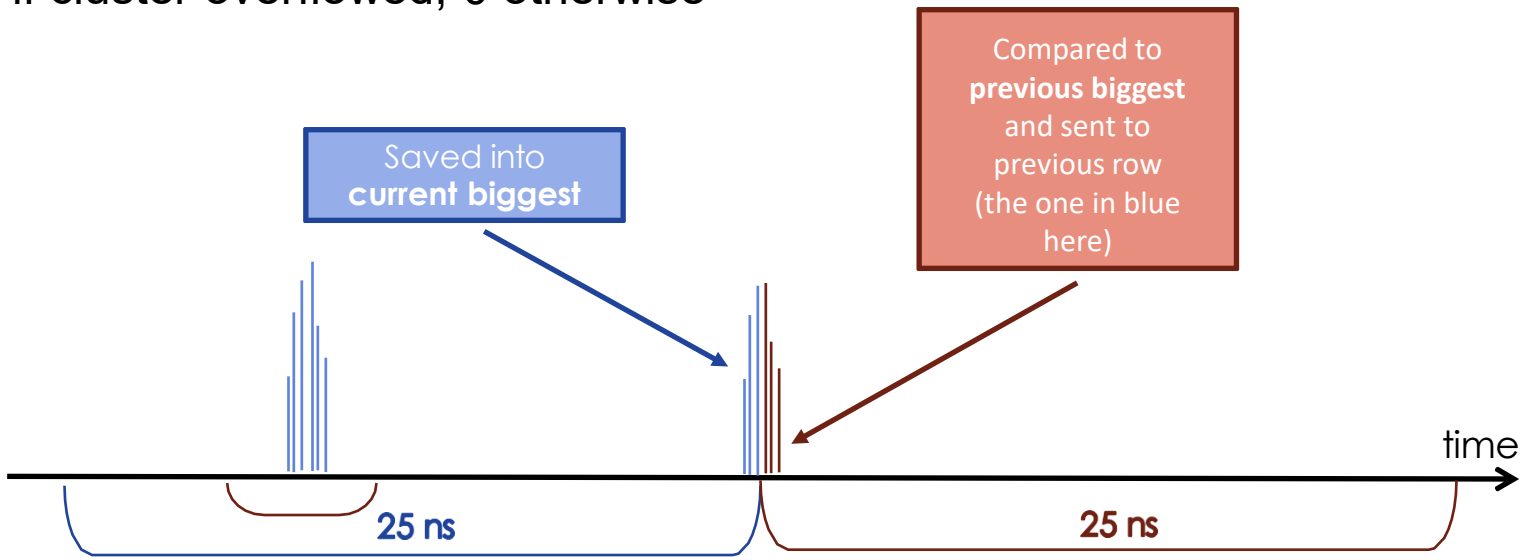
4 cells per row

- Each cell creates a cluster of hits (or clusters) if closer than a programmable time value
- Discard clusters if more than 4 in a timestamp
  - Instantaneous clusters rate of 160 MHz

Clustering Module: Data Distributor

- Rearrange input data into
  - 25-ns timestamp, 32 bits
  - 100-ps fine time, 8 bits
- Deliver data to the proper row
  - Splits in 25-ns time slots
- Handle corner cases of clusters spanning across two 25-ns time slots
  - 1 additional bit to signal this occurrence
  - 100-ps fine time uses 9 bits
    - 9th bit is 1 if cluster overflowed, 0 otherwise

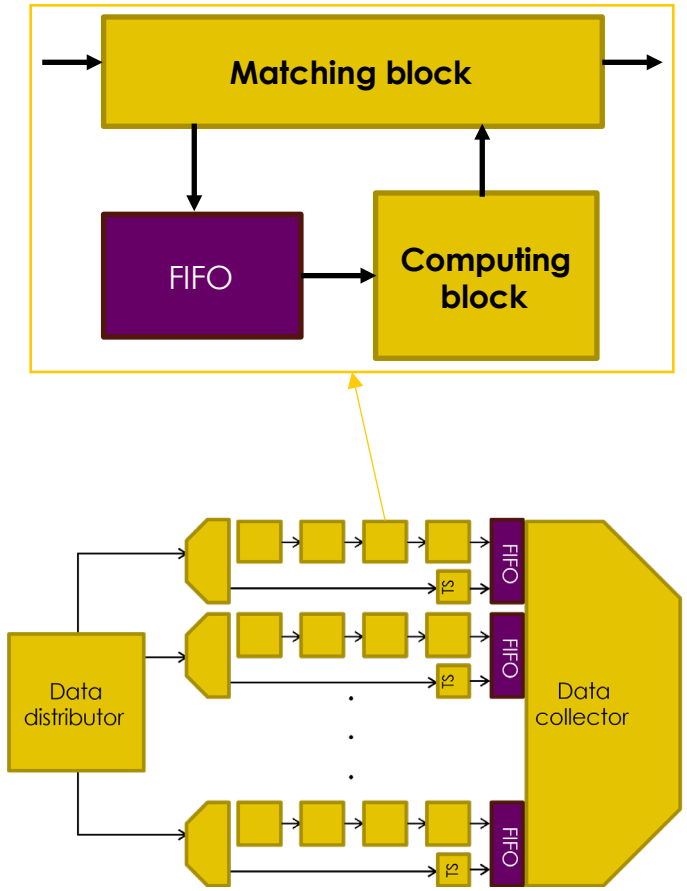




Clustering Module: Cell

When a cell receives a hit (or cluster)

- If empty
  - Store $Time_0$ as seed of the cluster
- If not-empty and inputs do NOT match
  - Pass the cluster to the adjacent cell
- If not-empty and input $Time_1$ matches $Time_0$
  - Merge the two clusters
    - $CTS_0 \mathrel{+}= N_1(Time_1 - Time_0) + CTS_1$
    - $N_0 \mathrel{+}= N_1$
  - Programmable time-matching window



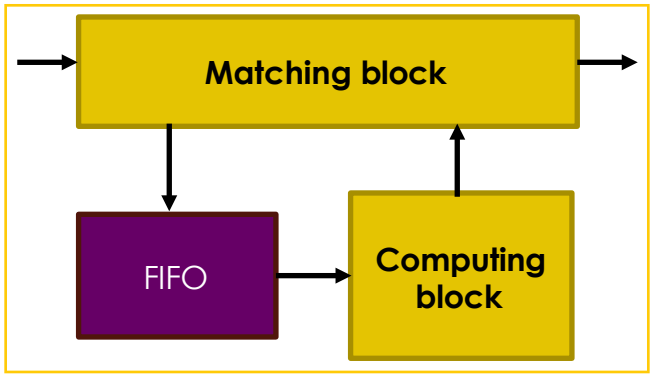
Each computing block embeds a DSP multiplier

- FIFO to compensate for multiplier delays

Clustering Module: Cell

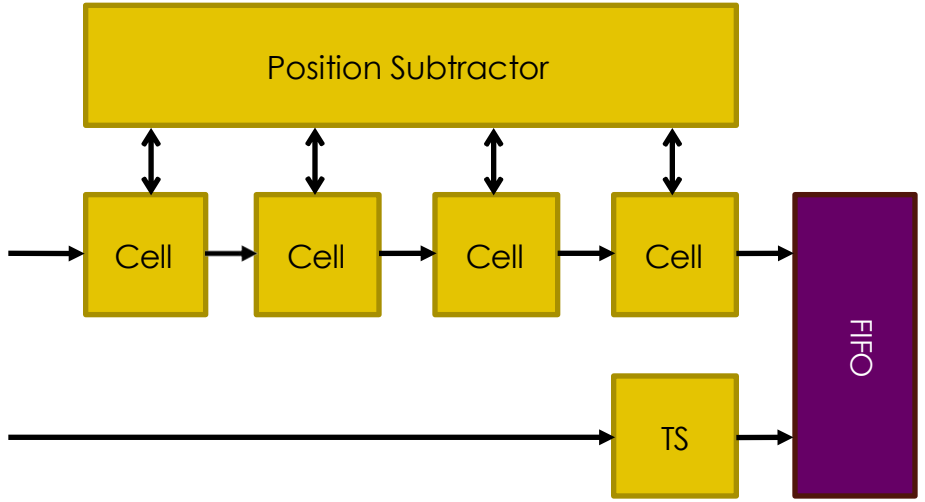
Data distributor flushes a $(n-2)^{th}$ row when filling the n^{th}

- In flush mode, row acts as a shift register
  - Cells output N_{hits}, T, and CTS to the right
  - Data go to the Data collector through a FIFO



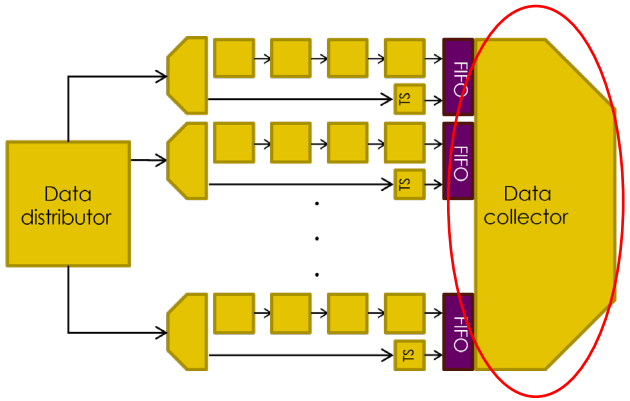
To sort clusters, cells have 8-bit position field

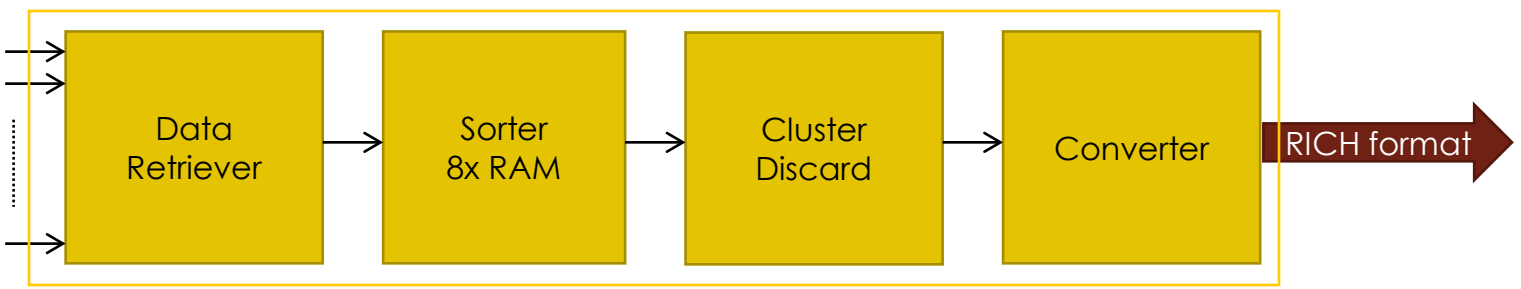
- Used by Data collector to quickly sort clusters
- To sort clusters:
  - Matching cluster in input:
    - Do nothing
  - Greater cluster in input:
    - Increase its position field before forwarding to next cell
  - Smaller cluster in input:
    - Increase the position of the stored cluster
    - **WARNING:** What if the cluster overflows the 4 cells?



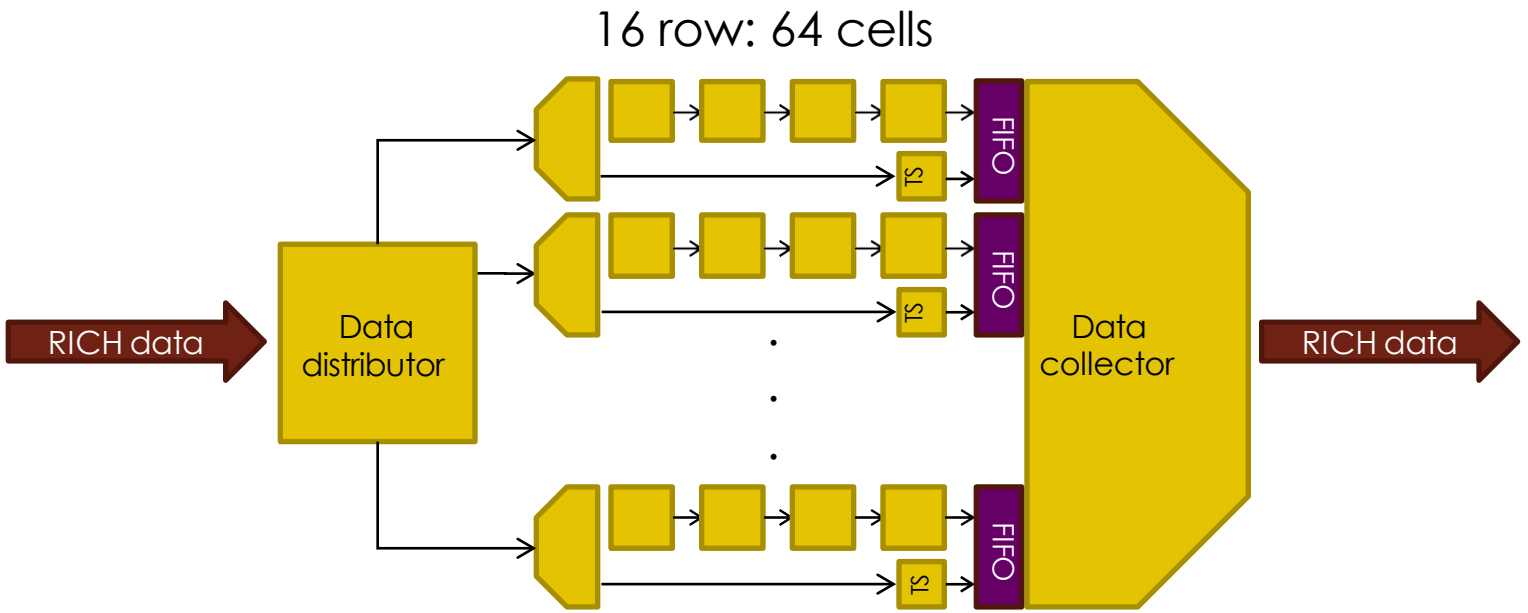
Clustering Module: Data Collector

- Data retriever
  - Reads data from rows' FIFOs
  - Checks empty signals of two consecutive rows
    - Goal is to read 1 word per clock cycle
- Sorter
  - Writes clusters by addressing a RAM with the cells' position field
  - 8 RAMs of 4 positions each
- Cluster Discard
  - Discard clusters with multiplicity out of a programmable range
  - Reduce noise from events with multiplicity at extremes (low or high)
- Formatter
  - Formats data in *RICH format*





Clustering Module: Sizing



- GOAL: throughput of clustering module must be 1 word per clock cycle
- 16 rows design:
  - Two rows concurrently filled
  - Once the second row is completed, the first can be read-out
  - The latency of a row is given by
  - $\mathrm{L} = 2 \cdot \langle \text{row depth} \rangle + \langle \text{multiplier latency} \rangle + \langle \text{FIFO latency} \rangle$
  - $\mathrm{L} = 2 \cdot 4 + 3 + 3 \ = 14$

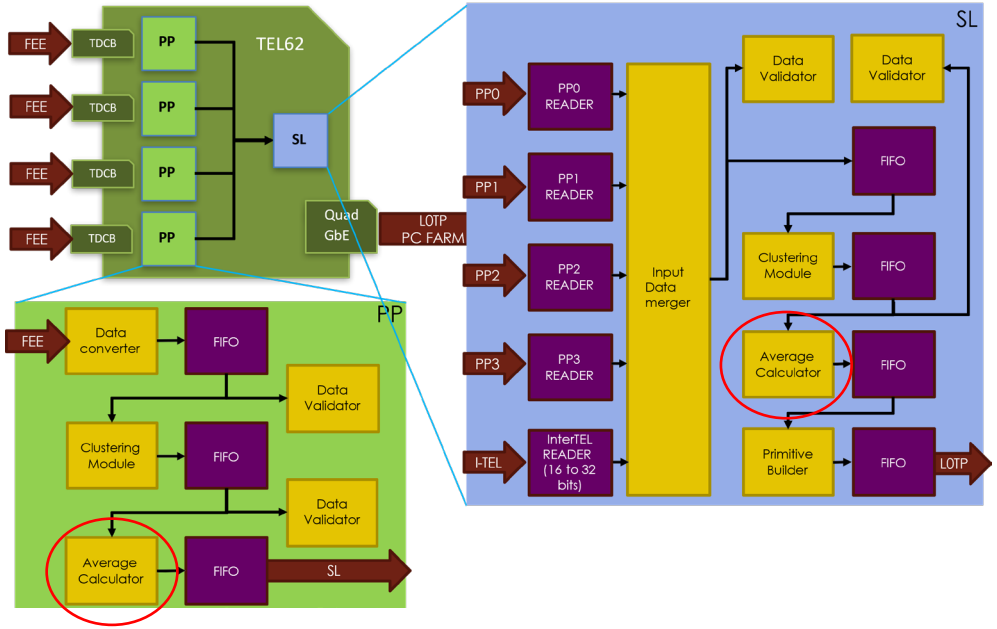
Average Calculator

CTS field to compute geometric average of cluster time

- Each time the clustering module has a data word in input, the sum is updated:
  - $CTS_{new} = CTS_{old} + N_{new}(t_{seed} - t_{input})$

- The average calculator computes the reference time as
  - $Seed_{new} = Seed_{old} + CTS_{old}/N_{old}$
  - $N_{new} = N_{old}$
  - $CTS_{new} = 0$

- Last module in PPs
  - Pre-clusters with more precise reference time
  - Avoid overflows of CTS field in the SL
- Last module in SL
  - Compute the cluster reference time for primitives

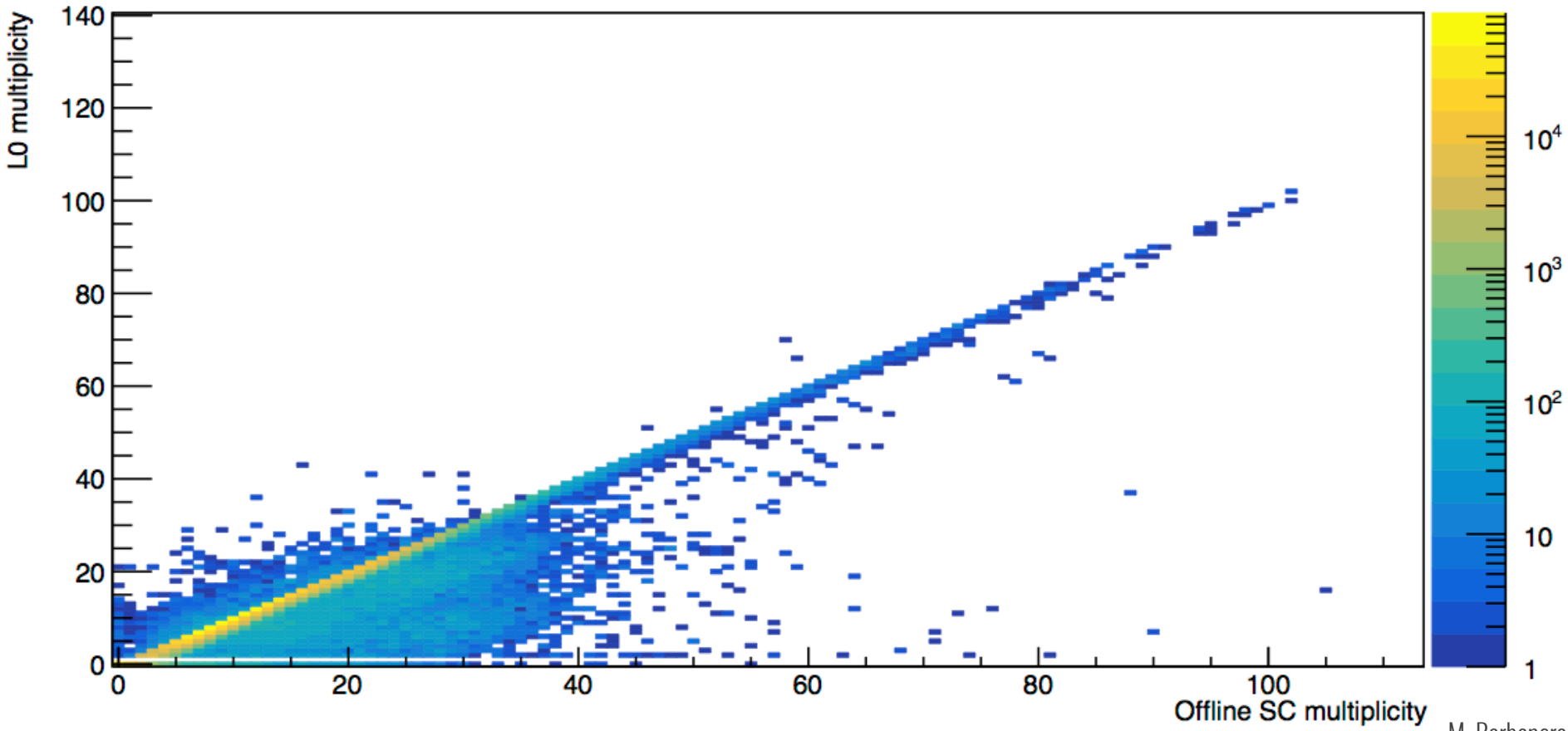


Multiplicity: Gateware vs Offline

Dedicated runs to verify design

- Data triggered by other detectors
- Correlations between primitives' time and offline-computed hits time

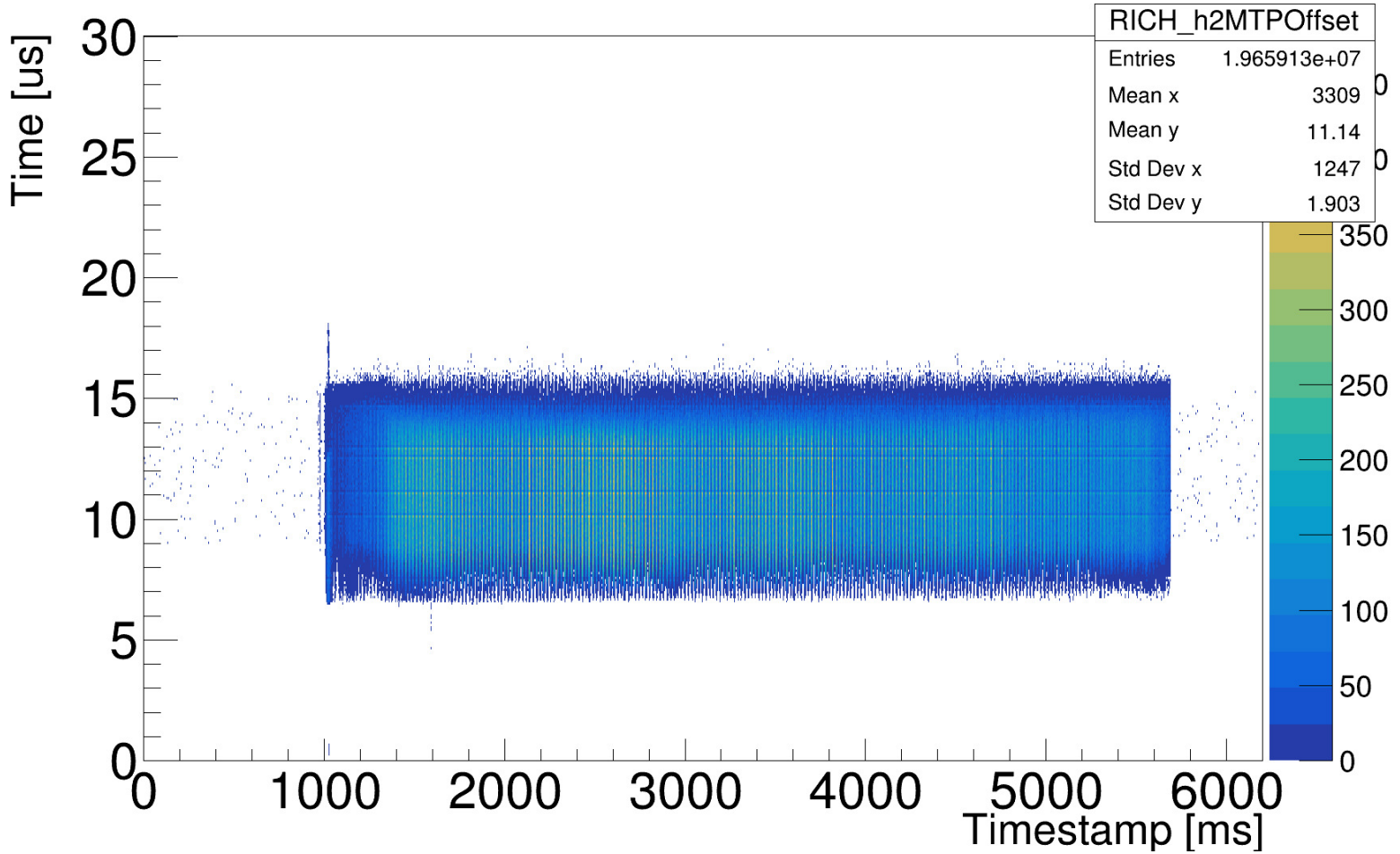
- Inefficiency: 1.2%
  - `L0 mult. = 0, offline mult. != 0`
- False positive: 0.005%
  - `L0 mult. != 0, offline mult. = 0`



Delay in Primitives' Production

Delay computed as

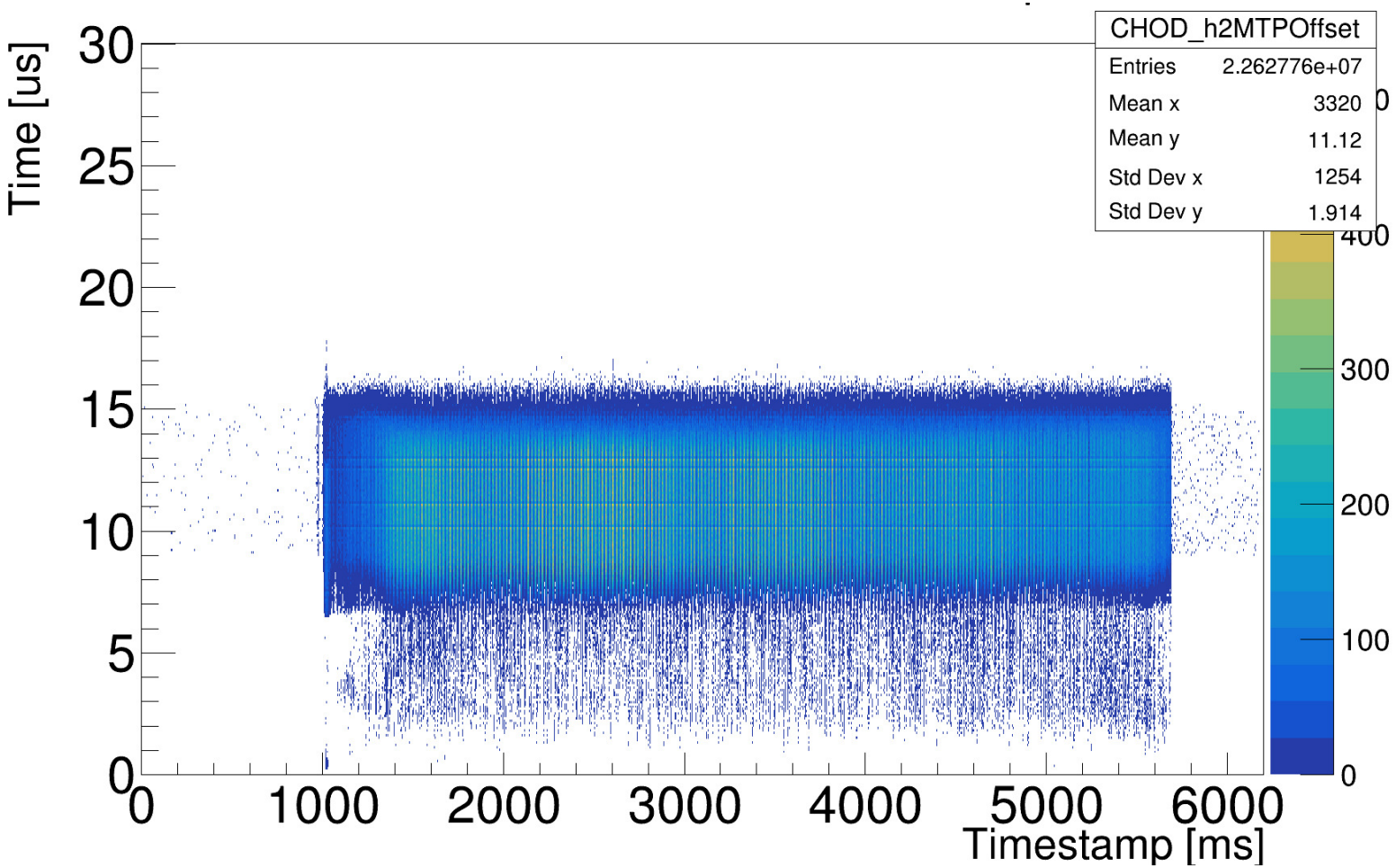
- `Time written in a primitive` vs `Time of production of the primitive`
- Stable and between 2 and 3 time-frames of 6.4 μs each



Performance in Other Detectors L0

No detector-specific information: GW used also in L0 of CHOD detector

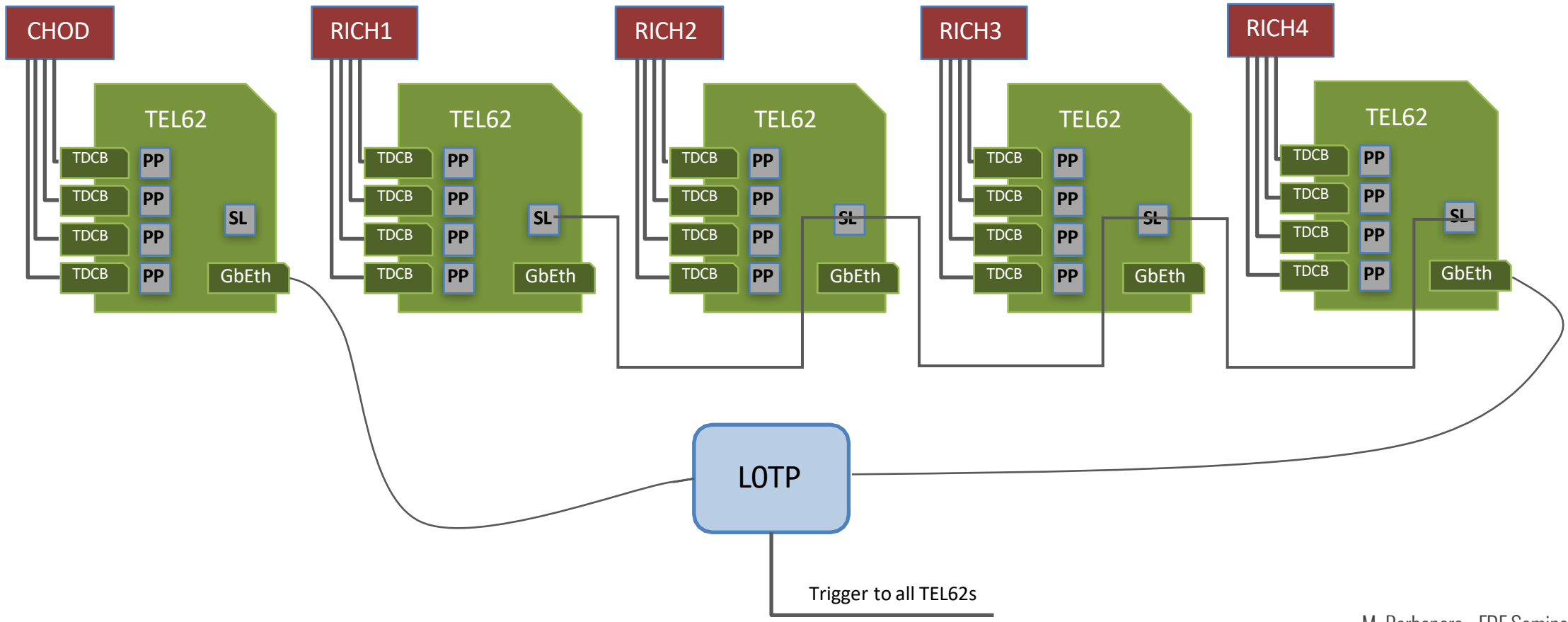
- Same results in efficiency
- With a higher rate (~15%), the delay diminishes



Ready for use with *InterTEL* Boards

RICH format and architecture allows to use InterTEL Boards

- Picture shows a possible use-case
- In the daisy-chain, only the last TEL62 sends primitives to the L0TP



Conclusions

RICH-L0 GW produces cluster of hits providing a precise time reference for the L0 trigger of the NA62 experiment

ACHIEVEMENTS	COMMENTS
Stands the full rate of detectors	Actual rate twice the one of simulations
	Efficiency of 98.76%
Primitives time-ordered down to 100 ps	
Maximum delay of 3 time-frames • Lower than the required 5 time-frames	The higher the rate, the lower the delay • Up to the saturation of the GbE links
GW fits the PP and SL FPGAs	No timing violations, full-scale clock frequency
No detector-specific information used	Employed also in the L0 of CHOD detector
InterTEL ready	All modules use the common RICH format