International conference on Technology and Instrumentation in Particle Physics, 24-27 May 2021

Muon Trigger with fast Neural Networks on FPGA, a demonstrator

M. Migliorini[1], F. Marini[1,2], A. Triossi[3], J. Pazzini[1,2], M. Zanetti[1,2], A. Zucchetta[1]

[1]INFN Padova, [2]University of Padova, [3]IPHC Strasbourg

Istituto Nazionale di Fisica Nucleare

Università degli Studi di Padova

Introduction: Use Cases

- Muon detectors are pivotal in several particle physics use cases:
  - Muon tomography is centered on efficiently detecting and tracking muons
  - In HEP colliders, muon final states often provide golden channels for rare processes

- Muon local trigger algorithms are among the first stages of online event selection, often having to cope with demanding conditions:
  - Background noise and large detector occupancy
  - Short available time for trigger decision

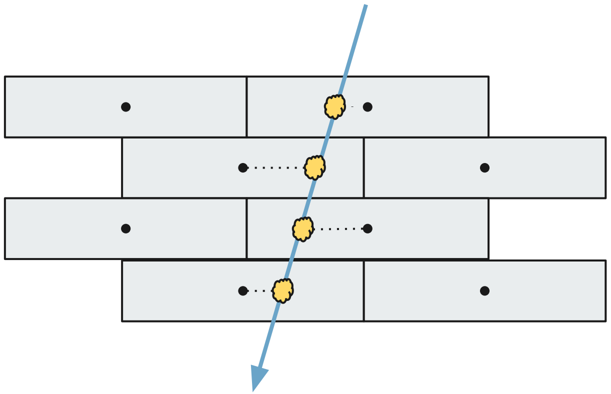
- Implemented an algorithm aimed at processing measurements of DTs and based on a **FPGA implementation of Neural Networks** for a fast local muon trigger

- Algorithm tested on a dedicated simulation and on real data acquired from a muon telescope installed in the INFN National Laboratory of Legnaro (LNL)

Introduction: miniDT

- Reduced area Drift Tubes detectors assembled and commissioned in LNL
  - Used for the test-beams of the LEMMA collaboration
  - Test-bed for multiple application related to DAQ and electronics

- Each miniDT is composed of 4 layers of cells (tubes) arranged with ½ cell staggering to allow an estimation of the muon track
  - 16 (42x14 mm^2) cells per layer
  - A total of ~70x70 cm^2 active area per chamber
  - Filled with an Ar-CO_2 (85/15%) gas mixture
  - Uniform electric field inside the cell providing a constant drift velocity



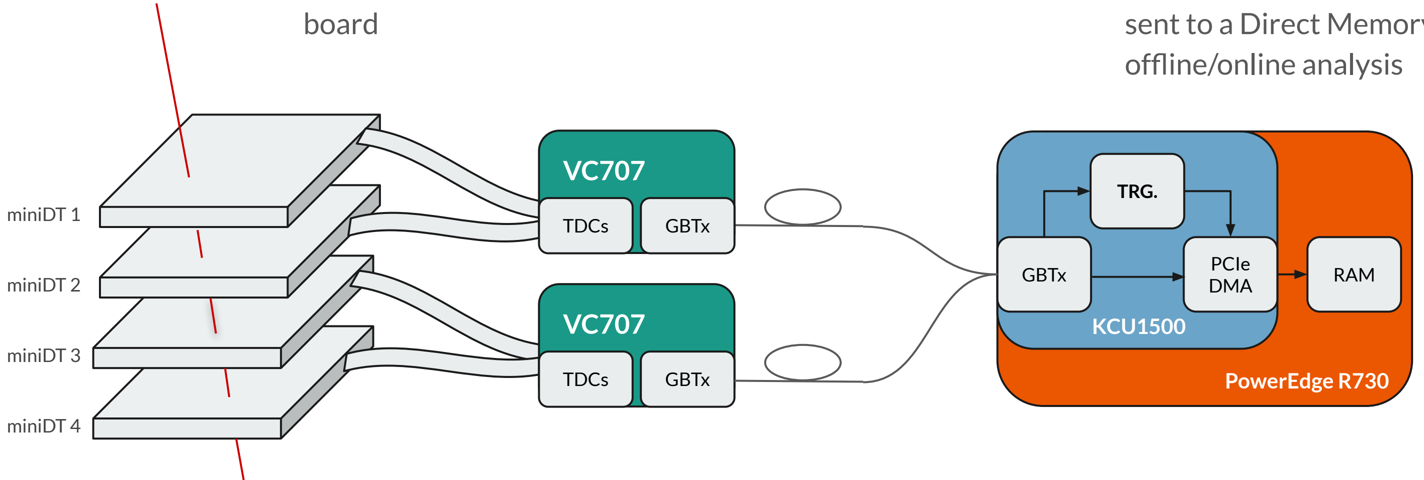
- Electrons avalanche produced in each cell by the passage of the muon
  - Charge collected by the wires

- Mean-Timer algorithm allows to determine the muon passage time without the need of and external trigger
  - Constant drift velocity allows to find the x coordinate
  - Track parameters (slope, intercept) can also be obtained

Introduction: Readout Electronics

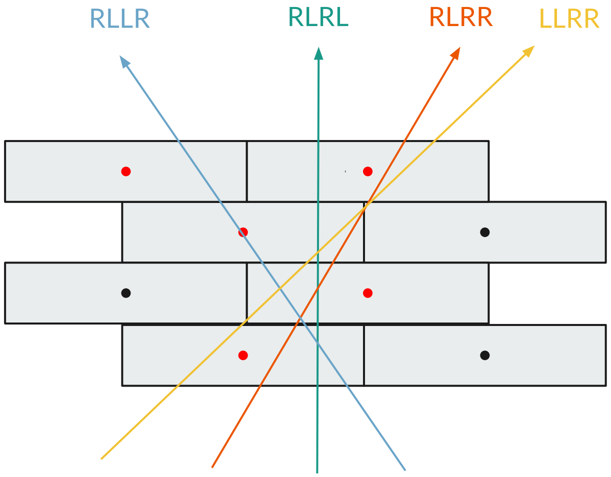
Signals produced by the e^- avalanche are amplified, shaped, and discriminated by custom ASIC chips in the Front-End electronics

- Two Xilinx VC707 evaluation boards, hosting Virtex-7 FPGAs:
  - Each VC707 receives signals from 128 channels (2xminiDTs)
  - Time-to-Digital Conversion (TDC) implemented on FPGA
  - Former TDC are serialized with the GBTx-FPGA protocol and sent to the next board
- One Xilinx KCU1500 evaluation board, hosting a Kintex Ultrascale FPGA:
  - Receives up to 8 GBTx links
  - Its firmware processes the streams of the entire set of TDC hits from all miniDTs
  - Results of the trigger re-injected into in the data streams
  - Streams are merged in a single stream and sent to a Direct Memory Access engine for offline/online analysis



Algorithm: role of Neural Networks

- Neural Networks excel at solving complex tasks based on a proper training
- Once the NN structure and its weights are defined, the evaluation can be fast
    - Fixed latency regardless of the inputs
    - Only multiplications and additions performed: suitable for a FPGA implementation
- Neural networks can be adopted to avoid the combinatorial
    - In the mean-timer algorithm all hits quadruplets and lateralities need to be probed
    - Only the one providing a valid result is considered
- Different conditions of the detector can be reflected by simply re-training the models



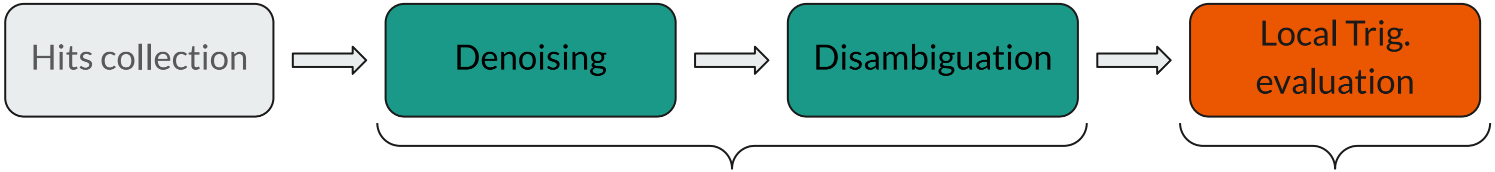
All lateralities need to be probed to find the correct track for each group of 3/4 wires

Algorithm: a Hybrid Approach

Local trigger algorithm based on an “hybrid approach”:

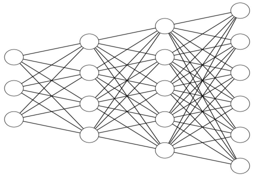
1. Avoid combinatorial by selecting all and only hits from genuine muons
    - Steps performed using neural networks

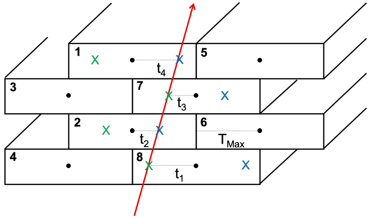
2. Apply analytical relations only on the selected hits
    - Mean-timer equations and least-squares fit



Identify hits and solve L/R ambiguity with neural networks → avoid combinatorial

Find TP using only the correct combination of hits and lateralities





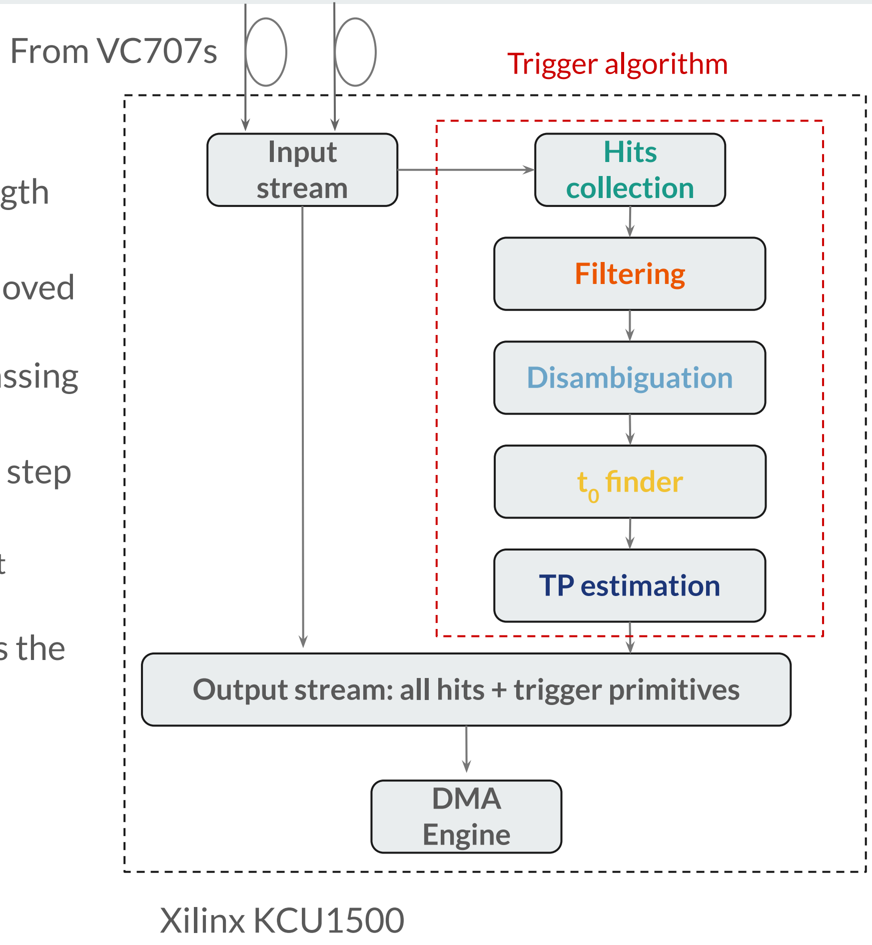
Algorithm: FW implementation Overview

Building blocks of the algorithm:

- Hits collection: stream of hits persisted inside a fixed-length time window
- Filtering: collected hits are filtered, i.e. noise hits are removed
  - If 3/4 hits are found, next steps are triggered
- Disambiguation: solve laterality ambiguity for the hits passing the filter
- T_0 finder: using the information from the disambiguation step find the crossing time using mean-timer technique
  - Apply the correct equation without probing all the different combinations
- Track parameters estimation: given t_0 and hits lateralities the points can be placed in the space and perform a linear fit

All hits are stored regardless the trigger decision:

- Offline evaluation of the performances

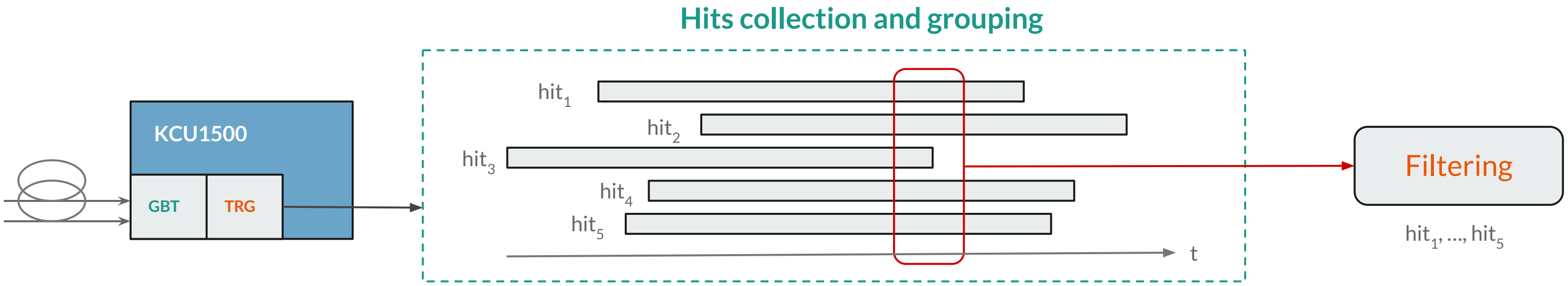


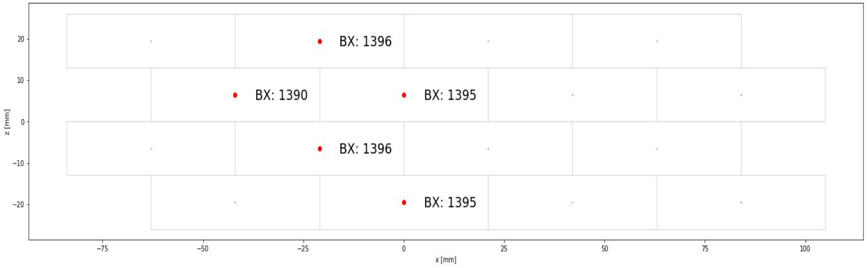
Algorithm: Hits Collection

Hits are collected with a sliding window with a persistence of 30 BXs

- Denoising is activated when a hit reach the end of it persistence
- Sub-patterns (e.g. 3-plets out of hits from 4-plet) are not re-evaluated

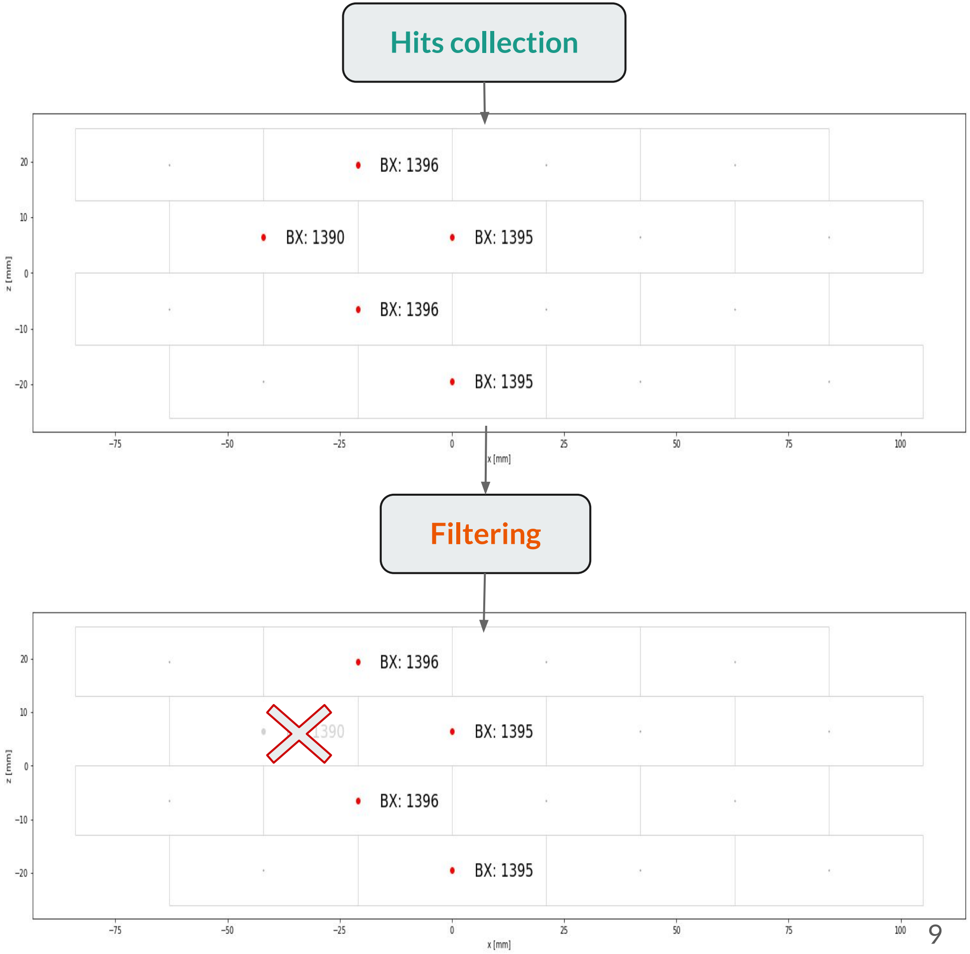
All patterns found at this grouping stage are sent to the denoising neural network





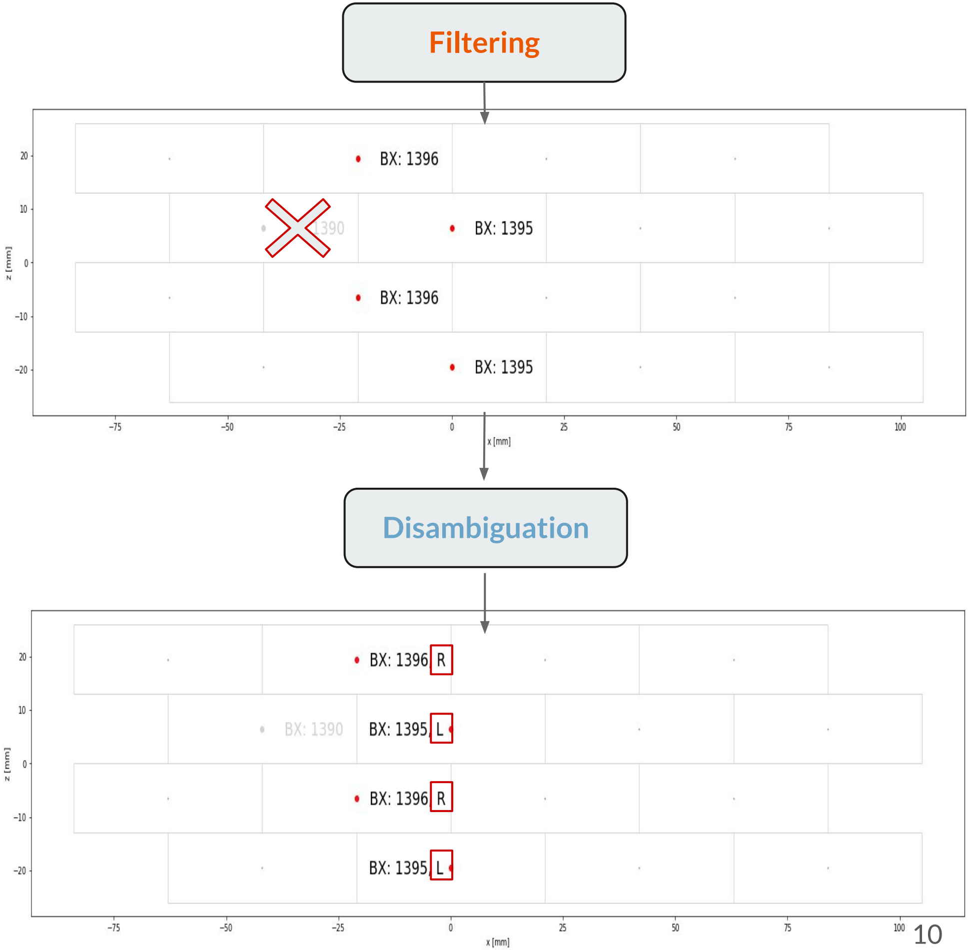
Algorithm: Denoising

- Denoising is performed with a NN to remove hits not compatible with a muon (time/space)
- Hits cell and time are used as inputs of the NN
- Outputs are signal/noise flags
- HLS4ML used to convert Keras model to HLS
- Tuning is needed to optimize the accuracy vs resource utilization
  - NN architecture
  - Weights quantization
  - Model pruning
- 3 layers dense neural network
  - 6 bit weights quantization, 50% sparsity



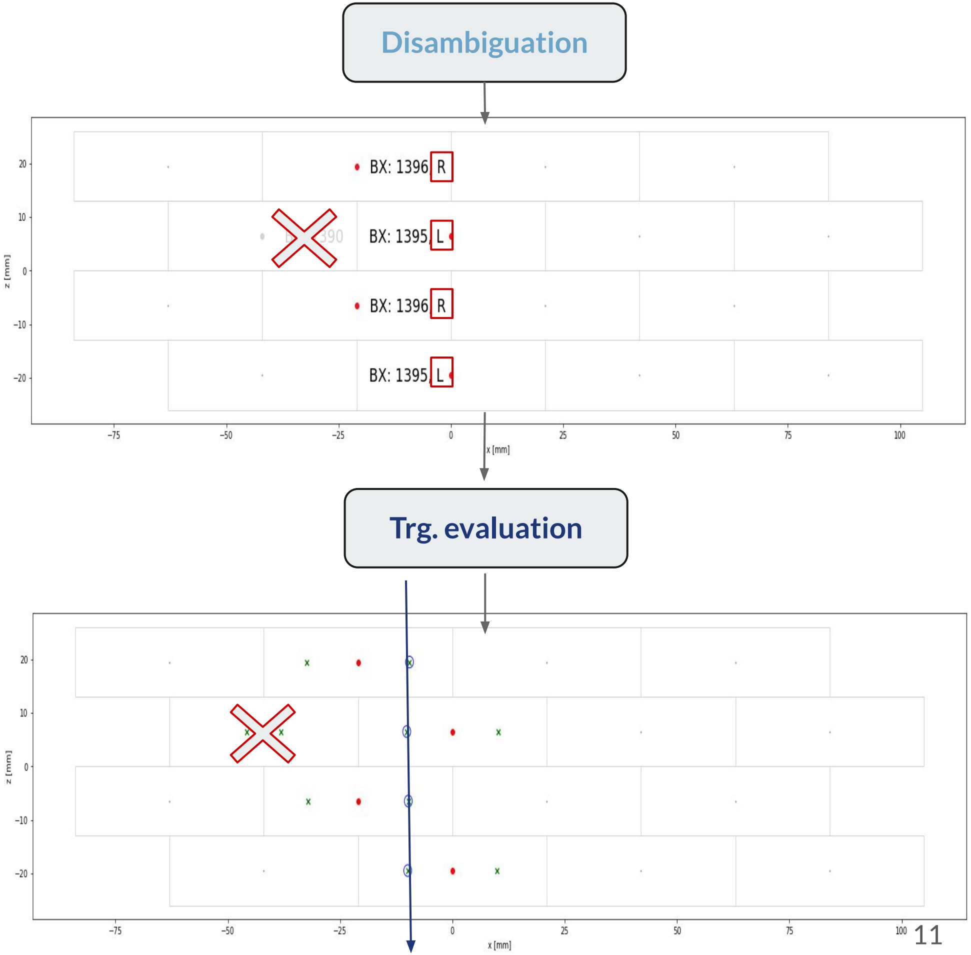
Algorithm: L/R Disambiguation

- A second NN resolves the laterality ambiguity directly after the denoising stage
  - Predict if track passed on the right/left of the wire
- Disambiguation network takes as input the 3/4 hits found by the filtering step
  - Number of input neurons is fixed to 4, one possible hit per layer
  - In case of missing hit a padding value is provided
- Input hits are classified as Left/Right
  - 0 if either the hit is missing in that layer or it has been wrongly classified as signal by filtering network
- 3 layers dense neural network
  - 6 bit weights quantization, 50% sparsity



Algorithm: local Parameters Evaluation

- The t_0 is assigned by means of the mean-timer technique:
  - 19 equations based on the triplet pattern
  - Once the hits are filtered and L/R is assigned by the neural network, there is only one equation that fulfill the combination (per triplet)
  - Computed using full TDC-precision, i.e. the maximum precision available
- The local angle and intercept are finally evaluated by performing a least-square fit
  - After finding the t_0 it is possible to compute the hit positions
  - 3/4 hits points fit used to estimate the track parameters
- Trigger primitive inserted in the hit stream for offline evaluation



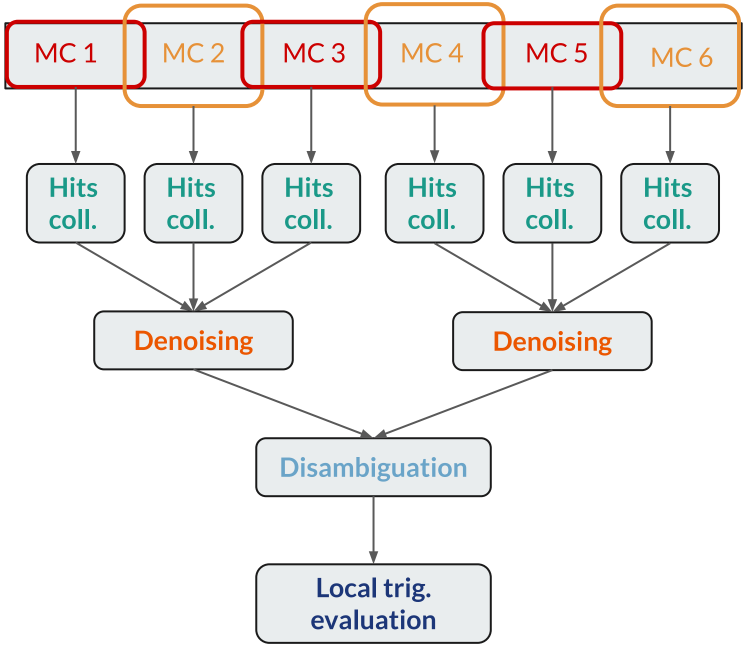
Algorithm: Resource Utilization

Resource utilization for a macrocell, i.e. group 4x4 channels:

- Denoising NN:
  - Latency: 2 clocks@40MHz
  - LUT: ~6K (~1% of available LUT in the KCU)
- Disambiguation NN:
  - Latency: 3 clocks@40MHz
  - LUT: ~5K (~1% of available LUT in the KCU)
- Trigger primitive evaluation
  - Latency 7 clocks@40MHz
  - LUT: ~6K (~1% of available LUT in the KCU)

Resources can be shared between macrocells:

- Based on the expected occupancy the ratio between different blocks can be tuned
- Optimal resource utilization



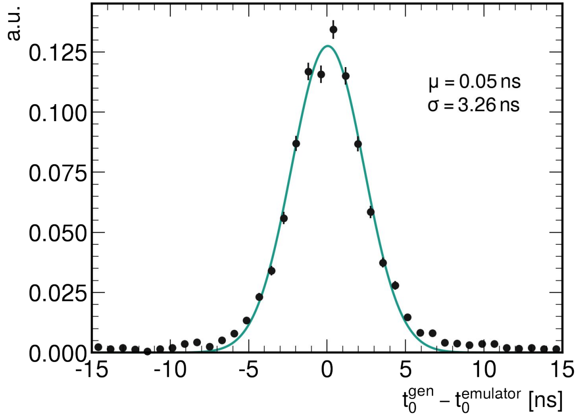
Performance: Simulation and Training

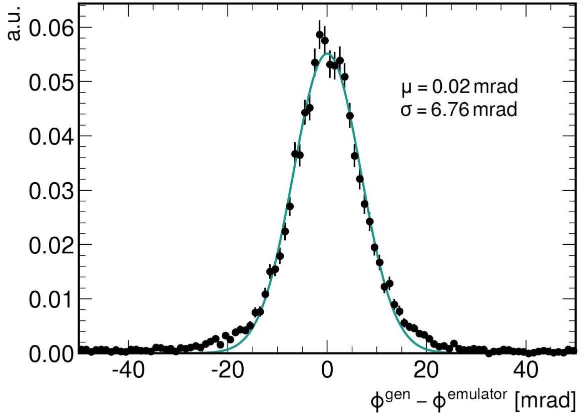
Dedicated simulation used as training dataset for the NNs:

- Flat muon spectra in bot local position and angle
- Hit smearing according to spatial resolution (250 μm)
- Cell inefficiencies/additional noise are simulated by removing/adding hits
- No full DT geometry / no material interaction sim. from GEANT / no E field homogeneity / …

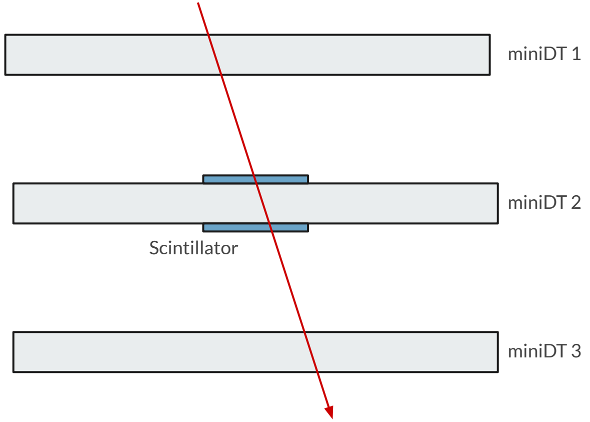
Pros and cons:

- Fast prototyping and full control over the simulation
- Not an official simulation, some effect poorly/not simulated



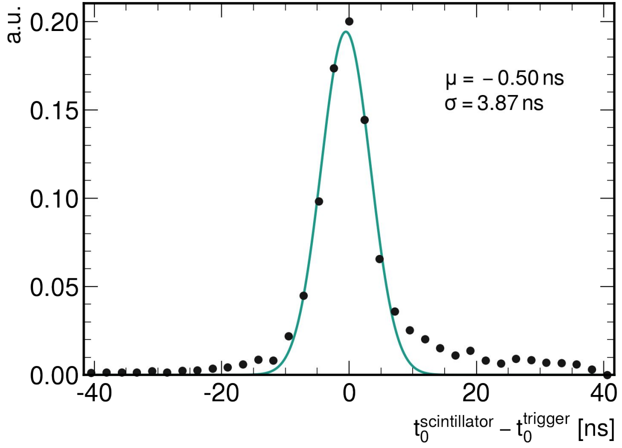


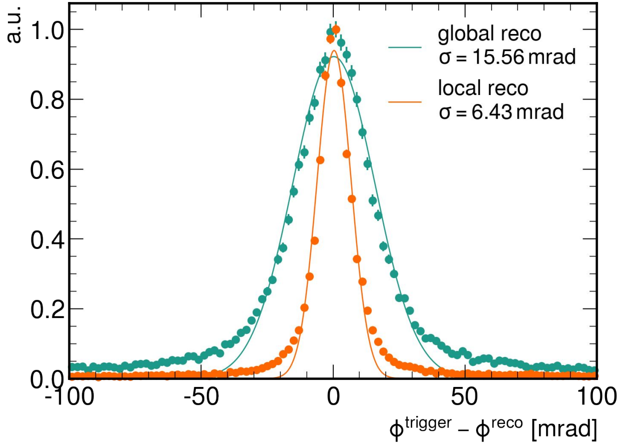
Performance: Cosmic Muons



Trigger performances compared to the global muon track:

- Global tracks are reconstructed out of all hits from the 2 external miniDTs
- Hits positioned using an external estimate of t_0 provided by two scintillator palettes
- Local track found using only hits from the miniDT where the trigger is running





Conclusions and Remarks

A NN-based local trigger demonstrator has been implemented and tested:

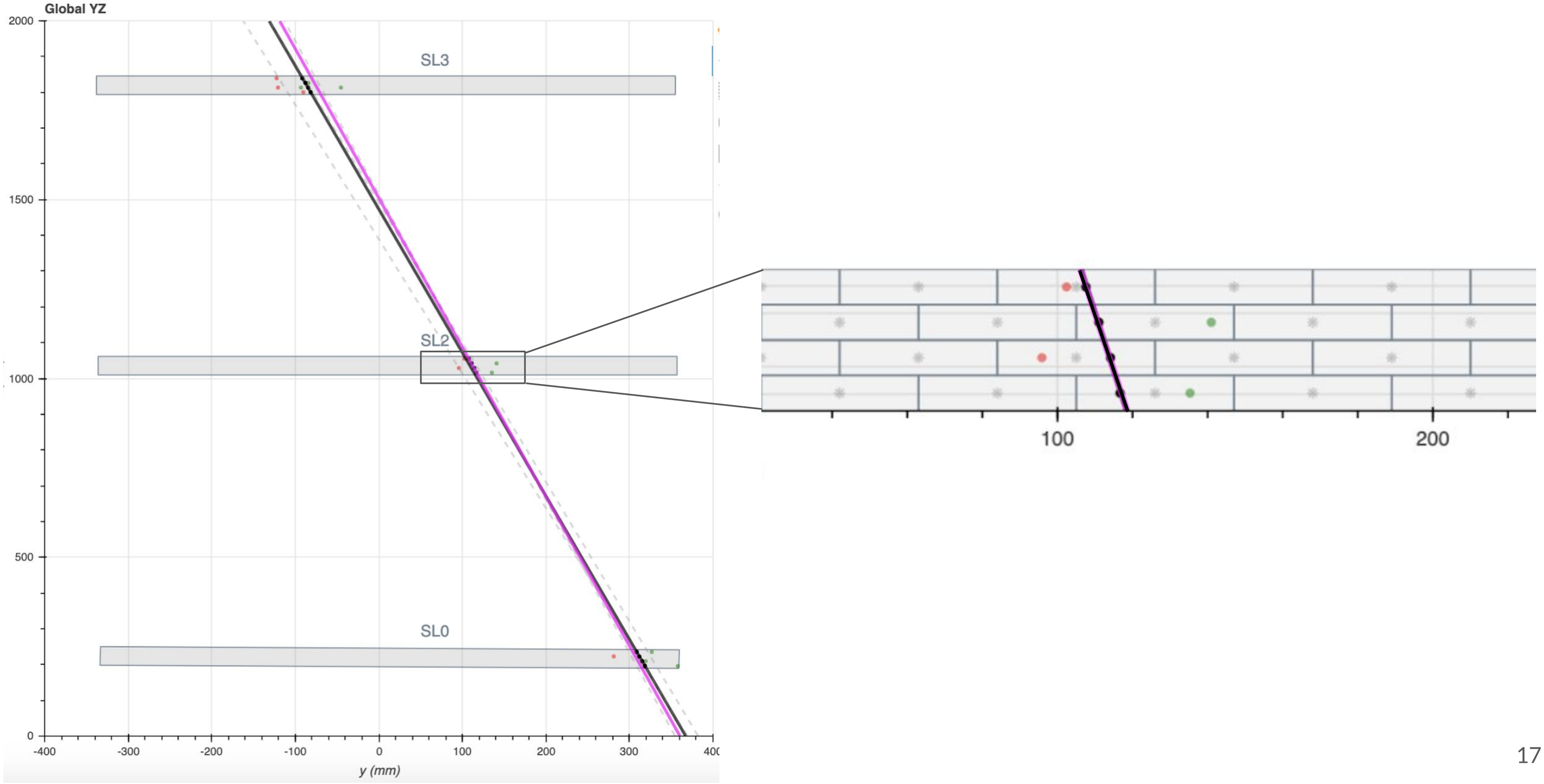
- NNs used to remove combinatorial before the application of analytical approaches
- NNs can be retrained to cope with different conditions
  - Can be easily replace without rewriting firmware
  - Increase flexibility of the method
  - Different models for different conditions/geometry
- Small area occupied on the board and contained latency
  - Optimal resource can be achieved with a “tree”-structure

Multiple improvements are possible:

- Model performance depends on the quality of the simulation
  - Retrain models on a more realistic simulation

Backup slides

Performance: Event Display



KCU1500 FirmWare