dRICH Online Data Filter

Cristian Rossi

ePIC DAQ meeting
08/04/2026

Data reduction with ML Classifier

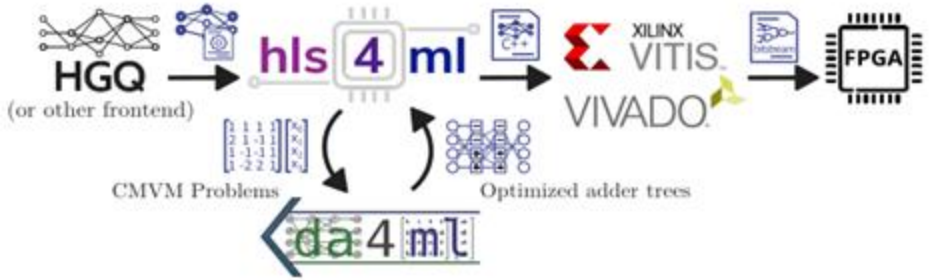
Online **Signal/Noise discrimination** using ML

- **Signal**
  (i.e Merged Phys Signal+ Bkg)
  - **Physics Signal:**
    - ➔ e.g. DIS, SIDIS, ...
  - **Physics Background:**
    - ➔ e/p interaction with beam pipe
    - ➔ synchrotron radiation
- **SiPM Noise:**
  - Dark Count Rate (DCR) modeled on the reconstructed dataset

Discriminate between **Noise-Only** and **Signal+Noise** events

Workflow: from SW model to HW firmware

- For the **quantization step,** we aligned our system with the new state-of-the-art libraries for FPGA ML model implementation:



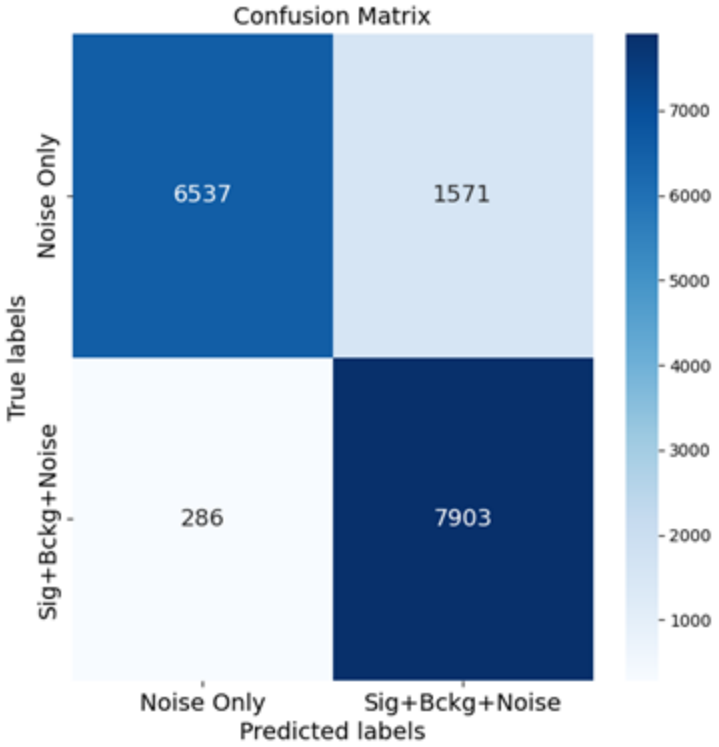
❑ **HGQ**2 (**H**igh **G**ranularity **Q**uantization 2) is a **quantization-aware training** framework built on Keras v3, targeting real-time deep learning applications on edge devices like FPGAs.

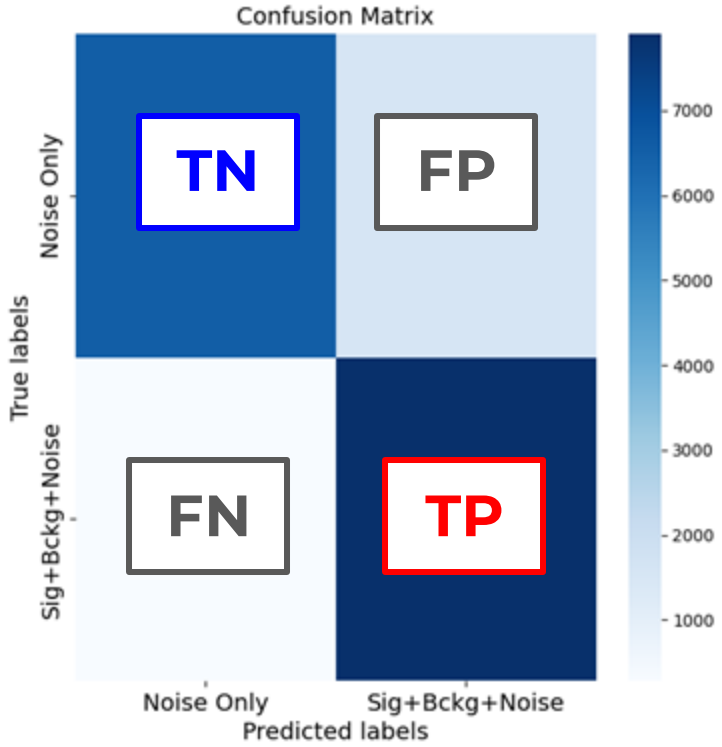
➢ HGQ2 implements an **gradient-based automatic bitwidth optimization** and quantization-aware training algorithm
⇒ **bitwidth optimization** at arbitrary granularity, up to per-weight and per-activation level.

❑ **da4ml** is a library implementing neural networks on FPGAs with DA (**D**istributed **A**rithmetic):

➢ **Constant Matrix Vector Multiplication** (CMVM) optimizer with graph-based pre-optimization and common sub-expressions elimination (CSE) to **reduce the firmware footprint for the CMVM operation**.

Positive (P) ⇒ Signal // Negative(N) ⇒ Noise



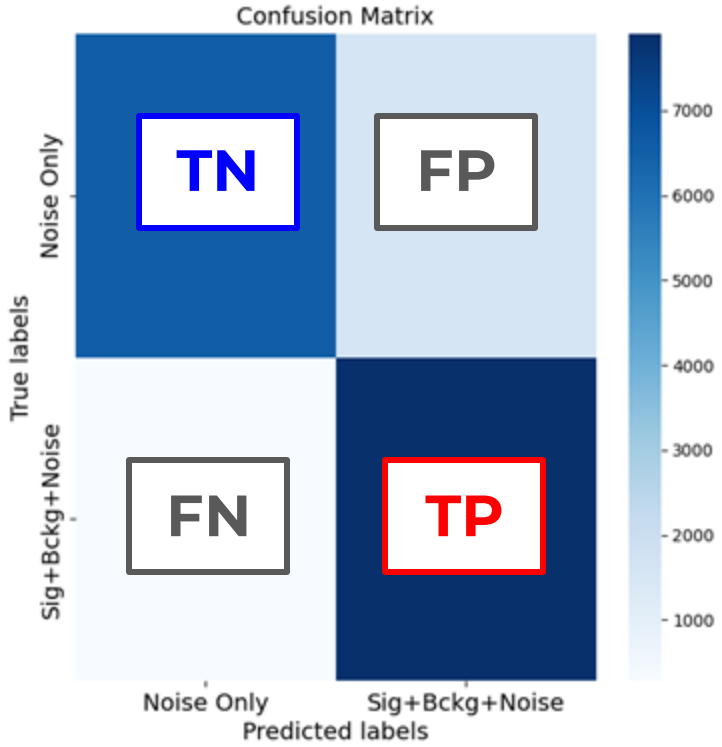


Performance evaluation and targets

- In this change of paradigm, to evaluate the model performance, we introduce:
  - **True Positive Rate (TPR):** percentage of correctly classified true-labeled Signal+Background+Noise events
  - **True Negative Rate(TNR):** percentage of correctly classified true-labeled Noise-Only events

$$TPR = \frac{TP}{TP + FN}$$ as high as possible

$$TNR = \frac{TN}{TN + FP}$$ >= 80% [⇒ reduction factor >=5]



Outline of the presentation

Refinement of Dataset

- 0-hits signal+bkg removal
- Use of data from EIC simulation/reconstruction campaigns

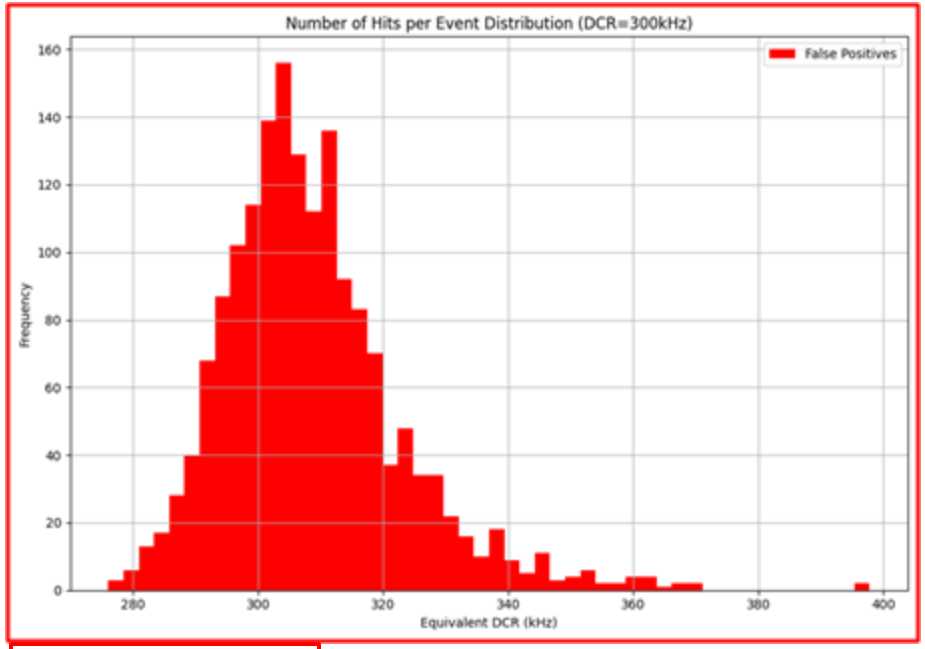
<u>Algorithmic improvements</u>

- **Optimization of the binary classifier**
- **Usage of timing information**

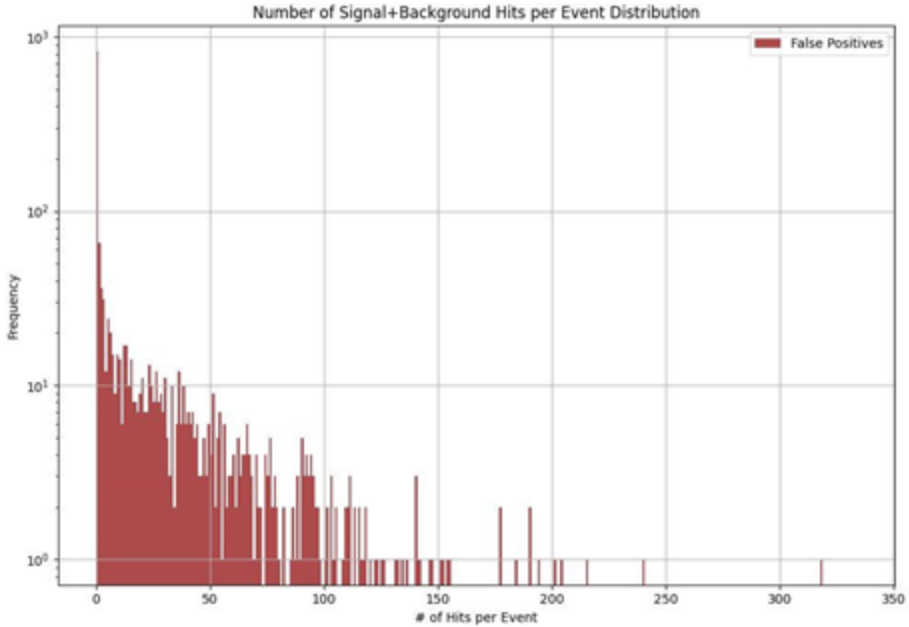
Prototyping of multi-FPGA implementation

Dataset generation ⇒ 0-Hits removal

False Negative events ⟹ (same indexes) Source Sig+Bckg reconstructed events



RICH2025 performance



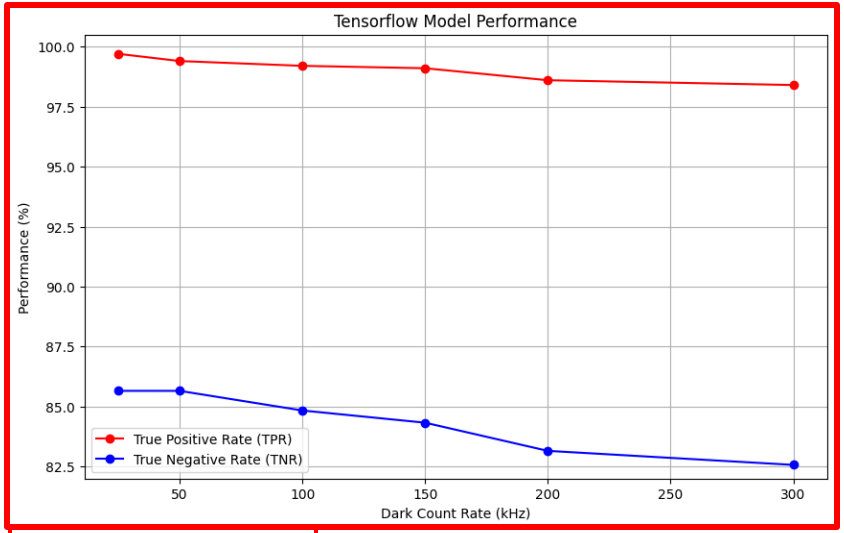
FN = signal events mis-predicted as noise by the model ➔ cause!

False Negative (FN) ⇒ 0-hits removal (⇒ “nozeros” dataset)

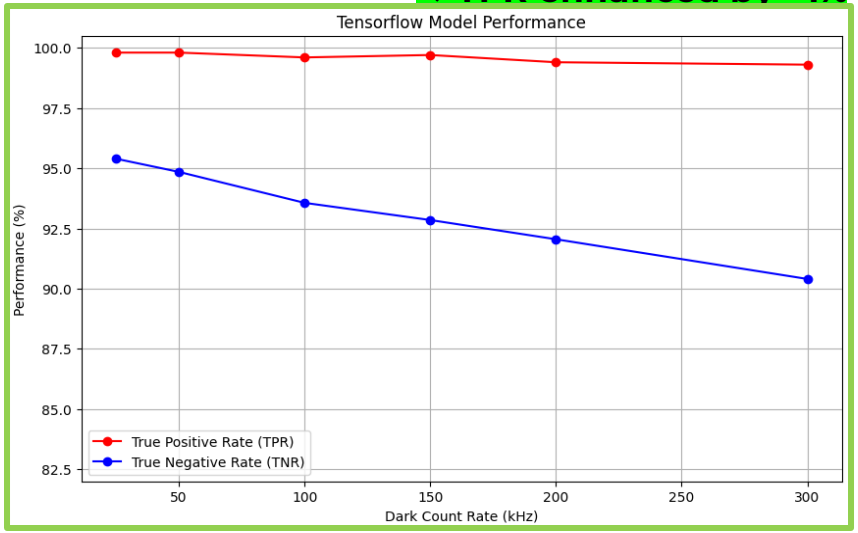
0-hits removal

⇒ TNR enhanced by ~8%

⇒ TPR enhanced by ~1%



RICH2025 performance



FN = signal events mis-predicted as noise by the model ➔ possible solution

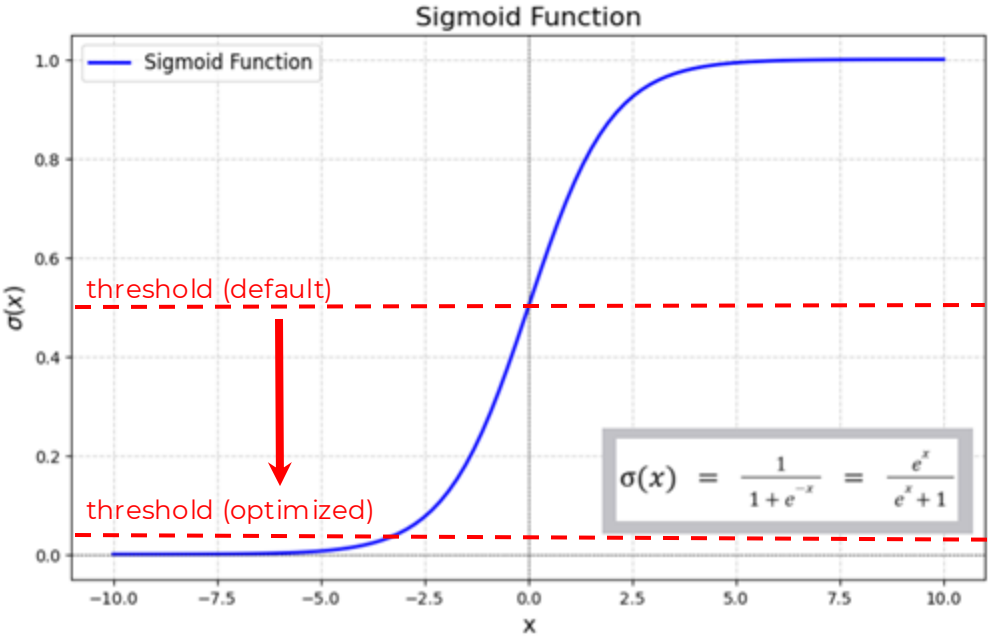
Optimized threshold for TPR maximizing

Binary Classifier Output (Sigmoid Activation)

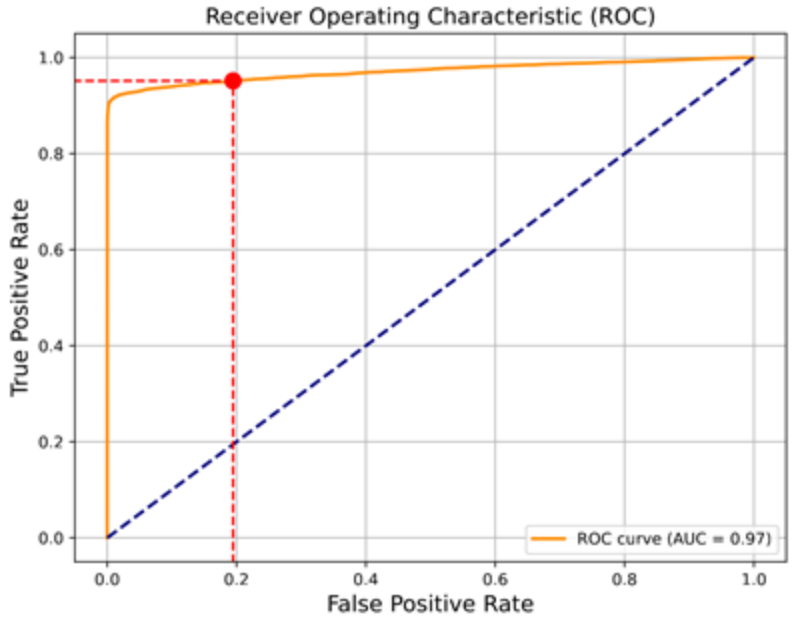
- The final layer uses a **sigmoid activation function**, producing an output in the range **[0, 1]**, interpreted as the **probability of belonging to the positive class**.
- A **decision threshold t** is applied to obtain a binary prediction:
  - *output* ≥ *t* ⇒ **positive class**
  - *output* < *t* ⇒ **negative class**
- The default choice is **t=0.5**, but the threshold can be **adjusted depending on the analysis goals**.

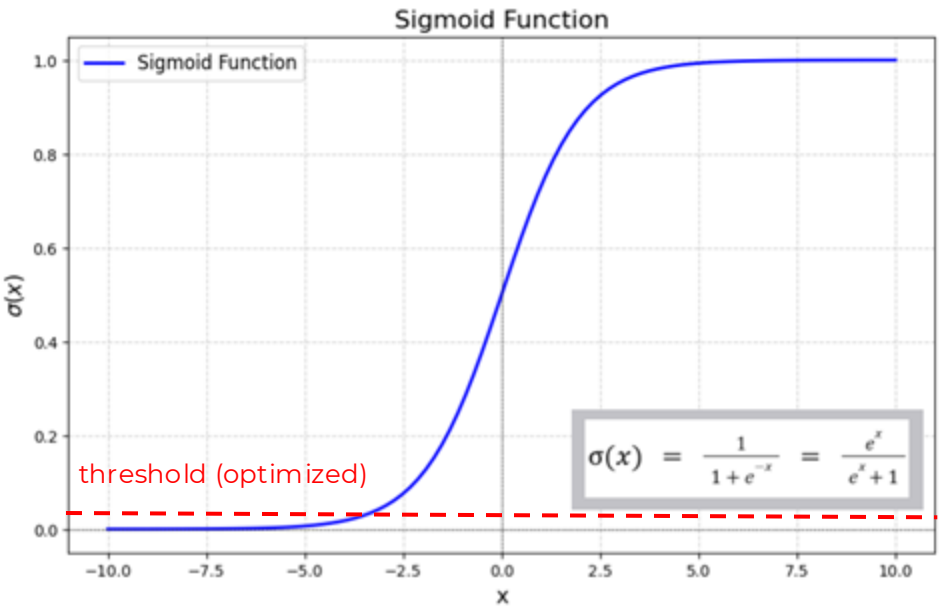
Effect of changing the threshold

- **Lower threshold** → more events classified as positive ⇒ higher **sensitivity/recall for positives**, but more **false positives** ⇒ higher **TPR**, lower **TNR**
- **Higher threshold** → stricter positive classification ⇒ higher **specificity for negatives**, but more **false negatives**.
  ⇒ higher **TNR**, lower **TPR**



ROC studies for TPR maximizing



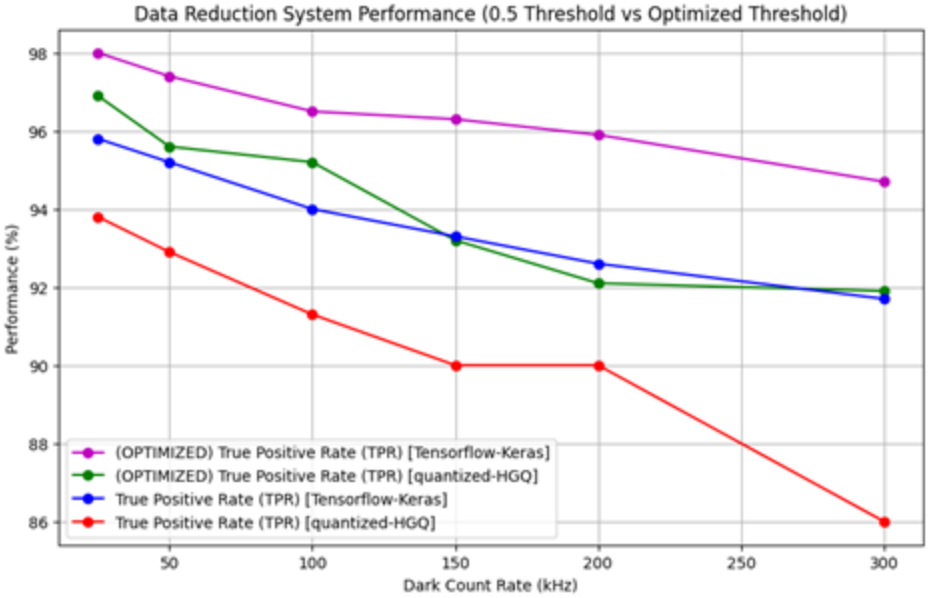


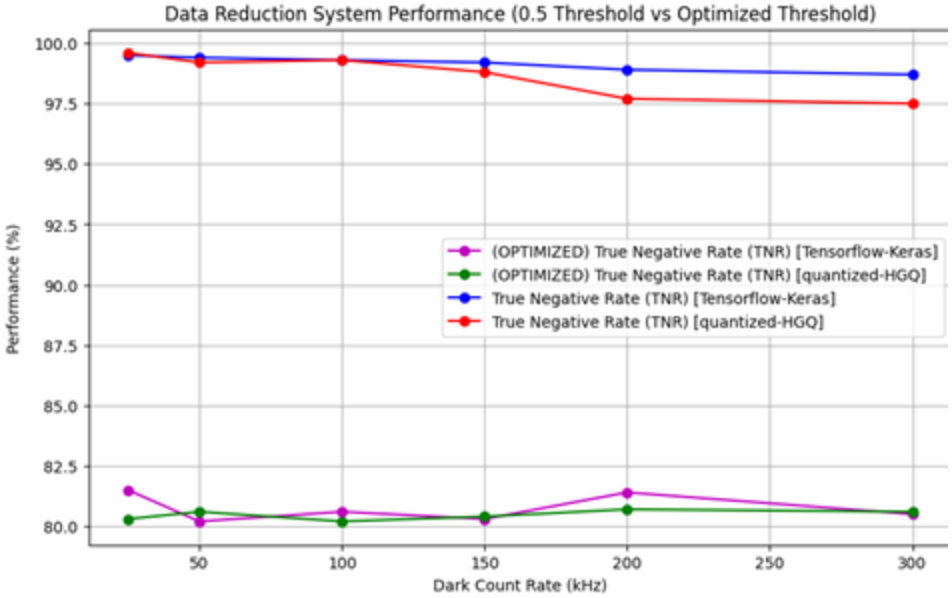
$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \quad >= 80\% \; [\Rightarrow \text{reduction factor} >= 5]$$

$$\text{FPR} = 1 - \text{TNR}$$

TPR maximizing: performance

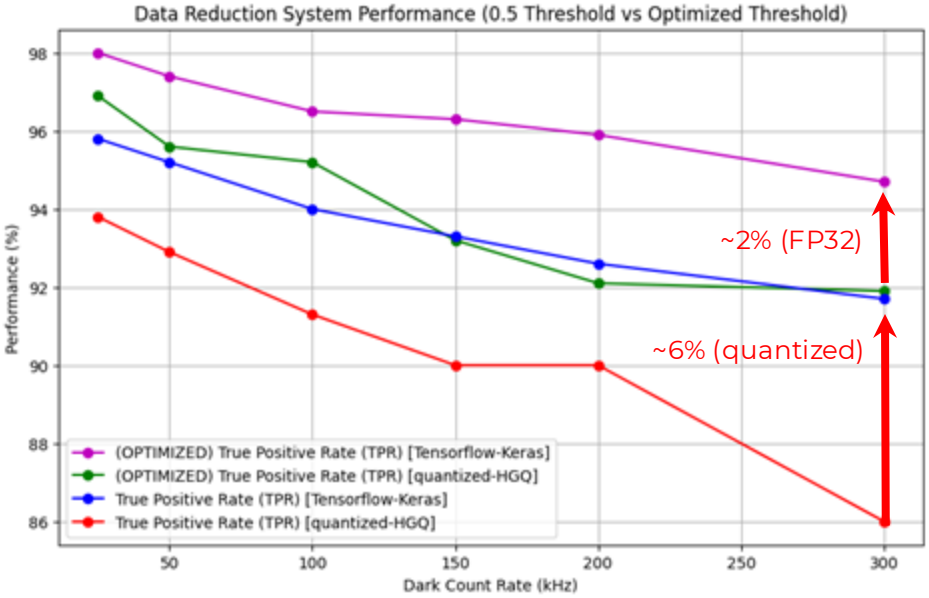
P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)

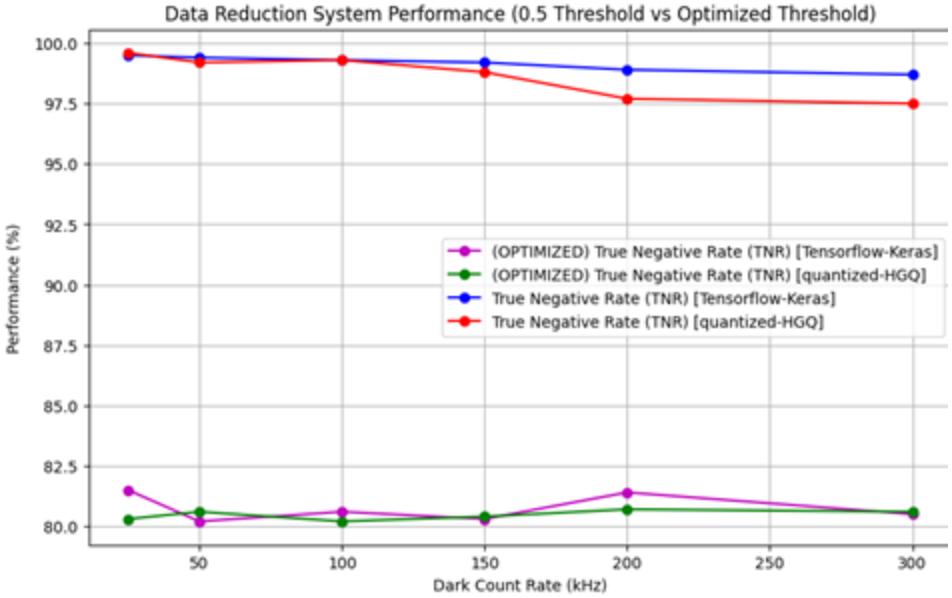




TPR maximizing: performance

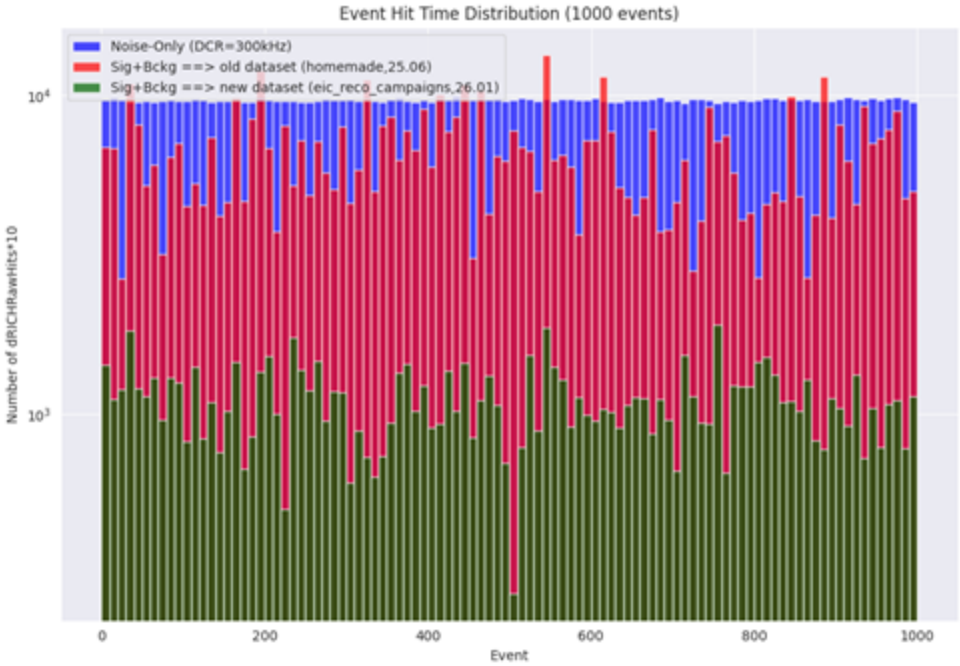
P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)





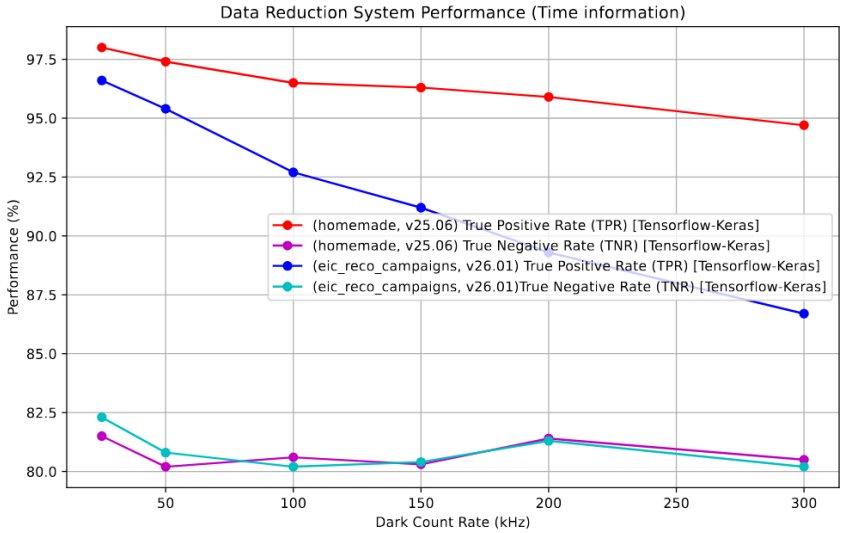
Dataset ⇒ from “homemade” to “eic_campaing”

- Dataset “homemade” generated using **EICrecon v25.06** ⇒ p**otential bias**
- To asses system performance and minimize potential dataset bias:
  ⇒ model was trained and validated on data from **EIC simulation/reconstruction campaigns**
- This ensures consistency with more recent software versions (e.g. **v26.01**)



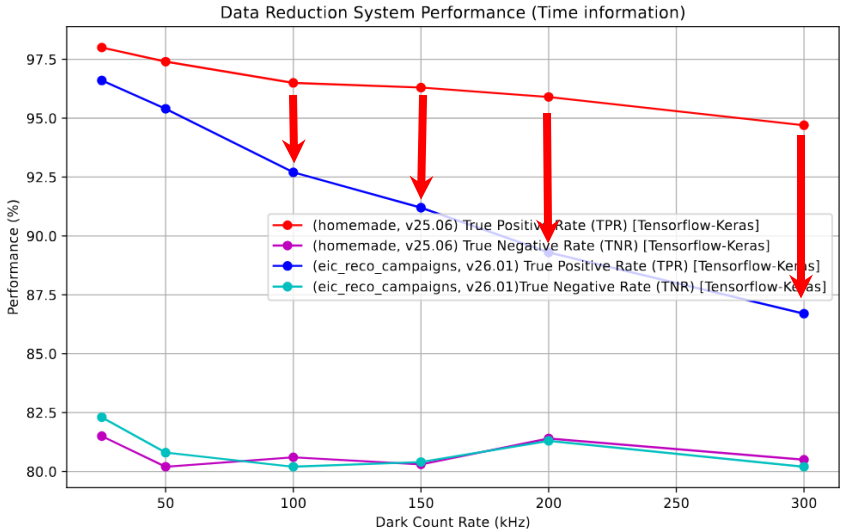
Performance comparisons

P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)

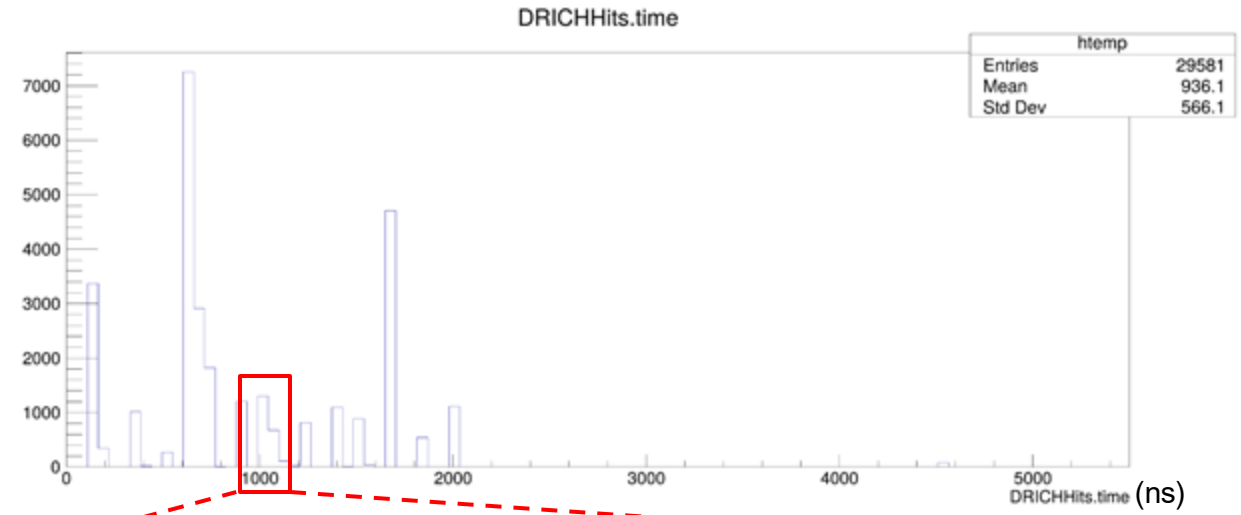


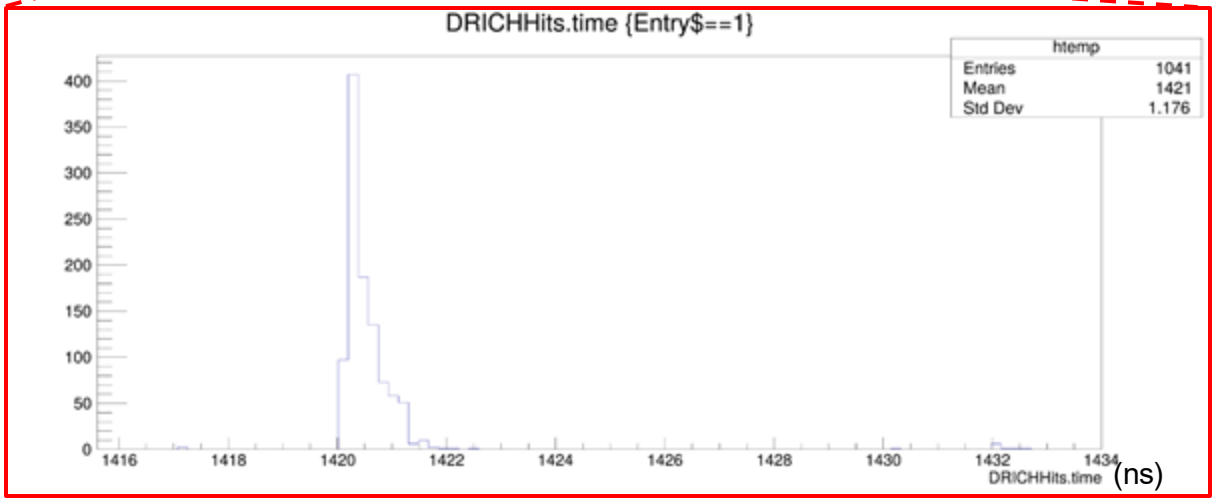
Performance comparisons

P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)



Time-Information Input for Multi MLP model





Signal+Background Simulated Events

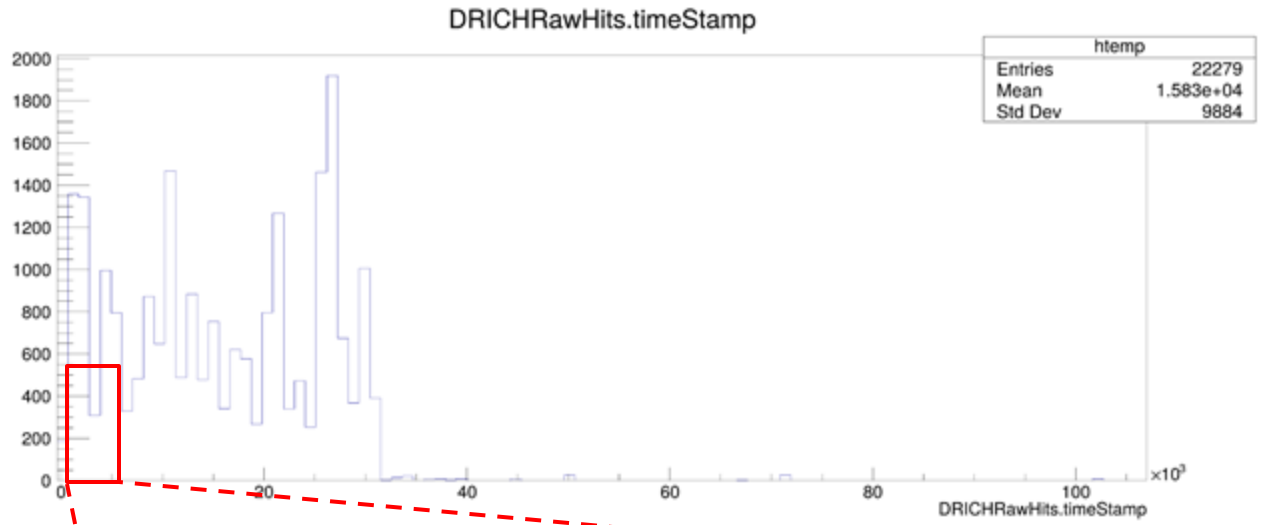
(FULL/26.01/Bkg_1SignalPer2usframe)

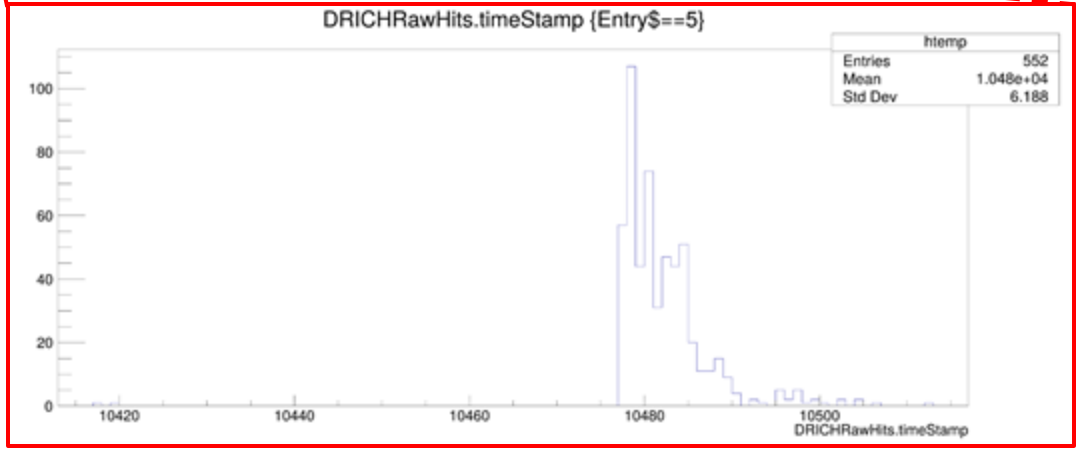
⇒ **ROOT File TBranch**: DRICHHits

⇒ **Entry**: DRICHHits.time

Focus on a single signal event time distribution

⇒ ~2ns wide time distribution (as expected)





Signal+Background Reconstructed Events

(RECO/26.01/Bkg_1SignalPer2usframe)

⇒ **ROOT File TBranch**: DRICHRawHits

⇒ **Entry**: DRICHRawHits.timeStamp

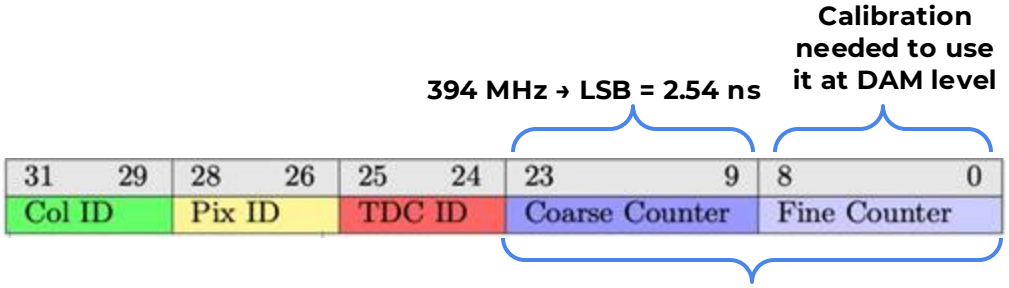
time(ns) = timeStamp / 16

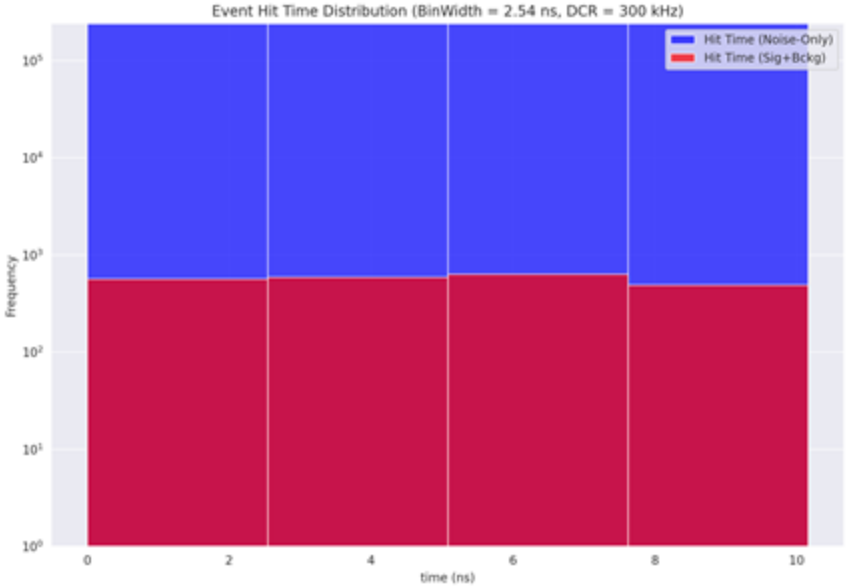
Focus on a single signal event time distribution

⇒ ~2ns wide time distribution (as expected)

Timing Distribution

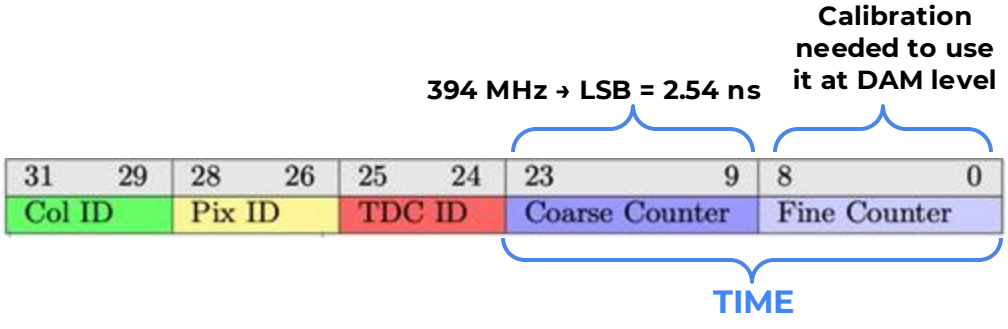
- We start evaluating how to implement the **timing information** to enhance the performance of the data reduction system.
- Thus, we started to look at the **timing distribution** of the datasets generated for the MLP NN model training and validation

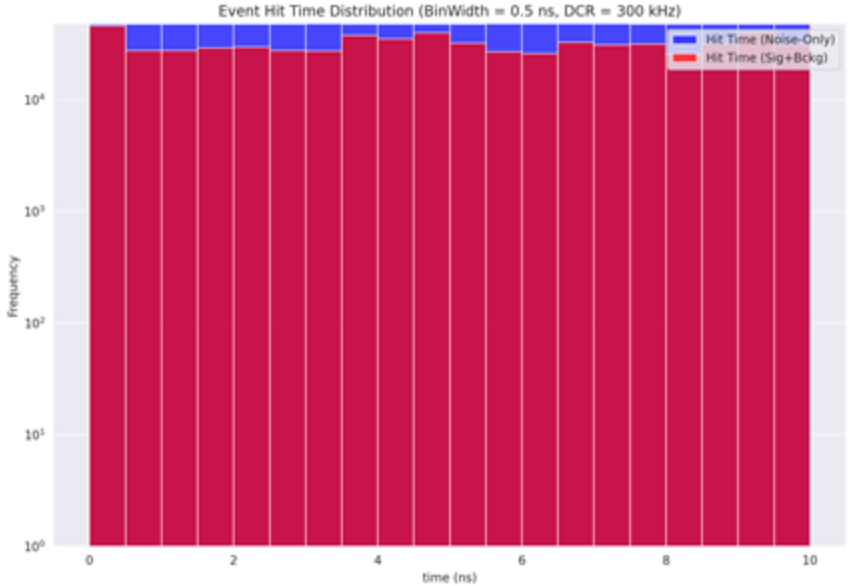




Timing Distribution

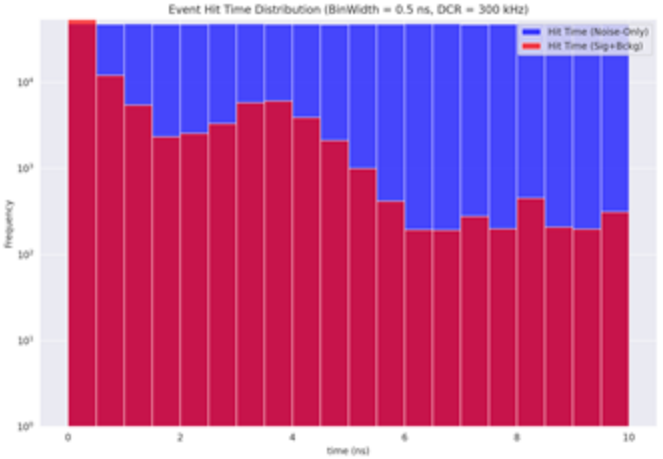
- We start to evaluate how to implement the **timing information** to enhance the performance of the data reduction system.
- Thus, we started to look at the **timing distribution** of the datasets generated for the MLP NN model training and validation
  ⇒ **binwidth** connected to the bit resolution available from **Coarse Counter** and **Fine Counter**

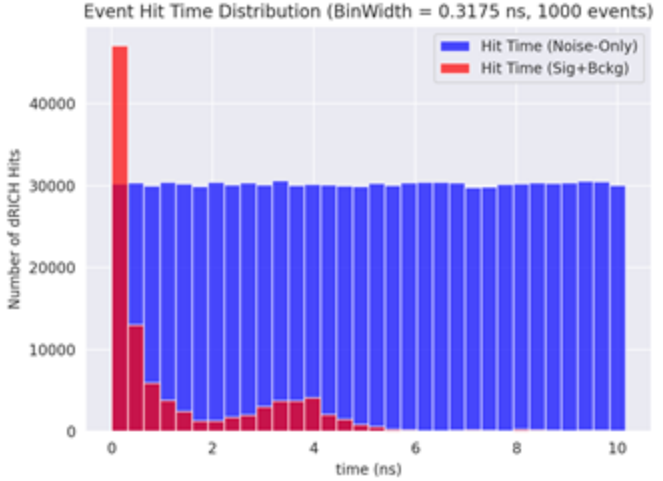




Time normalization

- Studied temporal distribution of hits within events
- **Time normalized to first hit in each event**
- Distribution shows:
  ⇒ **~2 ns wide peak** (⇒ primary interactions)
  ⇒ **tail + second peak** (~4 ns) (⇒ secondary interactions)



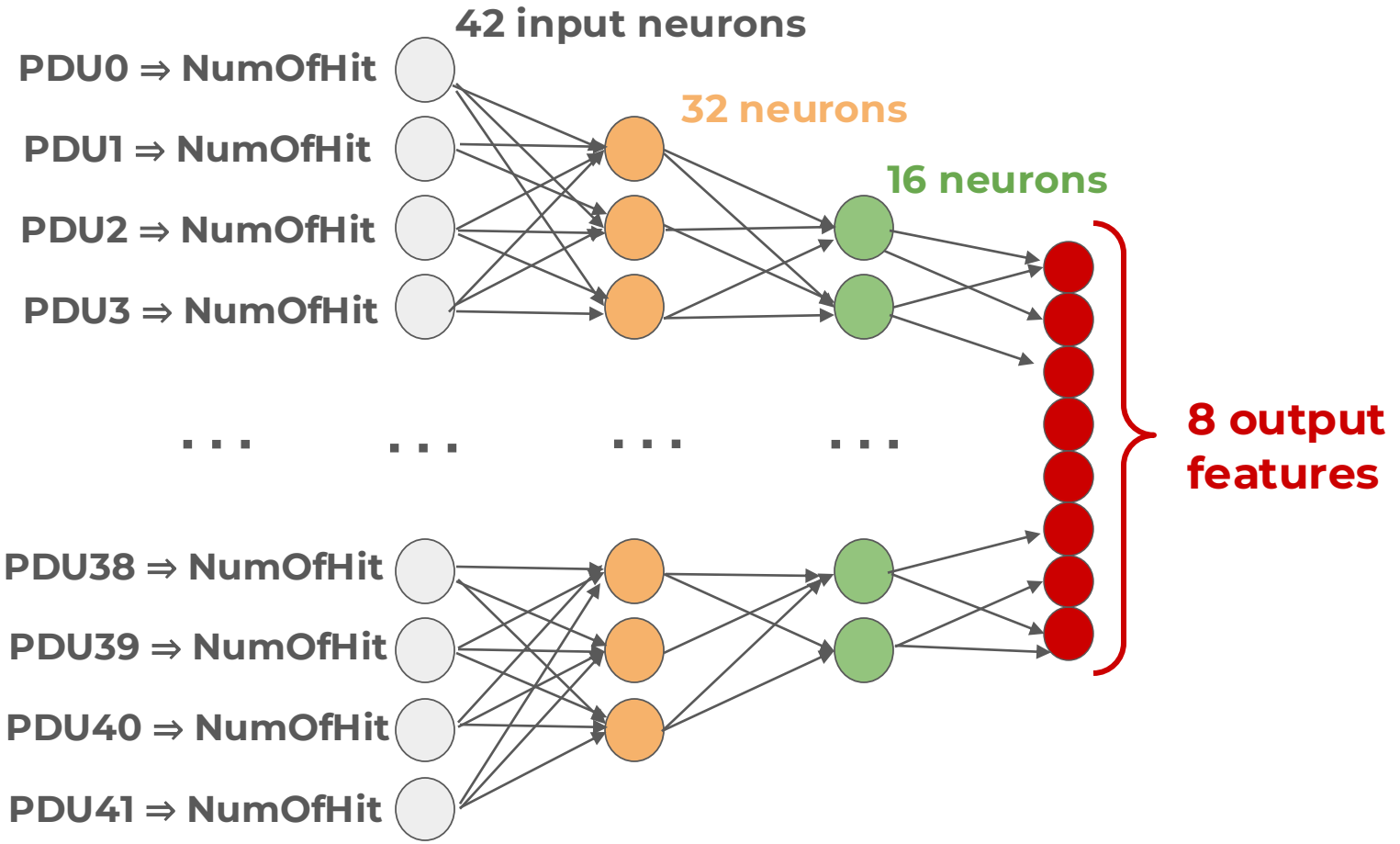


(Open Question)

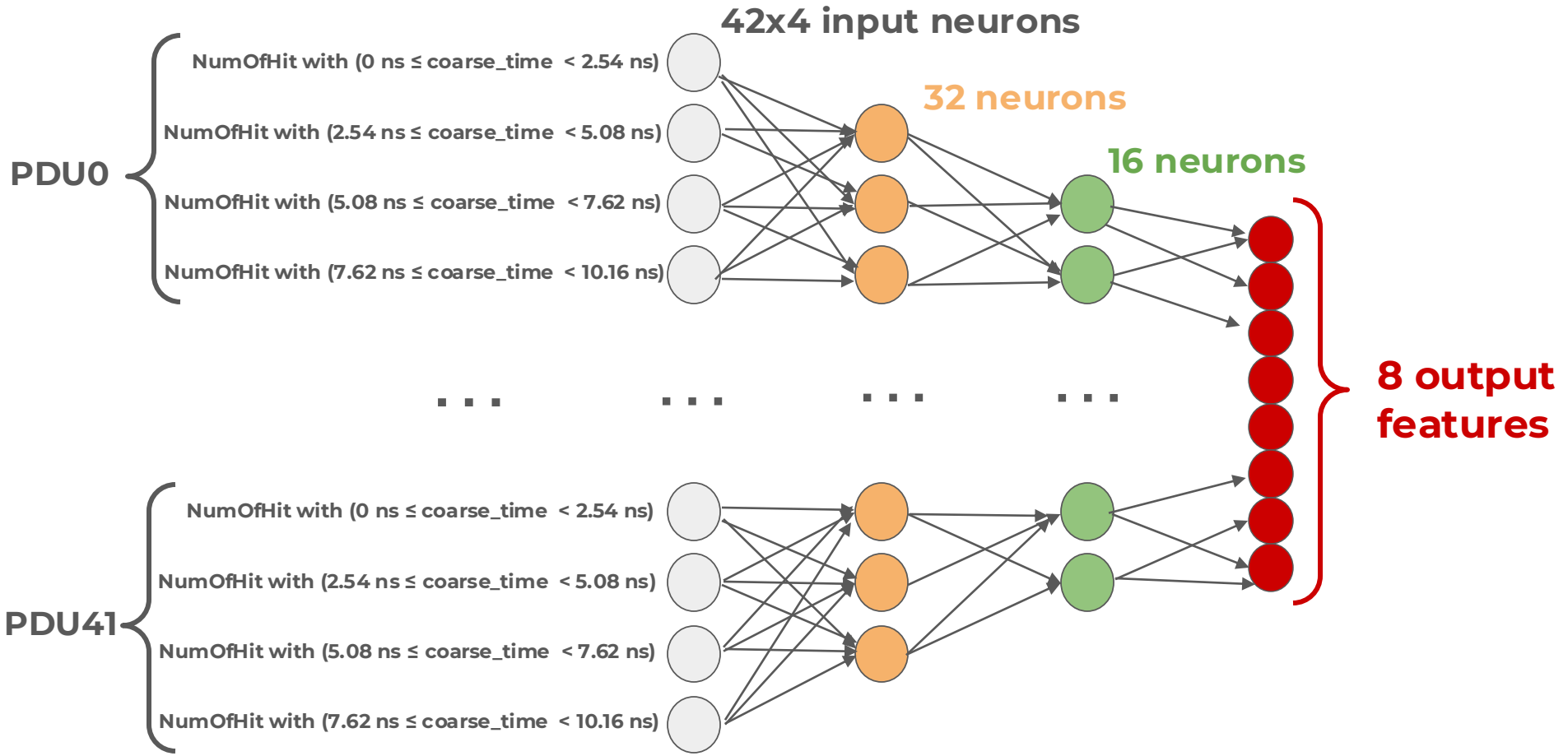
➔ Is this normalization method valid for dRICH readout?

Multi-MLP model with timing information
⇒ Keras/HGQ2 evaluation

DAM input ⇒ Subsector PDU Number Of Hit

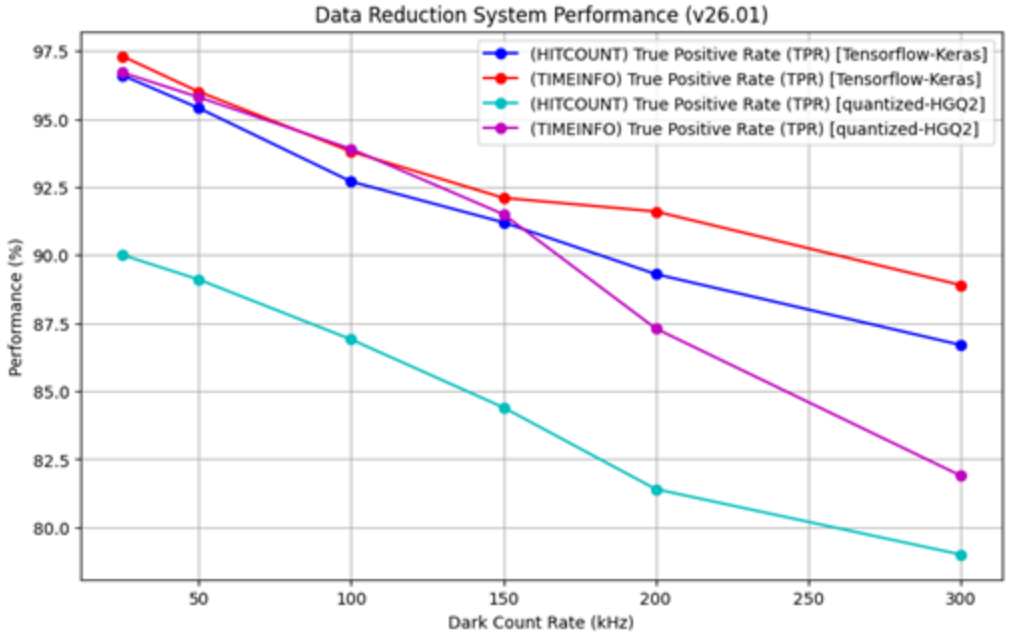


4 bin time-histogram (2.54ns binwidth) ⇒ DAM input



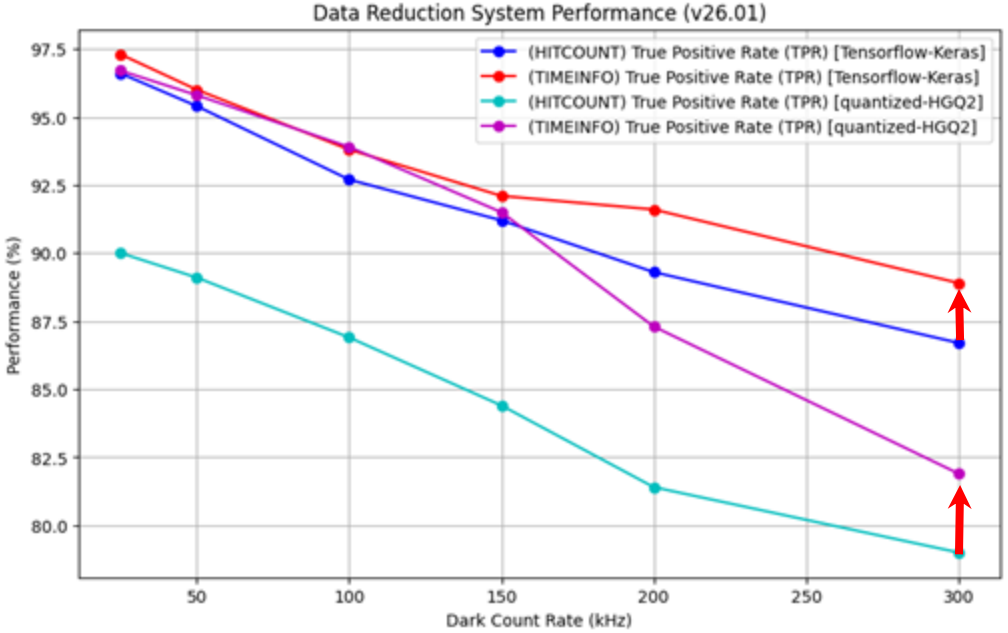
TPR comparison (time normalized)

P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)



TPR comparison (time normalized)

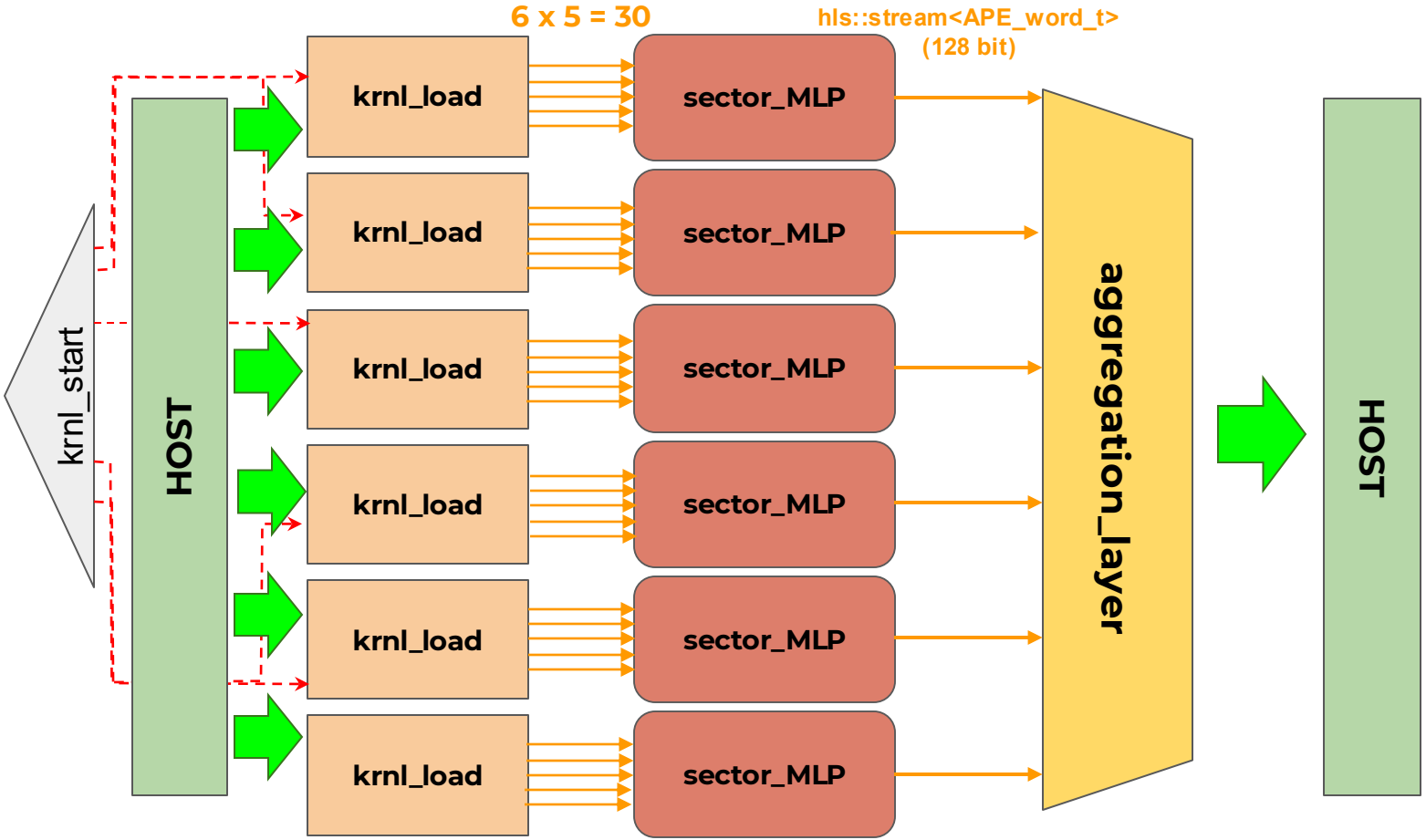
P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)



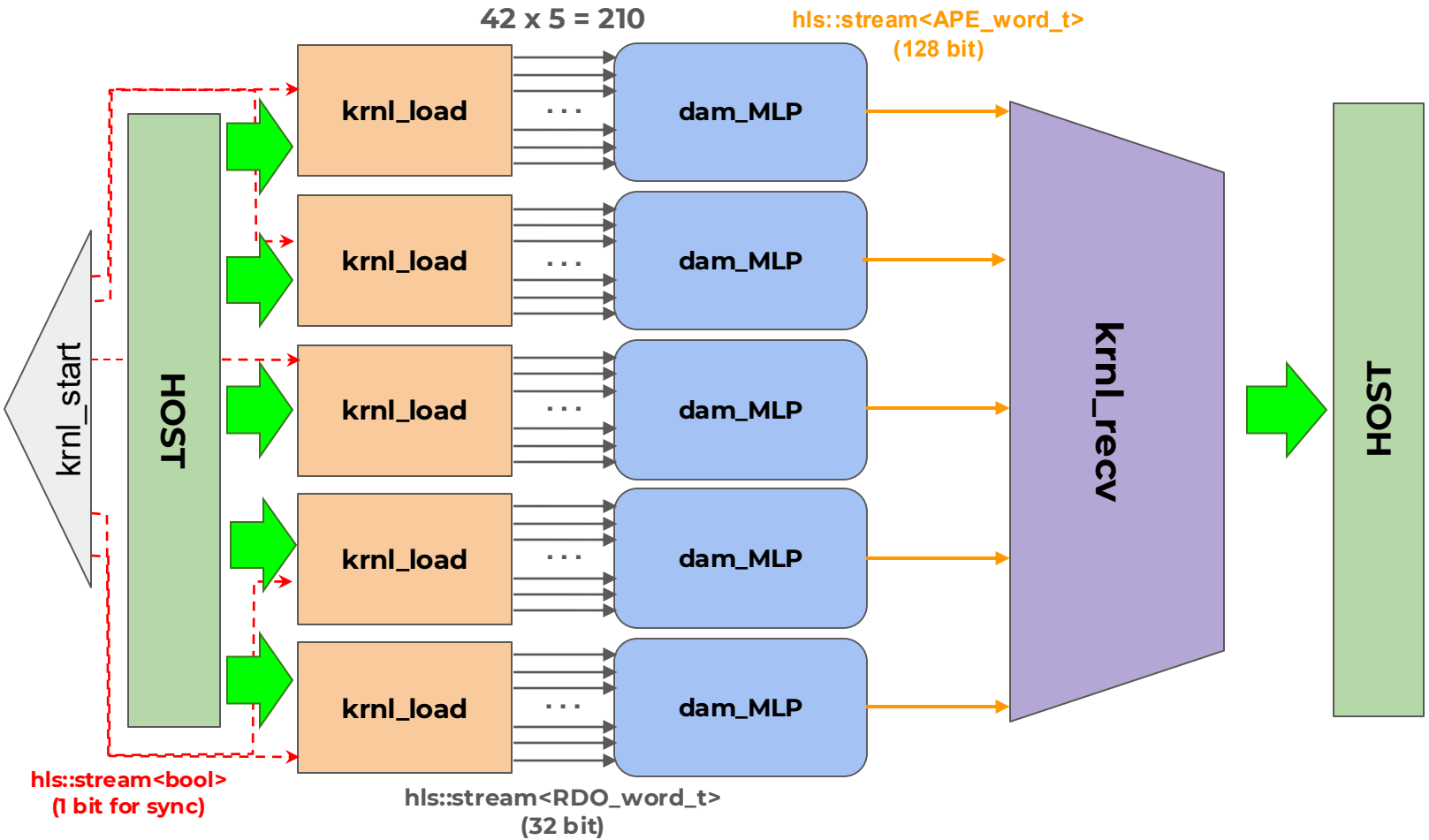
HW Multi-MLP model implementation

⇒ DAM to TP Communication testbed (preparation for RT2026 and possibile IEEE publication)

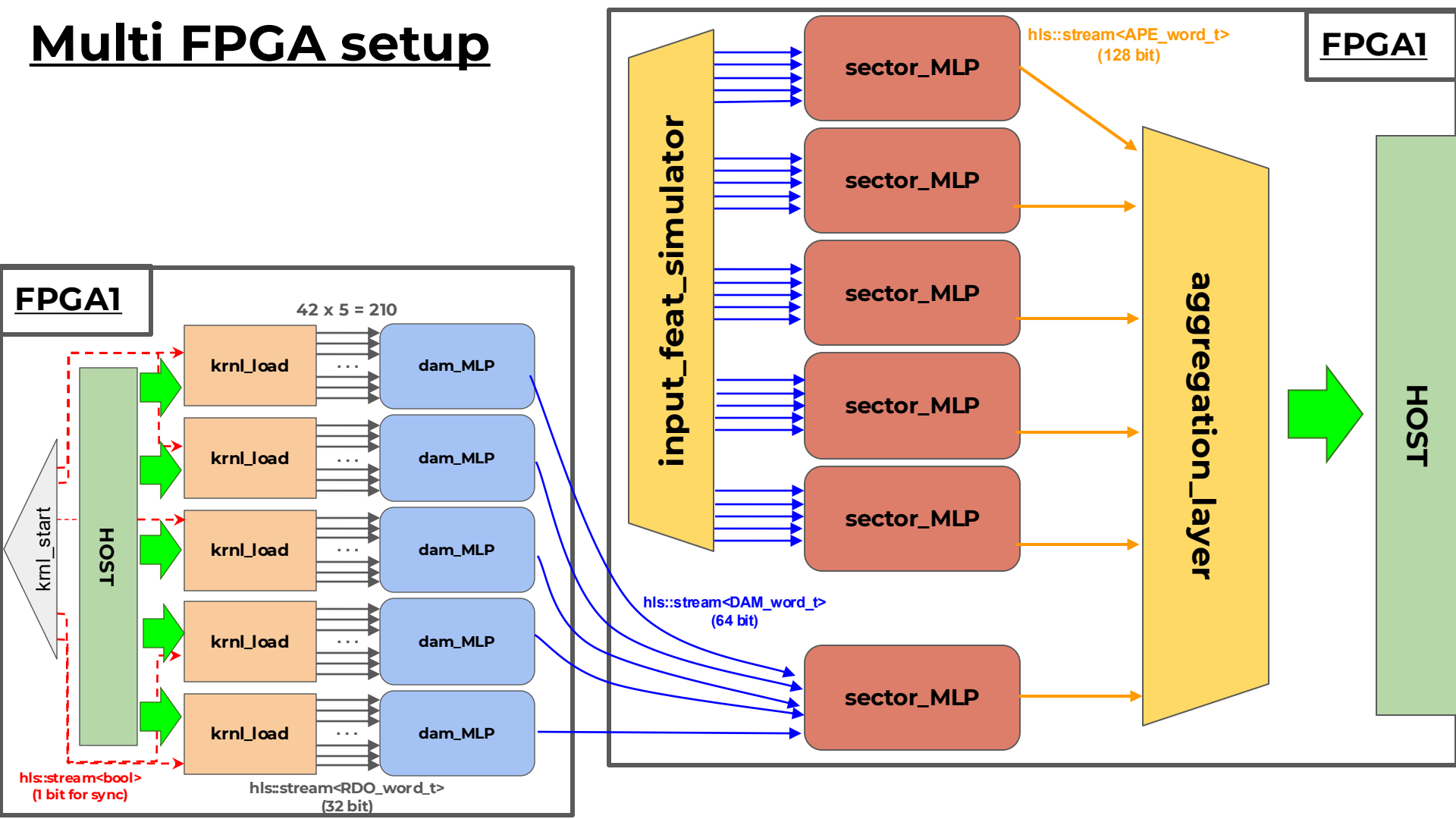
Single FPGA setup ⇒ TP architecture



Single FPGA setup ⇒ Sector architecture (5 DAMs)



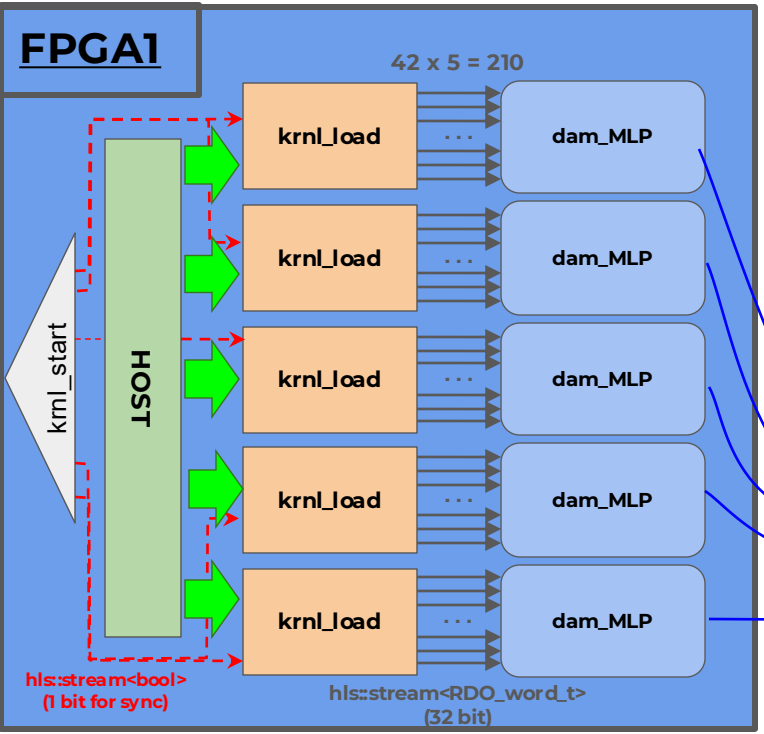
Multi FPGA setup

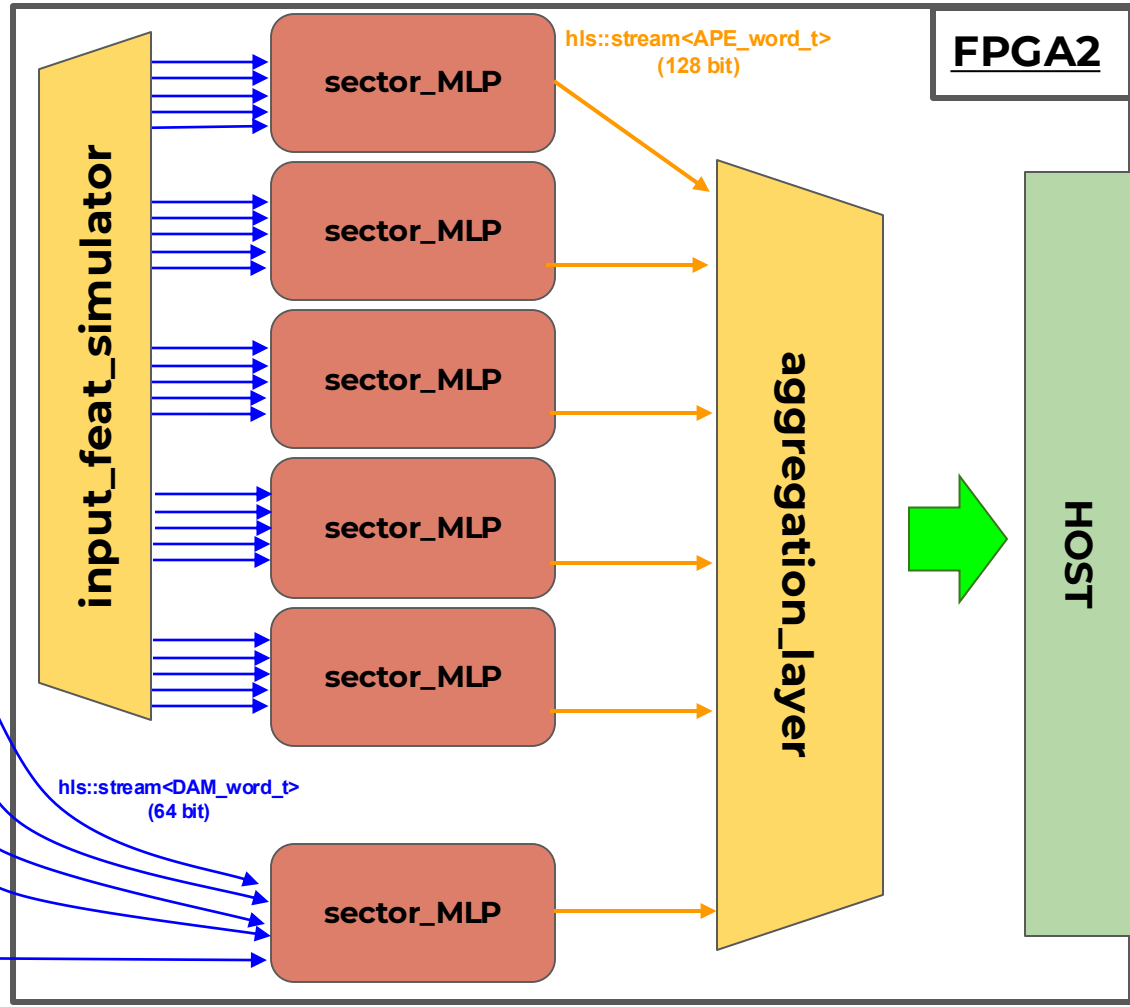


Multi FPGA setup

FPGA1

⇒ datastream simulation of an entire sector (5 DAMs)
⇒ ALINX AXAU15

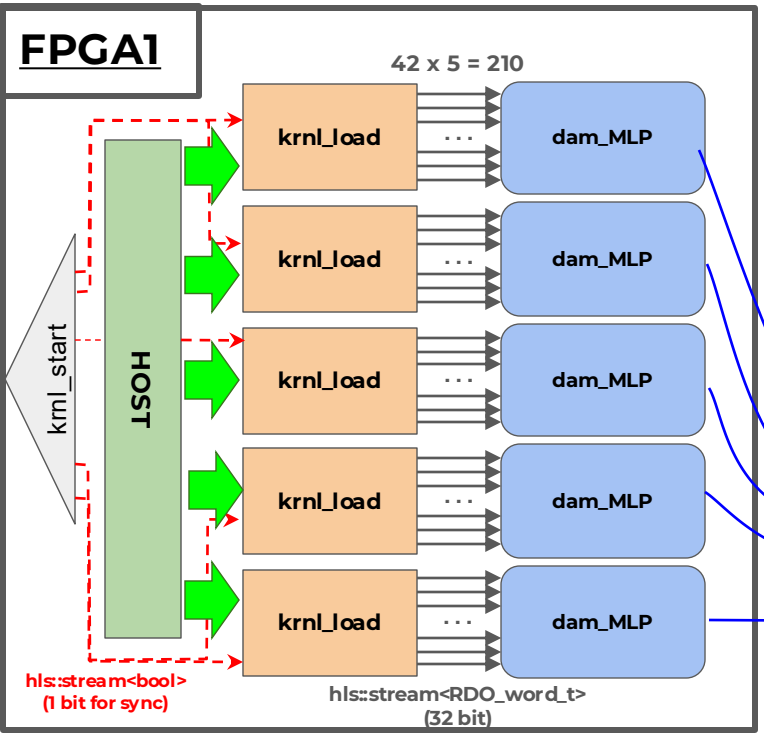


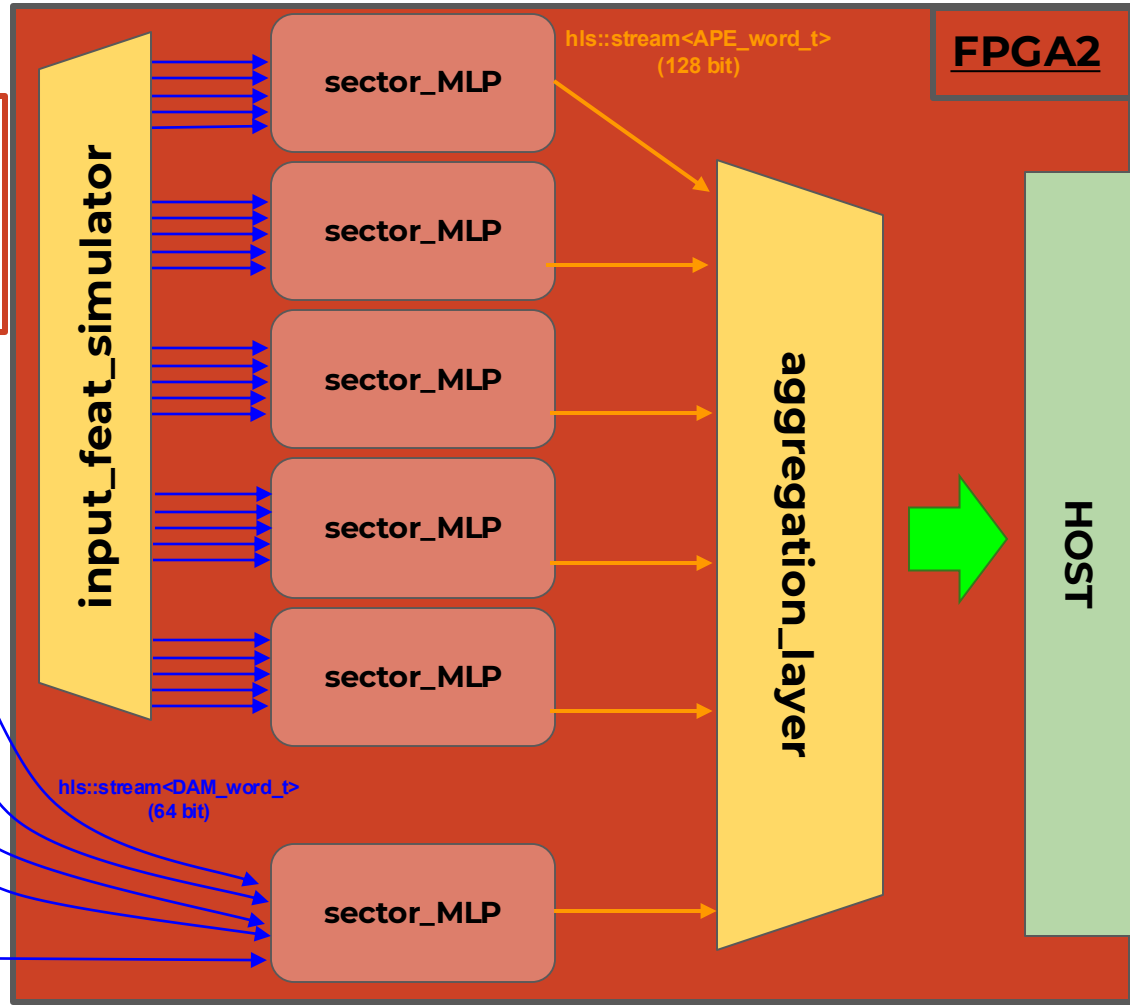


Multi FPGA setup

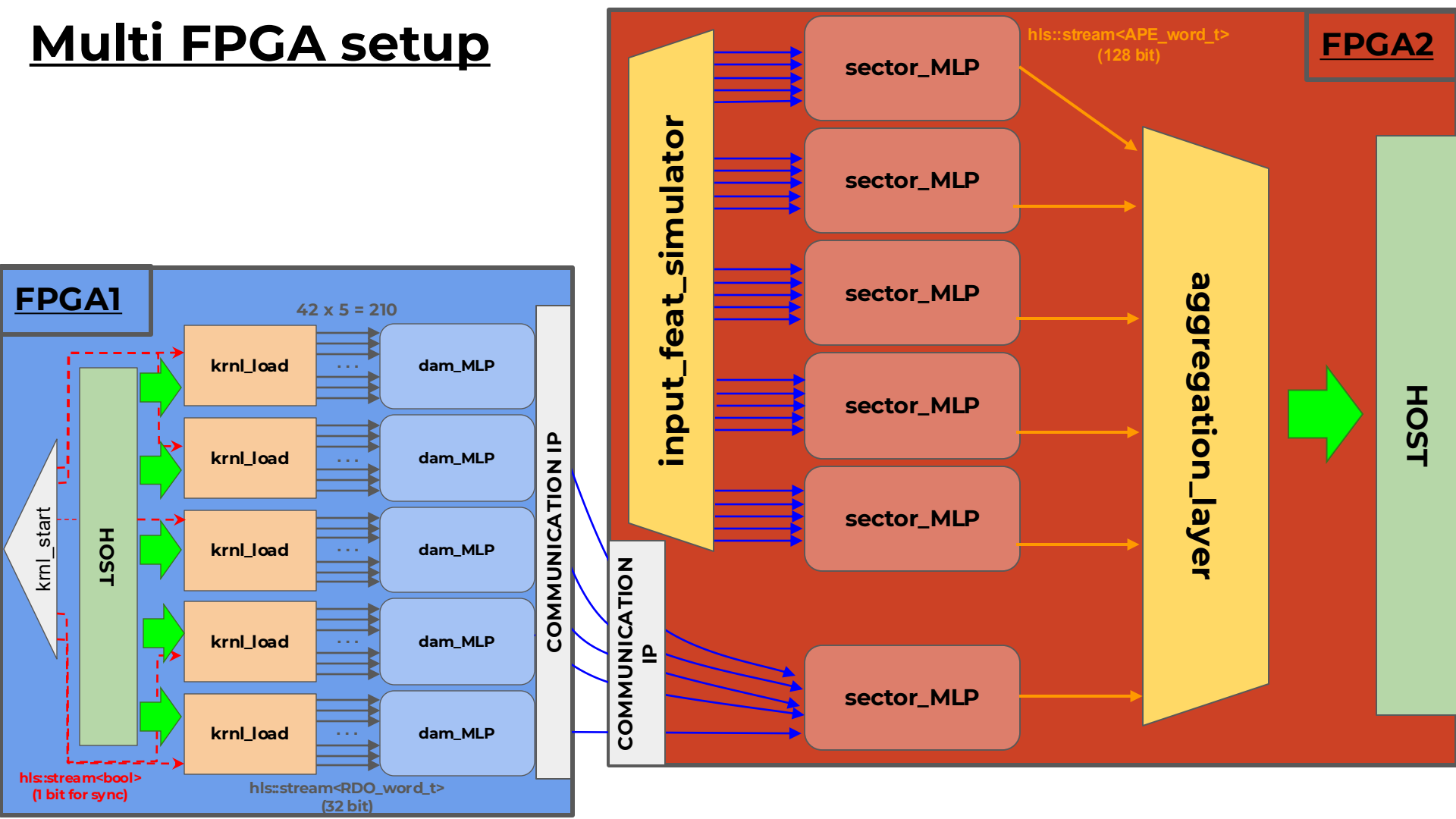
FPGA2

⇒ 5 input links coming aggregated as input of a single Sector_MLP (simulated datastream for the remaining 5 ones)
⇒ FLX-182



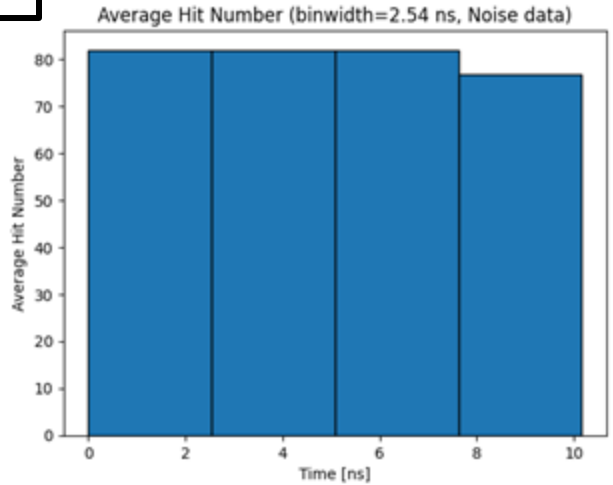


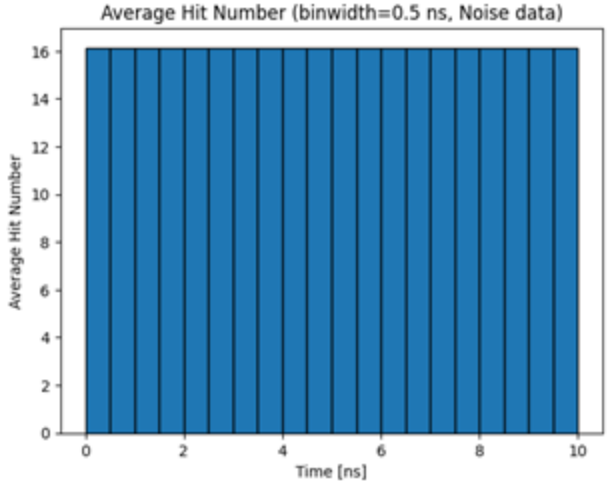
Multi FPGA setup

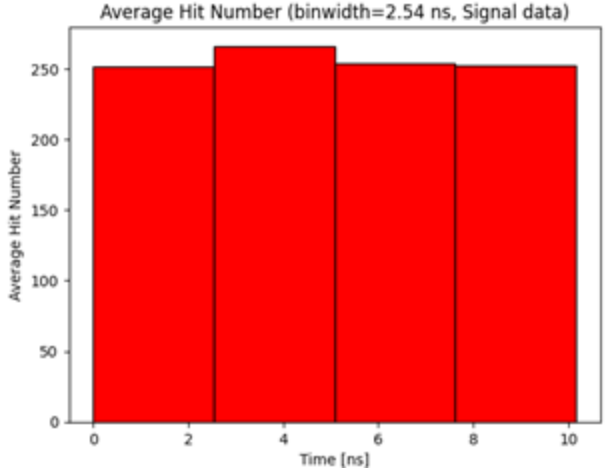


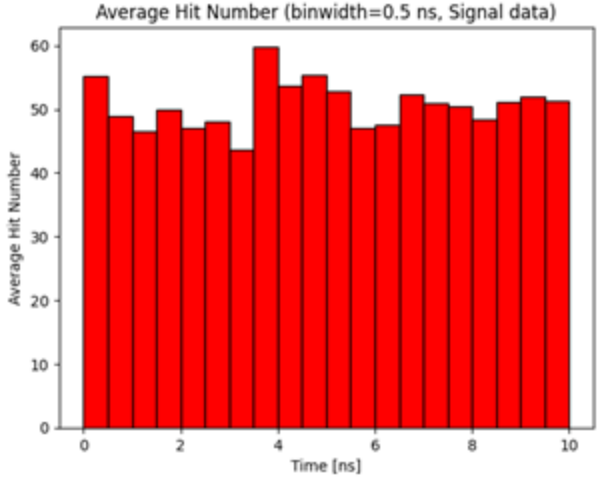
BACKUP

DCR=100kHz


Average Hit Number (binwidth=2.54 ns, Noise data)
Average Hit Number
Time [ns]


Average Hit Number (binwidth=0.5 ns, Noise data)
Average Hit Number
Time [ns]

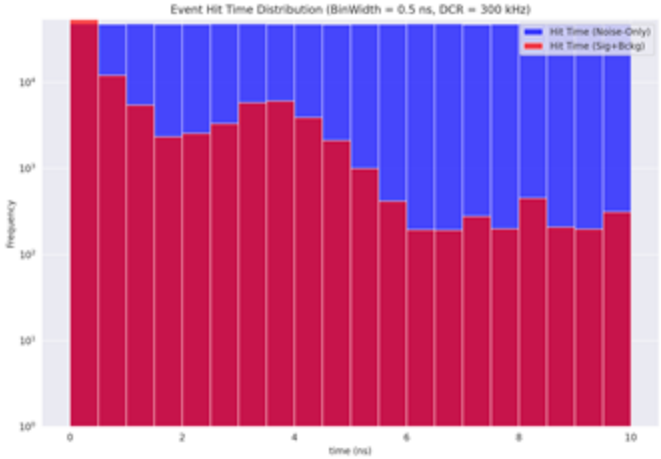

Average Hit Number (binwidth=2.54 ns, Signal data)
Average Hit Number
Time [ns]

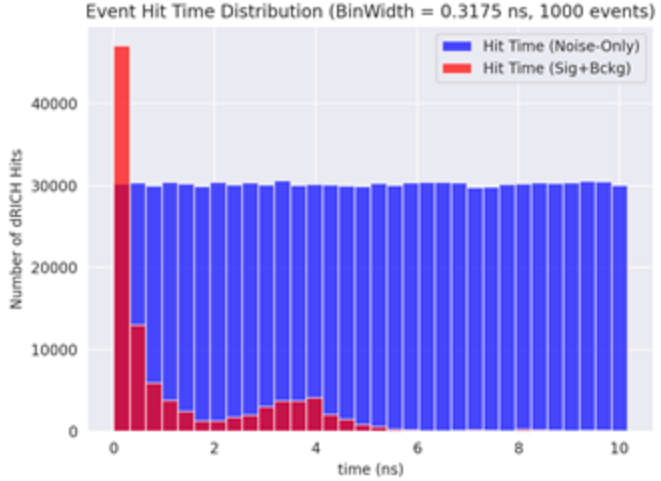

Average Hit Number (binwidth=0.5 ns, Signal data)
Average Hit Number
Time [ns]

Time normalization

(Questions)

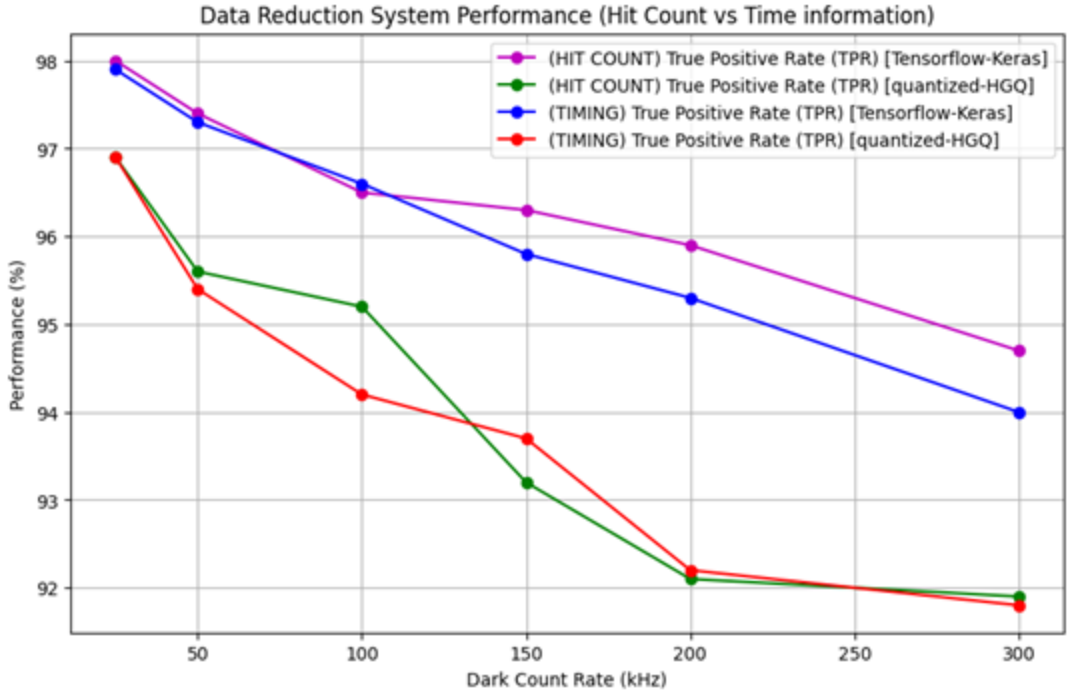
- ➔ How is the **event timeframe** structured?
- ➔ Is **bunch crossing timing** available in the ROOT data?
- ➔ Should hit distributions:
  ⇒ Peak near bunch crossing?
  ⇒ Or be spread within the acquisition window?





TPR comparison (no normalization)

P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)



- :(

Can we switch to another model?
⇒ CNN alternative

P = signal+background+noise
N = noise-only
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)

- Since the unexpected loss in performance when using the time information as input the MLP model, we tried to evaluate a different model
  ⇒ **CNN input layers** for DAM models, SectorMLPs remain the same

- **TPR** increases wrt the MLP one, however…
  ⇒ **CNN *hls4ml* istantiation interval: II ~ 4 clock cycles**
  ⇒ throughput ~ 50 MHz (@200MHz clock) < 100 MHz (EIC rate)

- **:(**