# Intelligent Experiment Through Real-Time AI: Fast Data Processing and Autonomous Detector Control for sPHENIX and Future EIC Detectors

Ming Liu
For the Fast-ML Team

Dec. 4th, 2024

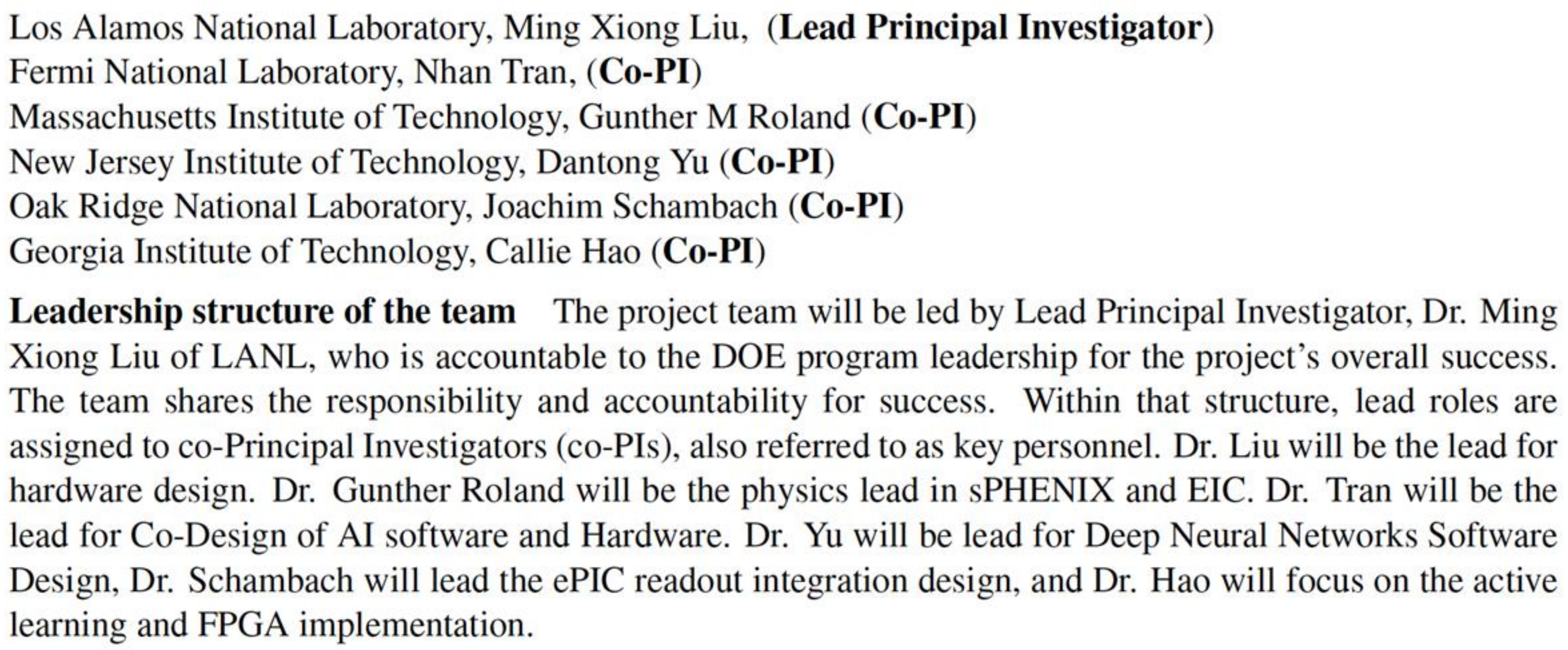
DOE NP AI-ML PI Meeting

**DOE Fast-ML project: FY24-25**

Team of NP, HEP, CS groups
- LANL (NP)
- MIT (NP, HEP)
- FNAL (HEP)
- NJIT (CS)
- ORNL (NP)
- CCNU (NP, HEP)
- UNT (CS)
- GIT(CS)

# Leadership and Technical Roles

Team of NP + HEP + CS/EE



Los Alamos National Laboratory, Ming Xiong Liu, (**Lead Principal Investigator**)
Fermi National Laboratory, Nhan Tran, (**Co-PI**)
Massachusetts Institute of Technology, Gunther M Roland (**Co-PI**)
New Jersey Institute of Technology, Dantong Yu (**Co-PI**)
Oak Ridge National Laboratory, Joachim Schambach (**Co-PI**)
Georgia Institute of Technology, Callie Hao (**Co-PI**)

**Leadership structure of the team** The project team will be led by Lead Principal Investigator, Dr. Ming Xiong Liu of LANL, who is accountable to the DOE program leadership for the project's overall success. The team shares the responsibility and accountability for success. Within that structure, lead roles are assigned to co-Principal Investigators (co-PIs), also referred to as key personnel. Dr. Liu will be the lead for hardware design. Dr. Gunther Roland will be the physics lead in sPHENIX and EIC. Dr. Tran will be the lead for Co-Design of AI software and Hardware. Dr. Yu will be lead for Deep Neural Networks Software Design, Dr. Schambach will lead the ePIC readout integration design, and Dr. Hao will focus on the active learning and FPGA implementation.

# DOE PI Meeting Presentations

**1. Overview**, 7’ (sPHENIX, EIC)

- **Ming Liu (LANL)/**Gunther Roland(MIT)/Nhan Tran(FNAL)/ Dantong Yu (NJIT)/Callie Hao (GIT)

**2. Physics simulation and AI-ML algorithms & plan for EIC**, 8’

- **Dantong Yu (NJIT)**/Giogian Borca-Tascuiuc(NJIT)/Cameron Dean(MIT)/Zhaozhong Shi(LANL) */Hang Qi(MIT)/Hao-Ren Jheng(MIT)/Pan Li(GIT)/*Yasser Corrales(MIT/LANL)

***3.* HLS4ML and firmware implementation**, 6’

- **Callie Hao** (GIT)/Hannah Bossi (MIT)/Jovan Mitrevski(FNAL)/Nhan Tran(FNAL)/Phil Harris(MIT)/Jakub Kvapil(LANL)

**4. Demonstrator implementation**, 7’

- **Jakub Kvapil (LANL)/Jo Schambach(ORNL)**/Yasser Corrales(MIT,LANL)**/**Kai Chen(CCNU)

***Q & A:*** *7’*

## Overview

- Ming Liu (LANL)

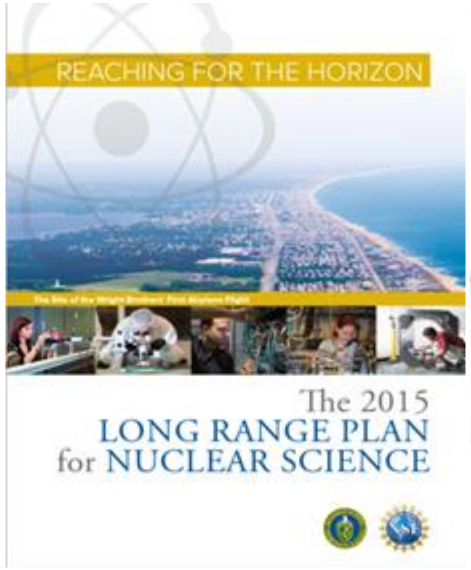
# sPHENIX at RHIC and the Future EIC

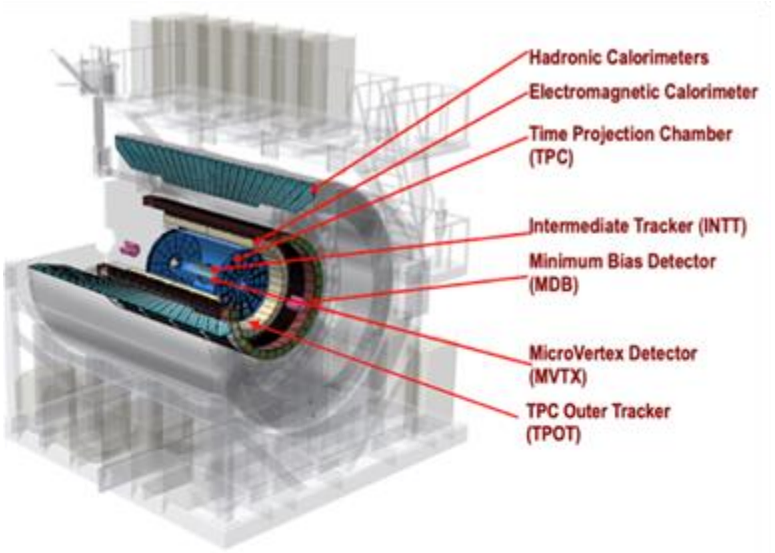
## 2015 NSAC Long Range Plan for Nuclear Science priority: sPHENIX Experiment at RHIC

- Probe the inner workings of QGP by resolving its properties at shorter and shorter length scales
- Complementary to LHC experiments to study relativistic heavy-ion collisions

**Heavy Quark physics – a key pillar of RHIC science**

Data Taking: 2023 – 2025+




Hadronic Calorimeters
Electromagnetic Calorimeter
Time Projection Chamber (TPC)
Intermediate Tracker (INTT)
Minimum Bias Detector (MDB)
MicroVertex Detector (MVTX)
TPC Outer Tracker (TPOT)

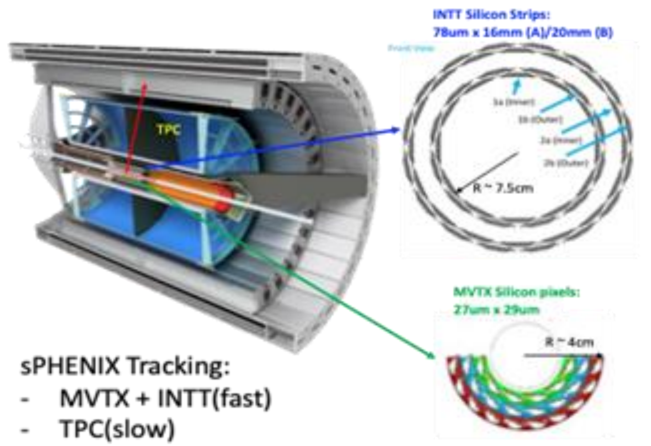
## 2023 NSAC priority: EIC at BNL

- Complete RHIC science mission
- EIC as the new NP flagship facility to study strong interactions and emergent phenomena

# Project Goals and Deliverables (I) ~ FY24

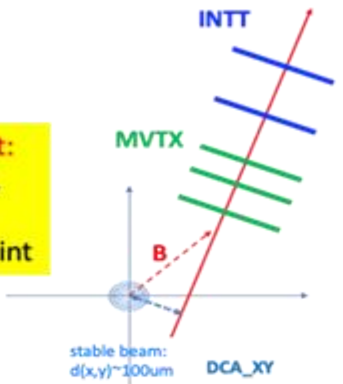
*- Heavy flavor event AI-trigger demonstrator in sPHENIX*

**Selective streaming real-time AI and autonomous detector control:**
Deliver a demonstrator for p+p and p+A running for sPHENIX - generalizable for applications in experiments at the EIC



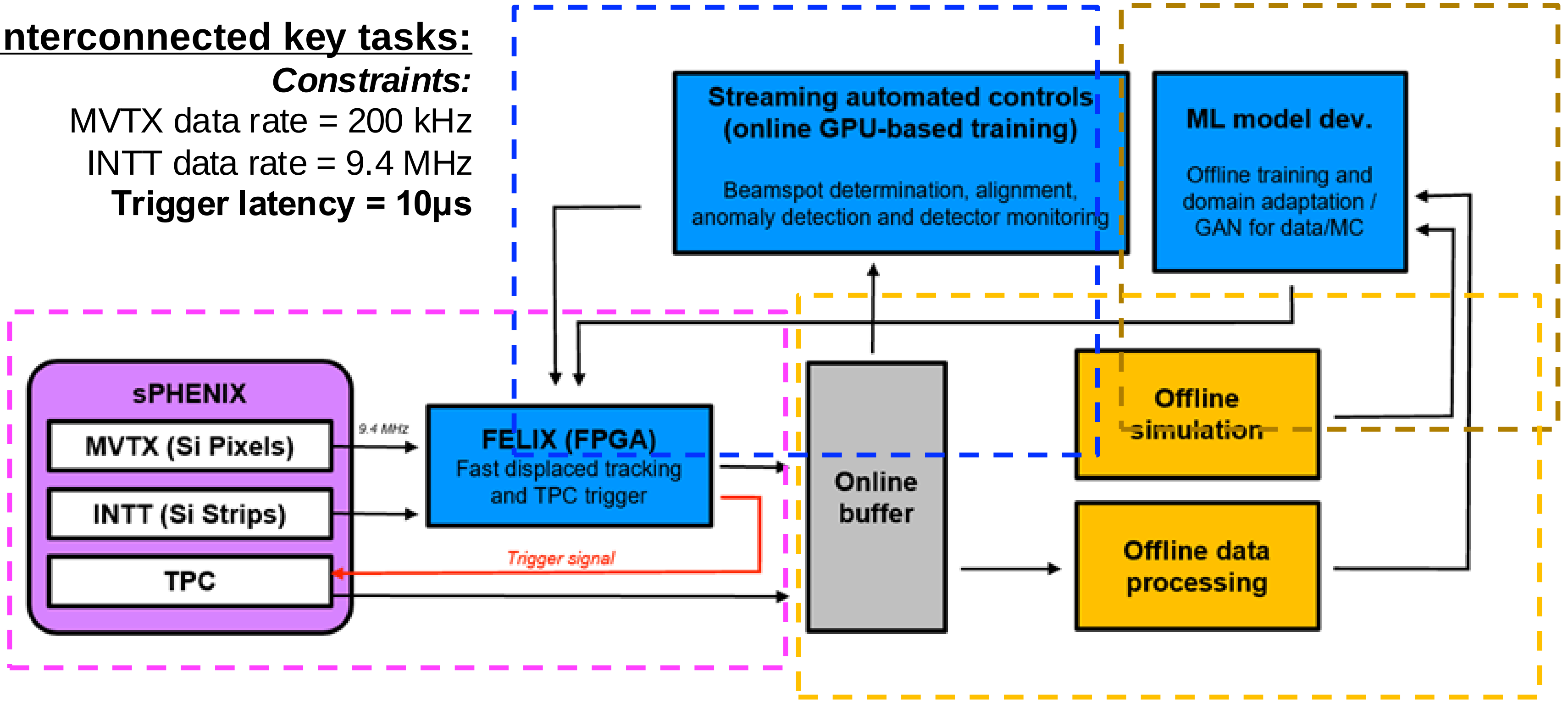
sPHENIX Tracking:
- MVTX + INTT(fast)
- TPC(slow)

Identify B-hadron event:
- Topology of B decay, with large DCA
- Monitor collision point



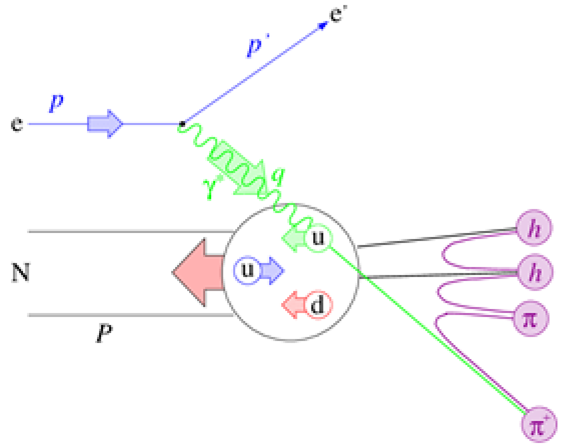
**4 interconnected key tasks:**

***Constraints:***
MVTX data rate = 200 kHz
INTT data rate = 9.4 MHz
**Trigger latency = 10μs**


Streaming automated controls (online GPU-based training)
Beamspot determination, alignment, anomaly detection and detector monitoring
ML model dev.
Offline training and domain adaptation / GAN for data/MC
sPHENIX
MVTX (Si Pixels)
INTT (Si Strips)
TPC
9.4 MHz
FELIX (FPGA)
Fast displaced tracking and TPC trigger
Trigger signal
Online buffer
Offline simulation
Offline data processing

## Project Goals and Deliverables (II) ~ FY25

*- DIS-electron Identification in Real-Time at EIC*



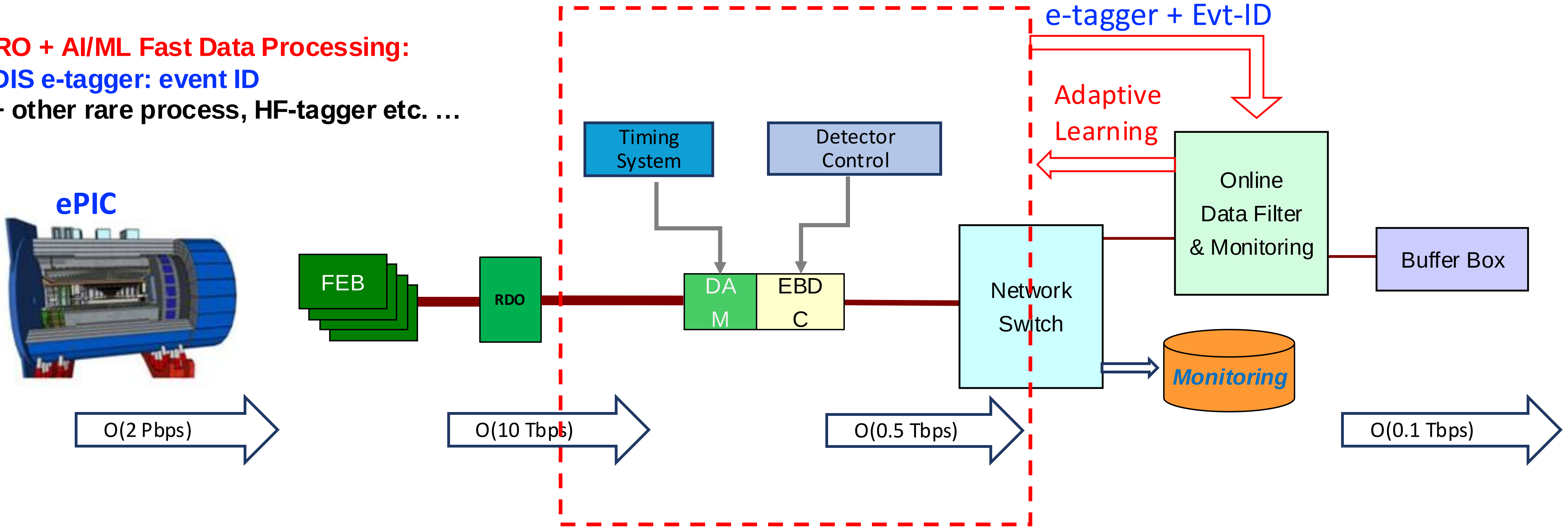
**Selective streaming readout for AI-Engine:**

- tag DIS-electron to define DIS event ID
  - ➢ EMCal + Trker + ePID
  - ➢ DCA~0
- tag other rare must-keep physics signals
  - ➢ HF with Trker etc.

**SRO + AI/ML Fast Data Processing:**

**- DIS e-tagger: event ID**

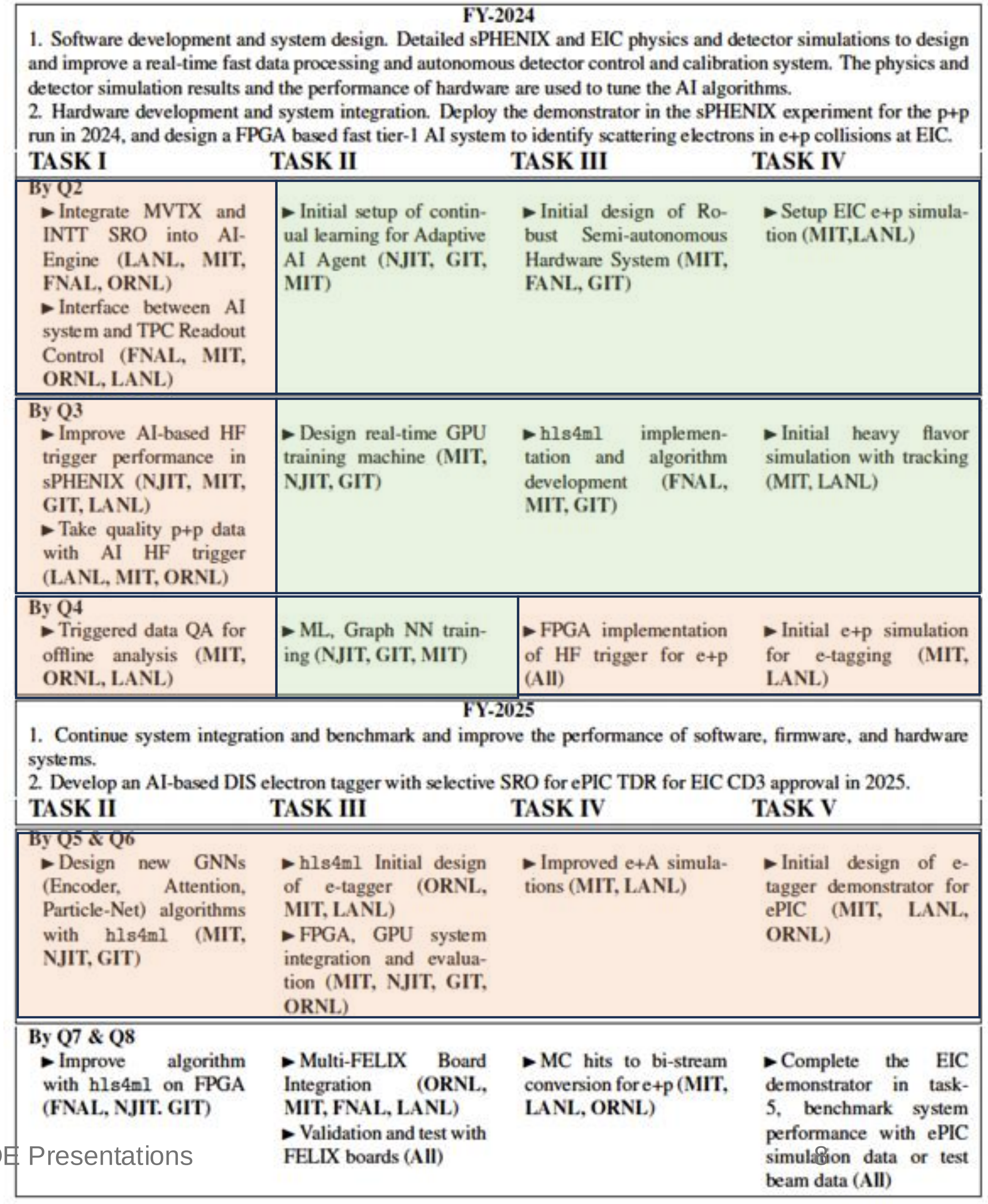
**+ other rare process, HF-tagger etc. …**



# Project Goals and Progress

- **FY-24:**
  - ➢ HF AI-Trigger demonstrator R&D
    - Robust AI-algorithms, **talk by Prof. Dantong Yu**
    - Firmware implementation, **talk by Prof. Callie Hao**
    - Deploy demonstrator in sPEHNIX, **talk by Dr. Jakub Kvapil**
  - ➢ sPHENIX experiment challenges in 2024
    - Delayed TPC and INTT commissioning, impacted FastML integration
      - Delayed INTT SRO deployment
      - Much delayed TPC operation due to gas/HV-related issues
    - **Limited p+p collected with full sPHENIX tracking detectors**
  - ➢ sPHENIX has proposed to BNL PAC to run high statistics p+p run (also p+A and other small systems) with full sPHENIX tracking detectors in 2026
    - **Our stretch goal: deploy AI-HF system in run 2026**
  - ➢ Big lessons learned on streaming readout and beam background challenge at RHIC!

- **FY-25:**
  - ➢ DIS-electron AI-tagger for EIC
    - EIC detector design reconsideration and optimization
  - ➢ Prepare for challenges of streaming readout



**FY-2024**

1. Software development and system design. Detailed sPHENIX and EIC physics and detector simulations to design and improve a real-time fast data processing and autonomous detector control and calibration system. The physics and detector simulation results and the performance of hardware are used to tune the AI algorithms.
2. Hardware development and system integration. Deploy the demonstrator in the sPHENIX experiment for the p+p run in 2024, and design a FPGA based fast tier-1 AI system to identify scattering electrons in e+p collisions at EIC.

| TASK I | TASK II | TASK III | TASK IV |
|---|---|---|---|
| **By Q2** ►Integrate MVTX and INTT SRO into AI-Engine (LANL, MIT, FNAL, ORNL) ►Interface between AI system and TPC Readout Control (FNAL, MIT, ORNL, LANL) | ►Initial setup of continual learning for Adaptive AI Agent (NJIT, GIT, MIT) | ►Initial design of Robust Semi-autonomous Hardware System (MIT, FANL, GIT) | ►Setup EIC e+p simulation (MIT,LANL) |
| **By Q3** ►Improve AI-based HF trigger performance in sPHENIX (NJIT, MIT, GIT, LANL) ►Take quality p+p data with AI HF trigger (LANL, MIT, ORNL) | ►Design real-time GPU training machine (MIT, NJIT, GIT) | ►`hls4ml` implementation and algorithm development (FNAL, MIT, GIT) | ►Initial heavy flavor simulation with tracking (MIT, LANL) |
| **By Q4** ►Triggered data QA for offline analysis (MIT, ORNL, LANL) | ►ML, Graph NN training (NJIT, GIT, MIT) | ►FPGA implementation of HF trigger for e+p (All) | ►Initial e+p simulation for e-tagging (MIT, LANL) |

**FY-2025**

1. Continue system integration and benchmark and improve the performance of software, firmware, and hardware systems.
2. Develop an AI-based DIS electron tagger with selective SRO for ePIC TDR for EIC CD3 approval in 2025.

| TASK II | TASK III | TASK IV | TASK V |
|---|---|---|---|
| **By Q5 & Q6** ►Design new GNNs (Encoder, Attention, Particle-Net) algorithms with `hls4ml` (MIT, NJIT, GIT) | ►`hls4ml` Initial design of e-tagger (ORNL, MIT, LANL) ►FPGA, GPU system integration and evaluation (MIT, NJIT, GIT, ORNL) | ►Improved e+A simulations (MIT, LANL) | ►Initial design of e-tagger demonstrator for ePIC (MIT, LANL, ORNL) |
| **By Q7 & Q8** ►Improve algorithm with `hls4ml` on FPGA (FNAL, NJIT. GIT) | ►Multi-FELIX Board Integration (ORNL, MIT, FNAL, LANL) ►Validation and test with FELIX boards (All) | ►MC hits to bi-stream conversion for e+p (MIT, LANL, ORNL) | ►Complete the EIC demonstrator in task-5, benchmark system performance with ePIC simulation data or test beam data (All) |

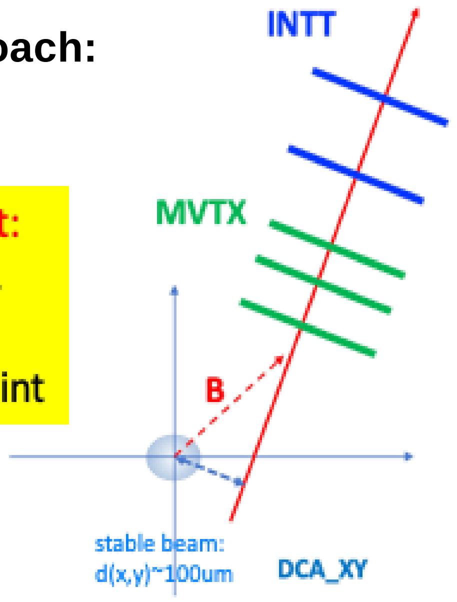
Dantong's talk

# New Finding … a good one

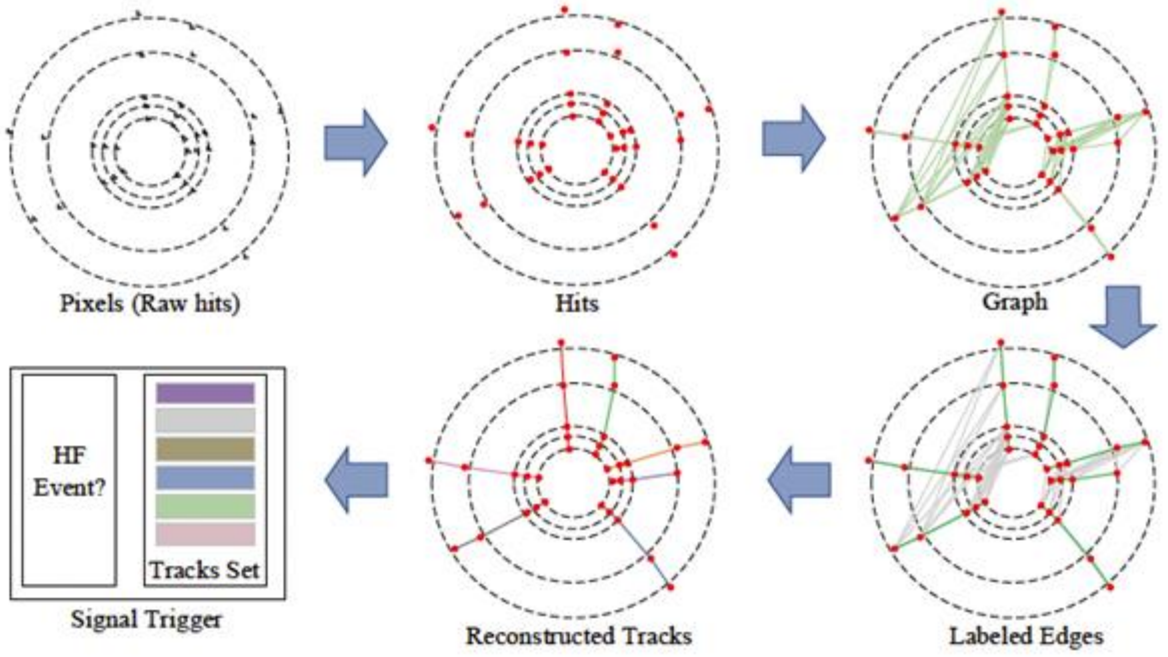
- AI HF-Trigger algorithms not sensitive to small IP shift!

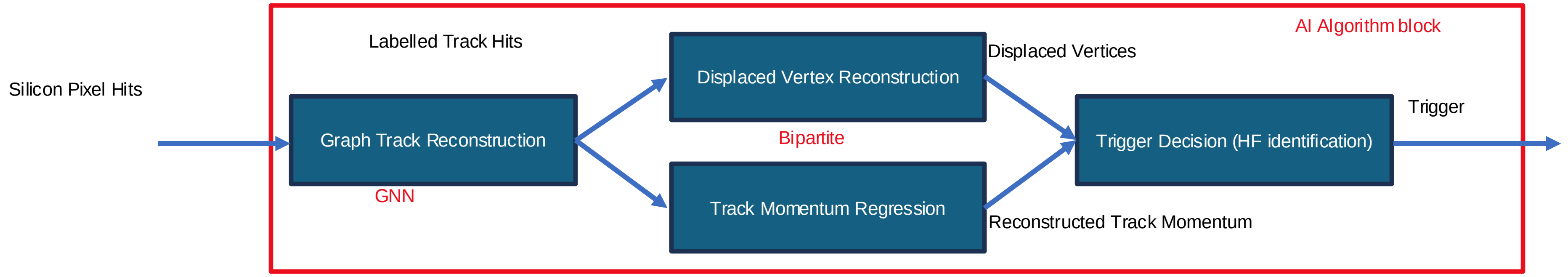
**Conventional approach:**

Identify B-hadron event:
- Topology of B decay, with large DCA
- Monitor collision point



**AI/ML approach:**




AI Algorithm block
Silicon Pixel Hits
Labelled Track Hits
Graph Track Reconstruction
GNN
Displaced Vertex Reconstruction
Bipartite
Track Momentum Regression
Displaced Vertices
Reconstructed Track Momentum
Trigger Decision (HF identification)
Trigger

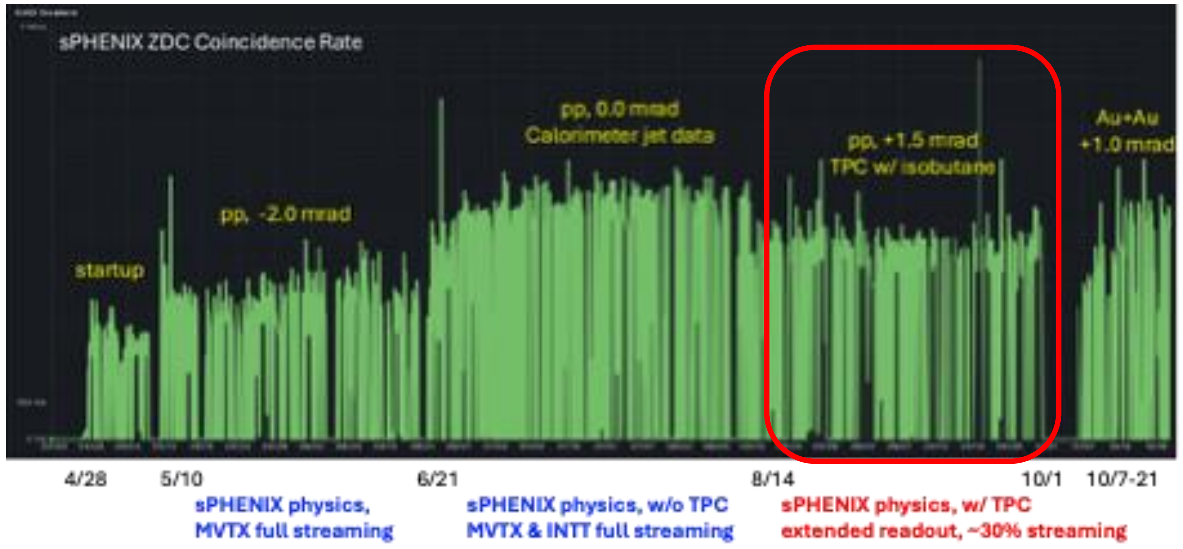
# Successes and Challenges in 2024

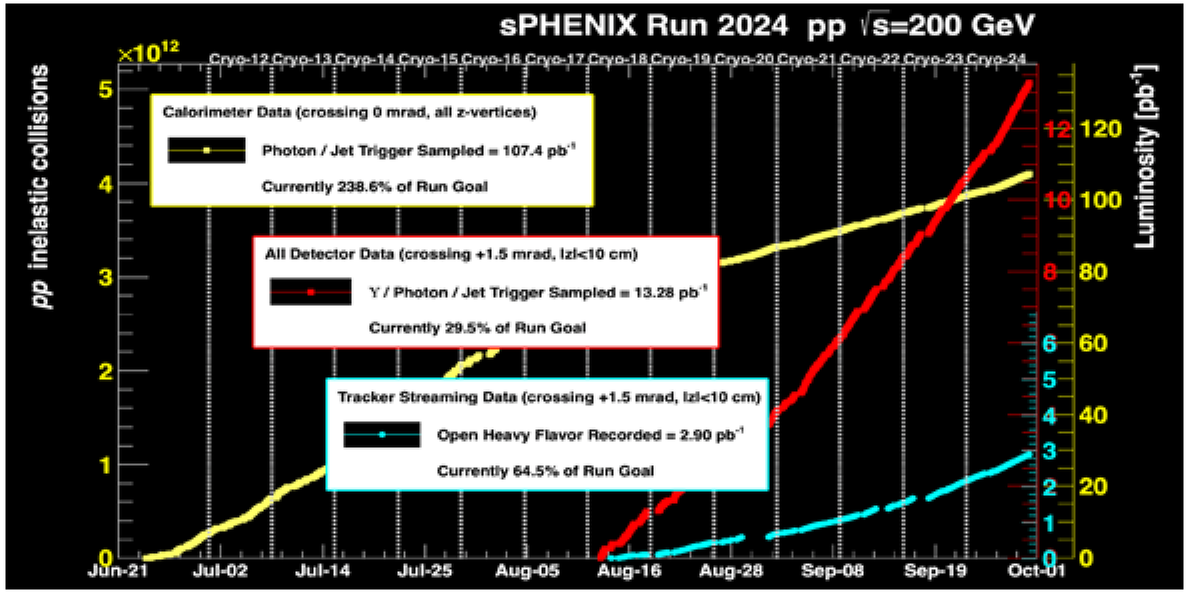
- **sPHENIX Run-24**
  - Installed and commissioned in 2024Run
    - p+p, and short AuAu commissioning run
  - MVTX re-installed and SRO commissioned (~5/1)
  - INTT SRO commissioned, w/ delay (~6/21)
  - Delayed TPC operation (~8/14)

**Implemented and evaluated Fast-ML demonstrator with MVTX telescope setup at BNL in September**

- robust AI algorithms
- AI model in firmware
- evaluated with real sPHENIX p+p data
- sPHENIX DAQ and other subsystems not available for full system integration in 2024, plan for 2026 pp run

- **Work in progress and challenges**
  - further improve AI-algorithms and firmware implementation
    - FPGA resource usage
    - Trigger latency
    - algorithm performance
  - MVTX and INTT SRO integration into FPGA/AI-Trigger
    - sPHENIX full DAQ system integration for TPC readout
- **Summary of expenditures**
  - Total budget, $1,600K (FY24-FY25);
  - About 50% spent, on track





# New Challenge Identified from sPHENIX 2024 Run

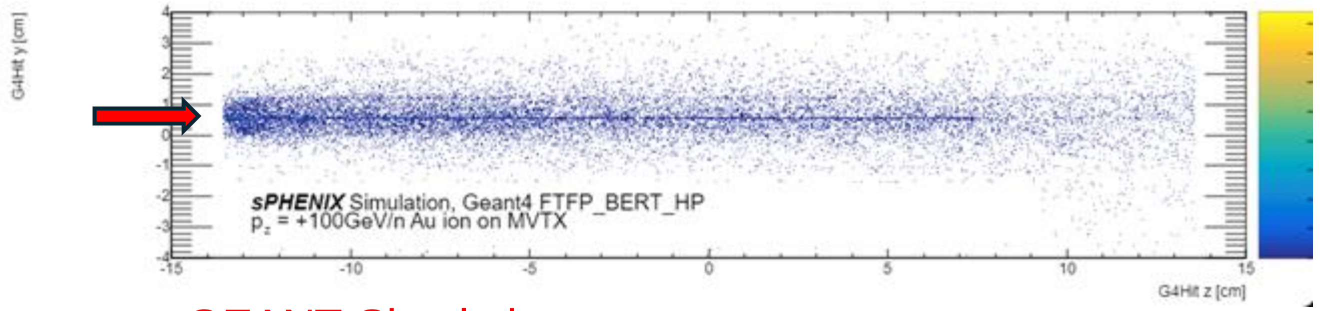
- Streaming readout in high beam background

- **NO problem in p+p collisions**
- **Major beam-related background with Au beam**
  - event with just one bunch of beam
  - Related to beam halo induced particles hitting sensor pixels in the MVTX detector sensors

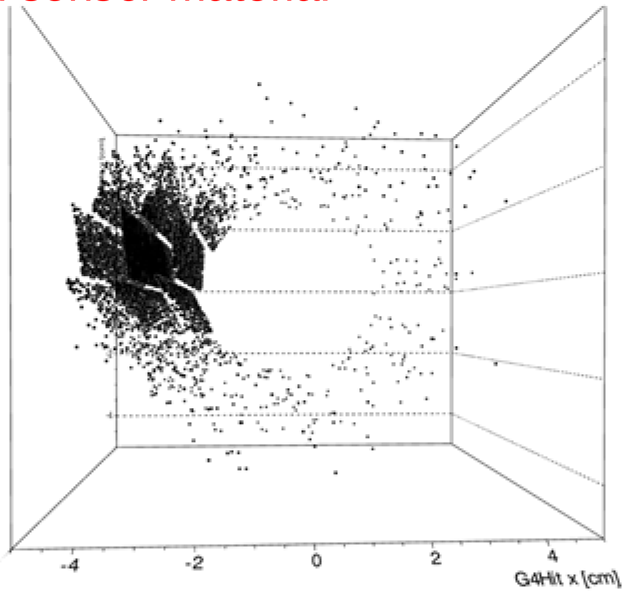
Data rate >> DAQ bandwidth! (>10^3)

EIC: phase-1, e+A program

- Could be in similar high beam background situation
- Smart data management desired on/near the detectors for full streaming readout in high background environment



GEANT Simulation:
Single 100 GeV Au ion striking the end of the 50um thick MVTX silicon sensor material



## Topical Highlights

# Physics simulation and AI-ML algorithms

- Dantong Yu and Giorgian Borca-Tasciuc (NJIT)

## Overview

- Goal: Create an efficient, end-to-end, robust trigger pipeline capable of handling event pileup
- Worst-case event pileup in p+p collisions: ~20 events
- Two stages of pipeline:
  - ➢ Stage 1: Tracking
    - Connect hits left by the same particle to create tracks
    - Reduce data size by eliminating hits left by pileup events
  - ➢ Stage 2: Trigger
    - Given tracks, accurately predict whether the event is a trigger event
- Given variation in interaction point, created algorithm to predict the interaction point
- Given variation in interaction point, perform trigger prediction
- Improve efficiency of algorithms by understanding which layers are important to trigger prediction

# Trigger Predictions

- Sliding scale between real experiment and ideal data
  - Experiment extreme: hits only + event pileup
  - Ideal extreme: ground truth tracks + no pileup
- Develop a *set* of models targeting each level between the worst-case experimental condition and the best-case data, allowing us:
  - Better understand the latency/accuracy trade-off
  - Use the more sophisticated models to verify the robustness of algorithms using data closer to the experiment
- Experimented with effect of interaction point (IP) offset and have verified small performance penalty from IP offset

## Trigger Prediction - Experiment Sliding Scale

More Experimental

1. GT Tracks, No Pileup ✓
2. Pred Tracks, No Pileup
   a. Tracking Algorithm - Predicts tracks ✓
   b. Trigger Algorithm - Takes predicted tracks and predicts trigger ✓
3. Hits, No Pileup ✓
4. Pred Tracks, Pileup ✓
   a. Tracking  Algorithm  - Predicts tracks and filters out pileup hits ✓
   b. Trigger Algorithm- Takes predicted and filtered tracks and predicts trigger  ✓
5. Hits, Pileup ✓

More Ideal

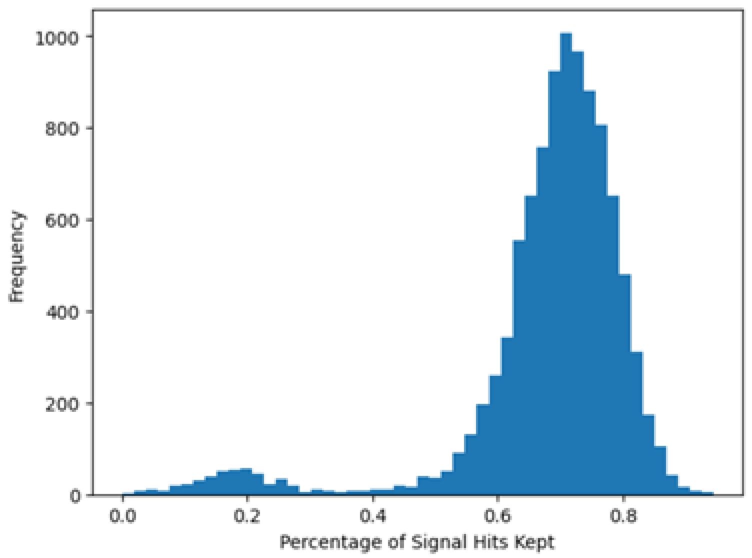
# Tracking Overview

- Edge candidates are created from hits using geometric criteria
  - Geometric criteria produces roughly O(n) hits, even with pileup date (usually ~2x as many edge candidates as there are hits)
- GNN classifies edge candidates based on hits information
- GNN also performs tracking depileup using INTT hits
- GNN trained to prioritize preserving edge candidates arising from trigger particles
- Created efficient, low-parameter count, FPGA-ready effective tracking algorithm

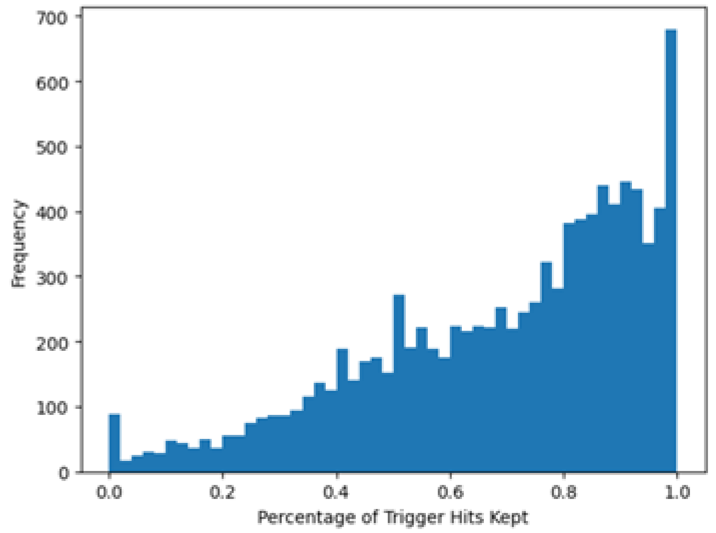
| Model Configuration | Precision | Recall | F1-Score |
|---|---|---|---|
| No Pileup | 92% | 90% | 91% |
| No Pileup, FPGA-ready | 79% | 87% | 83% |
| Pileup | 80% | 73% | 76% |

# Tracking Overview

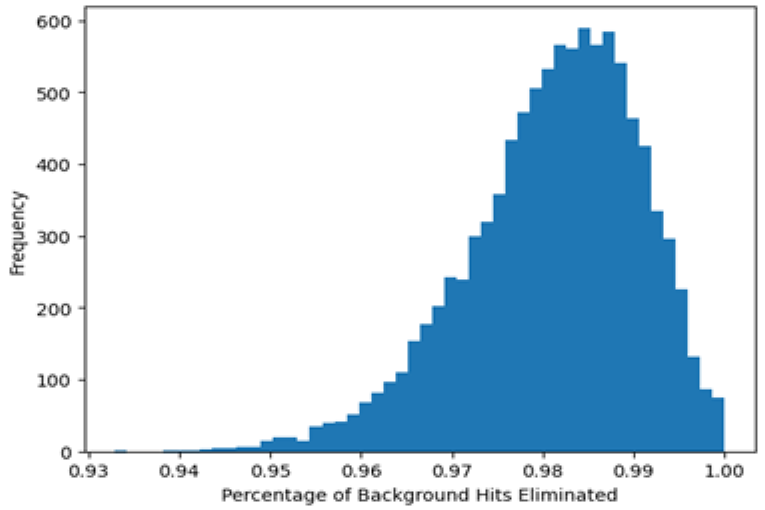
- Algorithm Effective:
  - ➢ Keeping non-pileup hits
  - ➢ Rejecting pileup hits
  - ➢ Keeping Trigger Hits



Keeping of non-pileup hits



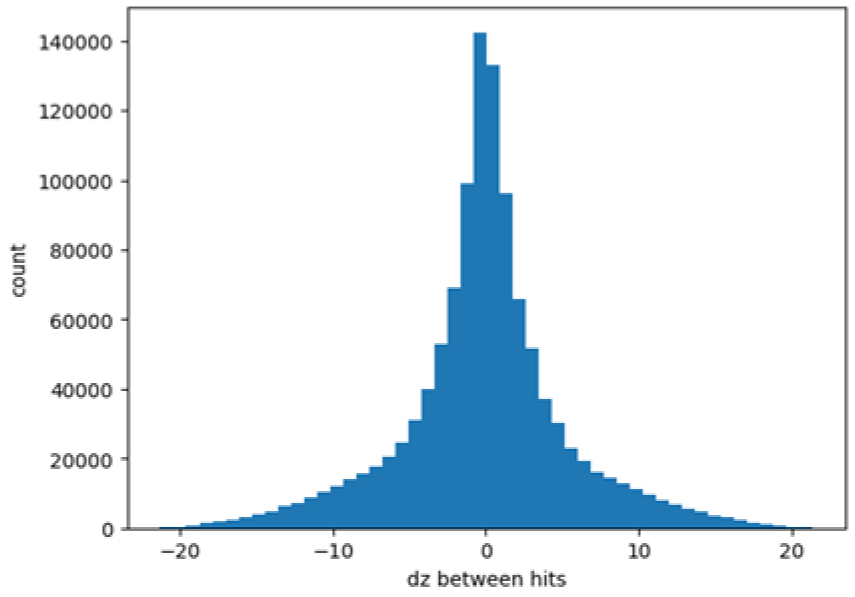
Keeping of trigger hits



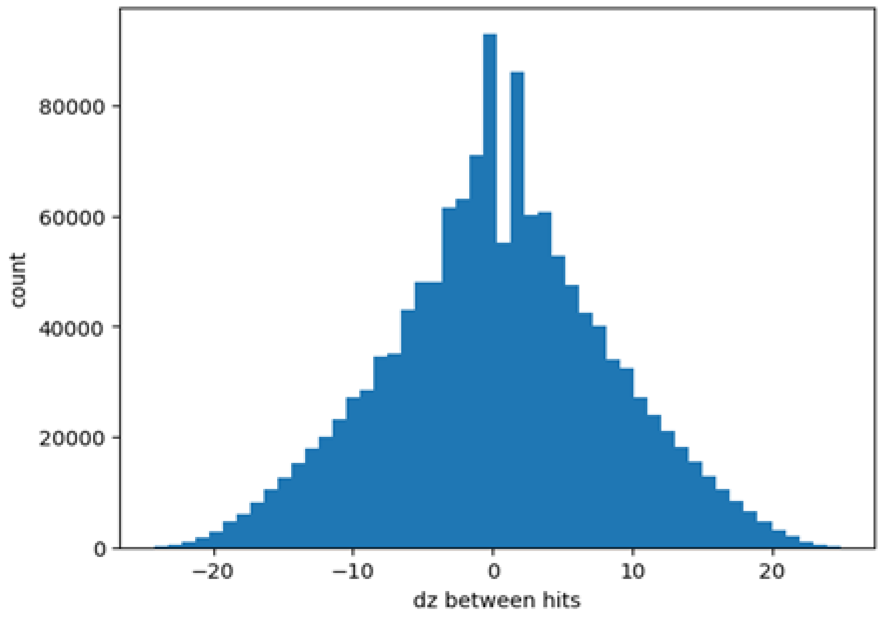
Rejection of pileup hits

## Tracking Algorithm Overview

- Tracking algorithm needs to reject beam halo background, which is not in simulation data
- Halos tend to have high ΔZ between hits as they appear parallel to the beamline.
    - ➢ Verify ability to reject halos by looking at distribution of ΔZ in rejected tracklets for real data vs simulation data
    - ➢ Real data shows a **fatter** distribution of rejected ΔZ than simulation data. We expect the real data to have more rejected tracks with high ΔZ, which we see when applying the tracking model to the real data.



Rejected Tracklets in Simulation Data



Rejected Tracklets in Real Data

# Trigger Prediction - Experiment Sliding Scale

More Experimental

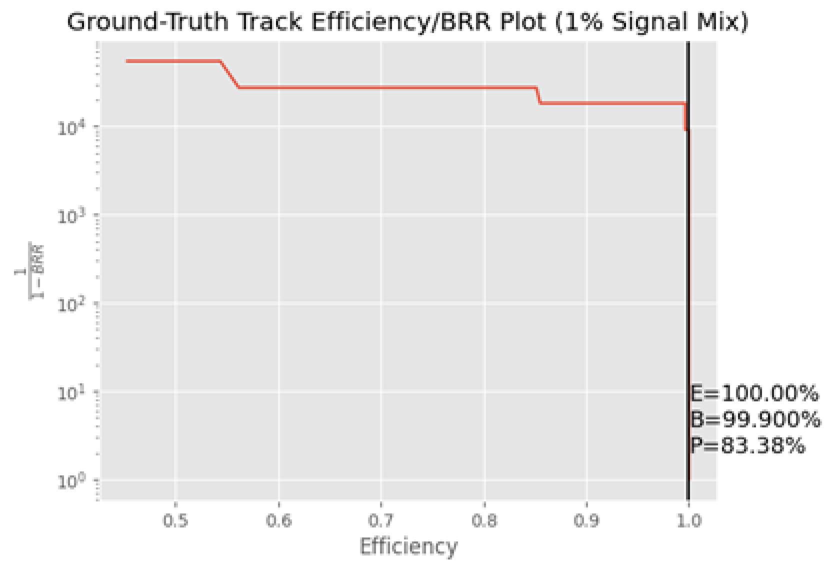
Results with latest data/model version (bbar):

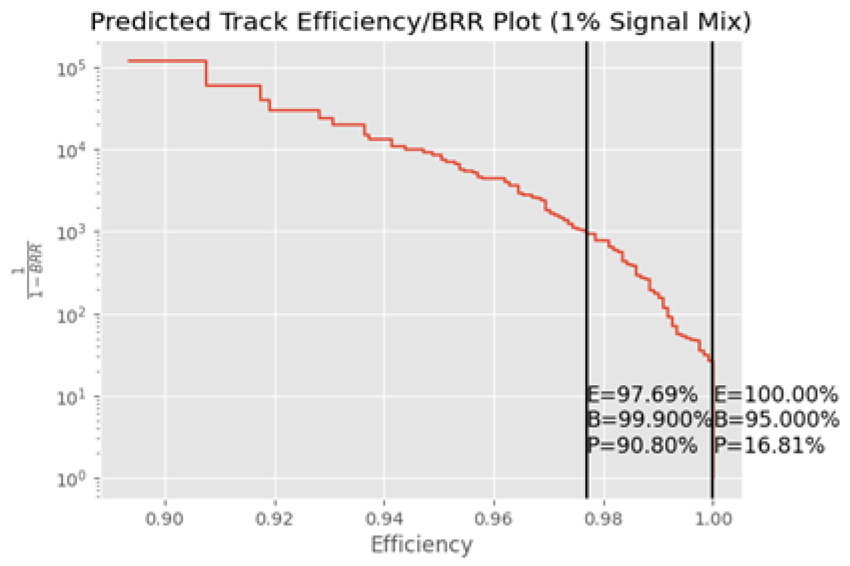
1. GT Tracks, No Pileup: 99.87% Accuracy
2. Pred Tracks, No Pileup: 99.12% Accuracy
3. Hits, No Pileup: 97.26%
4. Pred Tracks, Pileup
5. Hits, Pileup: 97.07%

Work in progress - improving and verifying the fidelity of the simulation

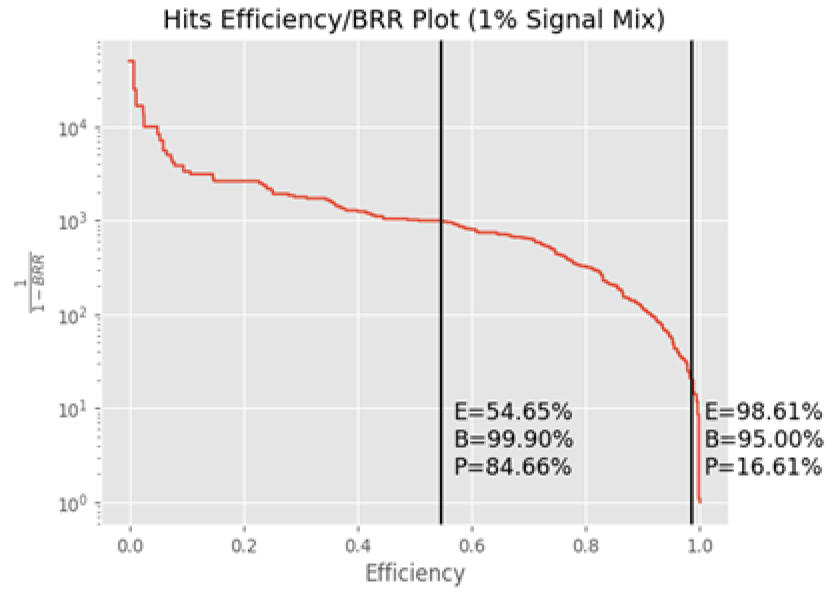
More Ideal

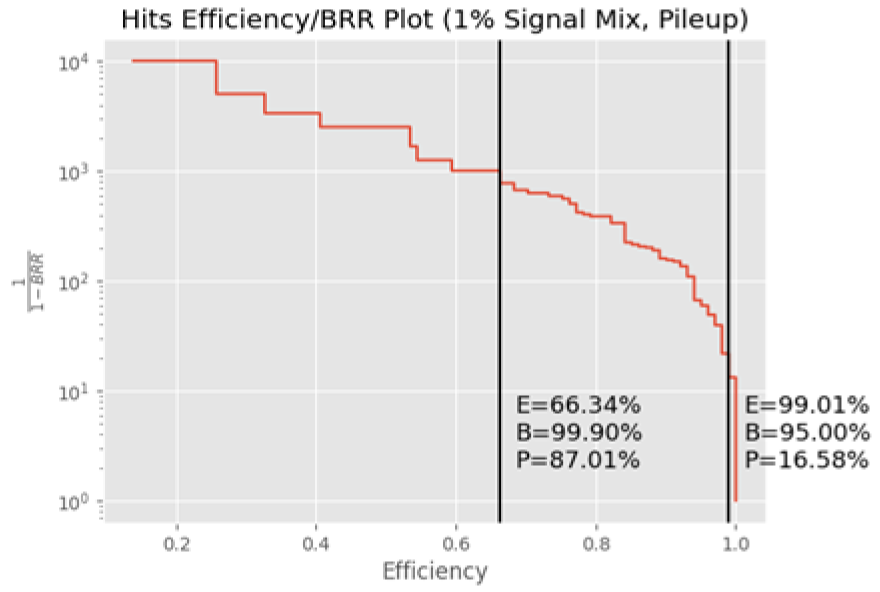
# Trigger Prediction





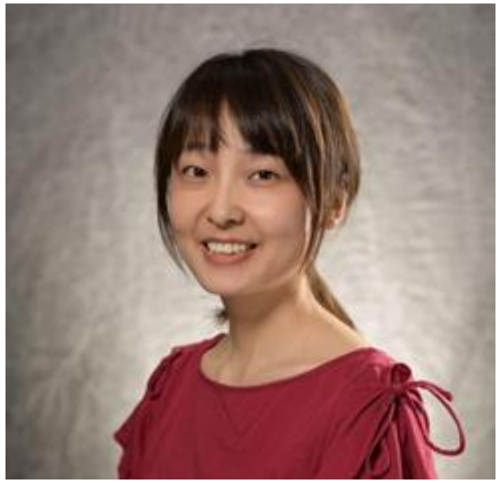
E: Efficiency
B: Background rejection rate
P: Purity





## Interaction Point (IP) Prediction and Effect

- Interaction Point in varies in the real event
- Predict the interaction point from the data
- Results:
  - ➢ $R^2$ = 99.6%
  - ➢ Maximum absolute error = 0.047cm
  - ➢ Root Mean Squared Error = 0.01cm
  - ➢ Mean Absolute Error = 0.008cm
- Impact of offset of interaction point on trigger accuracy:
- Apples-to-apples comparison:
  - ➢ Trigger Performance: 89.56% accuracy (w/offset), vs 91.53% accuracy (no offset)
  - ➢ Offset of IP leads only to a small drop (2%) in accuracy
  - ➢ No pileup used
- Take the model trained on without offset and see how does on the data with the offset

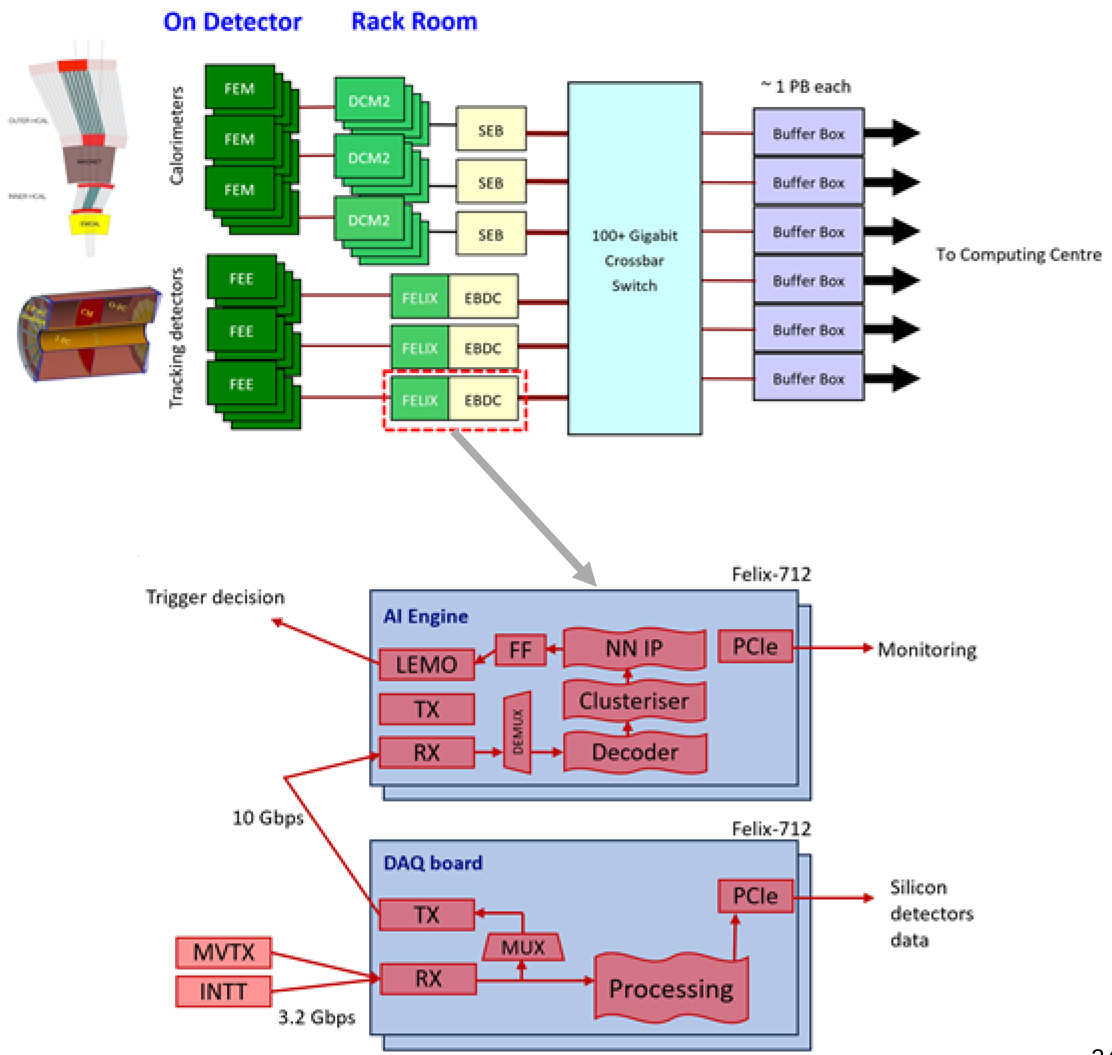


## *hls4ml* translation and firmware implementation

- Callie Hao (GIT) and Jovan Mitrevski (FNAL)

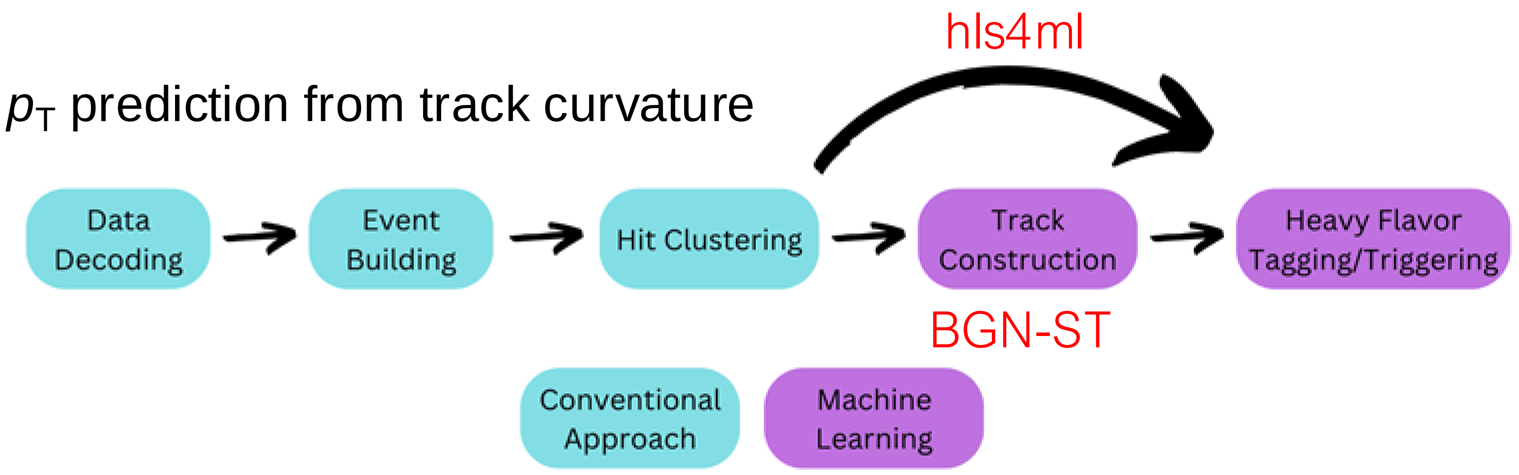
# Readout and HF AI-Trigger Implementation Plan

- The sPHENIX tracking detectors use FELIX 712 PCIe-based boards for the readout
  - ➢ Contain an AMD/Xilinx Kintex UltraScale FPGA (xcku115-flvf1924-2-e)
- To the readout DAQ boards, add AI Engine boards to perform the b-tagging using AI
- Exploring implement graph neural networks (GNNs) with two alternatives
  - ➢ FlowGNN (arXiv: 2204.13103)
  - ➢ hls4ml (arXiv: 1804.06913)



# Algorithm Pipeline

1. Hit decoding and clustering - conventional algorithms (will be discussed in next section)
2. Event building
3. Track reconstruction - using GNNs in two parts
   - Edge candidate generation - connect clusters (nodes) with edges, with geometric constraints
   - Edge candidate classification - using graph convolutional network (GCN) (arXiv: 1609.02907)
   - Construct final tracks
4. Use a least squares method to perform $p_T$ prediction from track curvature
5. Tagging of the heavy flavor signal



*Also consider an alternate implementation, taking the clusters directly without explicit track reconstruction.*

## Approach 1: FlowGNN

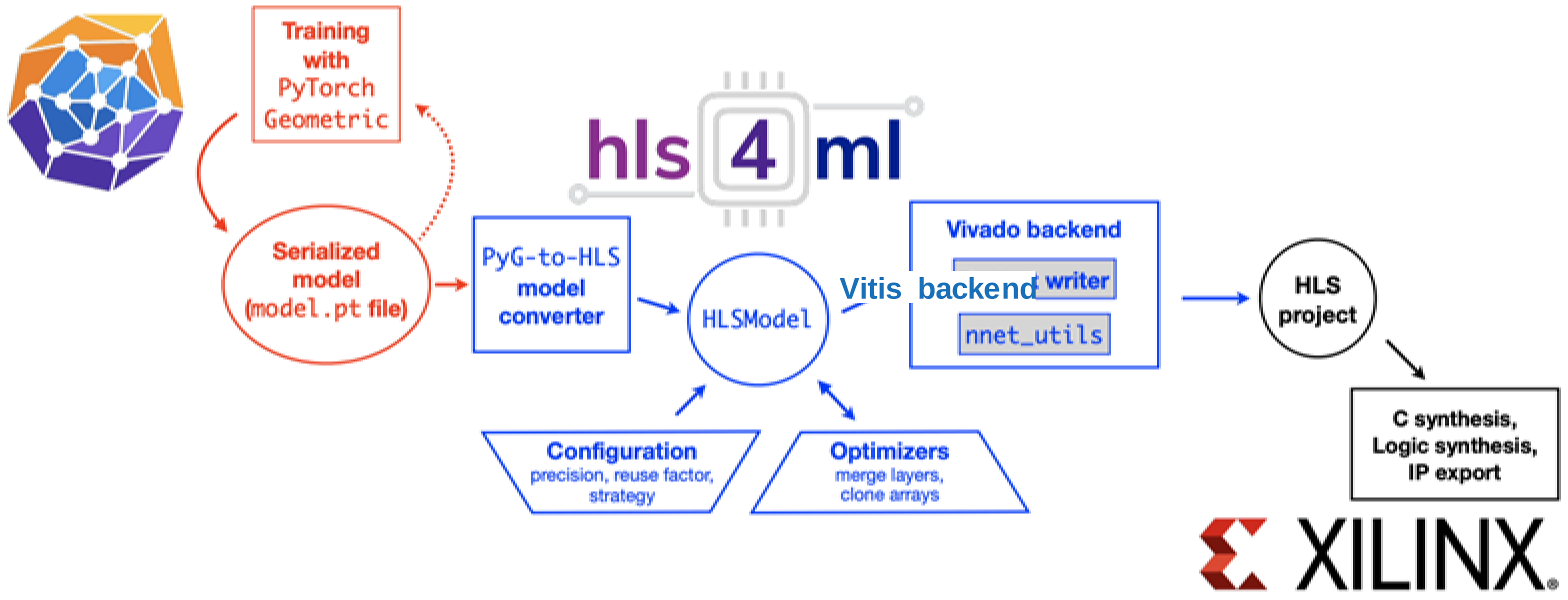
- FlowGNN is a flexible architecture for GNN acceleration on FPGAs https://arxiv.org/abs/2204.13103
- Two manual implementations, from PyTorch → C++ → Verilog, using High-Level Synthesis
  - ➢ Version 1: Track construction only:
    - 8.82 us per graph (Freq. 285 MHz), tested with: 92 nodes, 142 edges
  - ➢ Version 2: from Hit Clustering → Triggering:
    - 9.2 us per graph (Freq. 180 MHz), Tested with: 92 nodes, 142 edges

- (New this year) Extending to supporting more types of GNNs, e.g., EdgeConv, to facilitate better algorithm support
- (New this year) Perfecting the automation flow from PyTorch → Verilog, based on GNNBuilder https://arxiv.org/abs/2303.16459

• hls4ml (arXiv: 1804.06913)

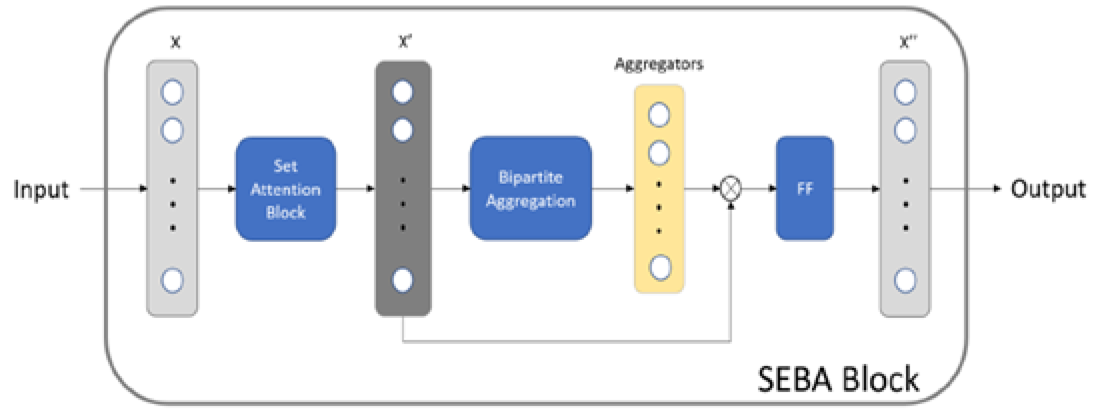
## Approach 2: hls4ml

- hls4ml is a compiler taking Keras, Pytorch, or ONNX input and producing High Level Synthesis (HLS) code implementing the network as spatial dataflow.
- HLS code is usually C++ or similar with directives to guide the produced hardware.
- hls4ml has different "backends" for the different flavors of HLS desired by tools.
- GNN support is under development: currently the process is not as automated as for other network types.


Training with PyTorch Geometric
Serialized model (model.pt file)
hls 4 ml
PyG-to-HLS model converter
HLSModel
Vitis backend
Vivado backend
writer
nnet_utils
HLS project
Configuration
precision, reuse factor, strategy
Optimizers
merge layers, clone arrays
C synthesis, Logic synthesis, IP export
XILINX

# Heavy Flavor Event Tagging

- Trigger detection using Bipartite Graph Network with Set Transformer (BGN-ST) (DOI: 10.1007/978-3-031-26409-2)
  - ➢ Input vectors contain a total of 37 features including: 5 hits (INTT + MVTX), length of each edge, angle between edges, total length of the edges, track radius
  - ➢ **Not yet supported in hls4ml**
  - ➢ **97.38%** accuracy for *b*-decays, no pileup



- **Current approach:** The initial implementation uses an **MVTX-only** MLP-layerwise tagger that takes the clusters directly:
  - ➢ Uses MLP to bring clusters into higher-level embedding space: layer wise pooling is done before final trigger prediction using aggregated layerwise embeddings.
  - ➢ Has better hls4ml support
  - ➢ **59.5%** accuracy for b-decays, with pileup (350 hits + 65 noise)
  - ➢ **88.5%** accuracy w/ above setup with INTT included.

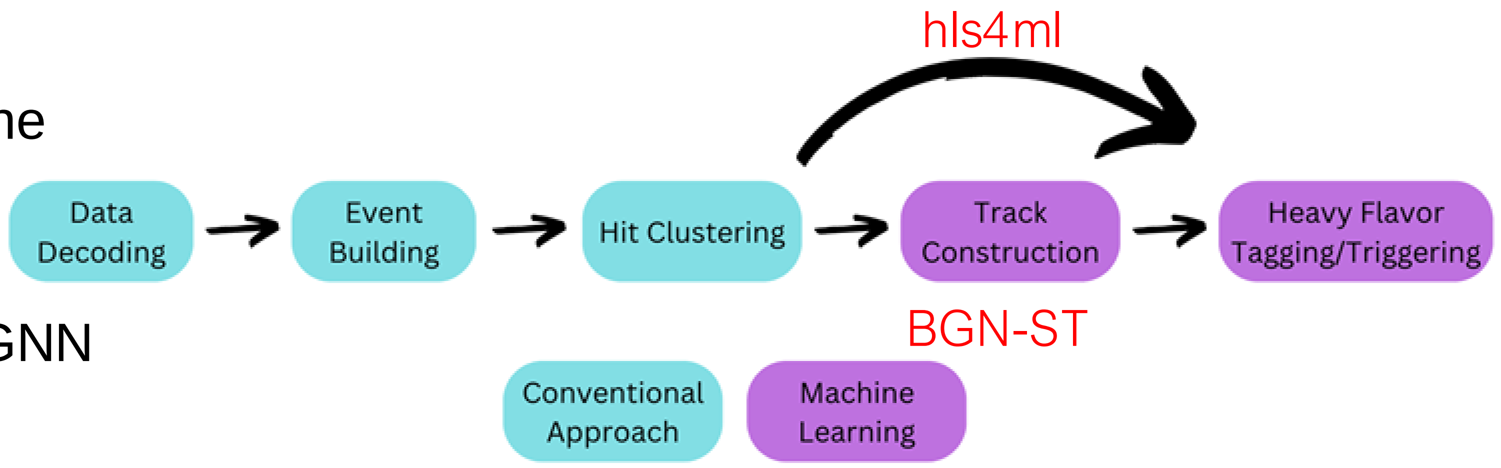
## hls4ml Initial Implementation (MVTX-only MLP)

- **The MLP-layerwise model has been synthesized for the FPGA**
- The model consists of two parts
  - The first part, called the **aggregation step**, collects all the clusters. It is called for each cluster in a bunch crossing. This needs a high throughput: initiation interval (II) every 1 clock cycle, 117 ns latency
  - The second part, called the **prediction step**, is called once per bunch crossing, to make a prediction based on the ingested clusters: II 63 clock cycles, 308 ns latency
- The two models are synthesized separately, with the FPGA utilization for the FELIX 712 given below, using Vitis HLS and Vivado 2024.1.

| | aggregation step | prediction step |
|---|---|---|
| LUT | 23 587 (3.56%) | 16 582 (2.50%) |
| FF | 15 129 (1.14%) | 31 226 (2.35%) |
| DSP | 19 (0.34%) | 498 (9.02%) |
| BRAM | 0 (0%) | 30.5 (1.41%) |

# Next Steps and Challenges

- ML-components of the analysis pipeline are synthesized for the FPGA.
- Future improvements
  - ➢ Further reduce utilization of flow GNN
  - ➢ Improve performance with pileup
  - ➢ Test the inclusion of INTT hit information



- In parallel, also working on a demonstrator for the full procedure!
  - ➢ See next few slides by Jakub and Jo!
- Need to synchronize input/output format for the ML with the other steps and start to test with real data!!

## Demonstrator Implementation

- Jakub Kvapil (LANL) and Joachim Schambach (ORNL)

# Demonstrator Implementation

- We are using FELIX-712 board as the target due to its use in sPHENIX readout
  - two boards, each analyzing a single hemisphere
- There are several modules to develop and validate:

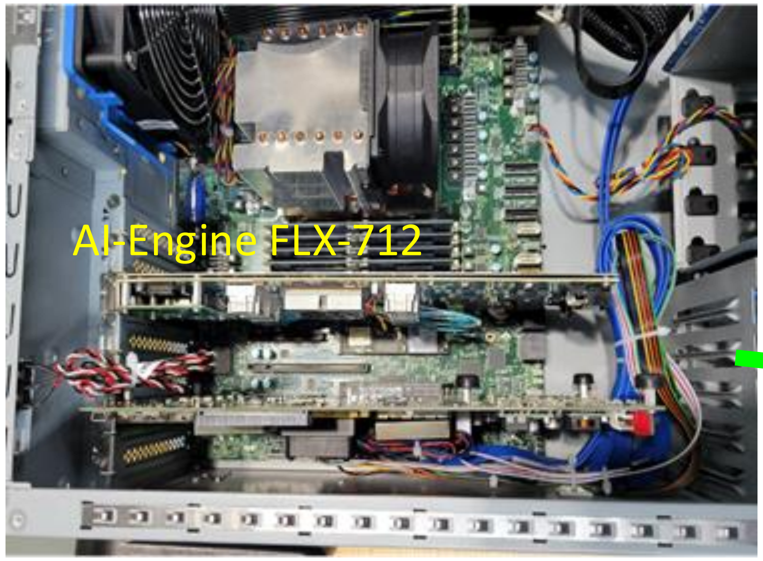
New from last year

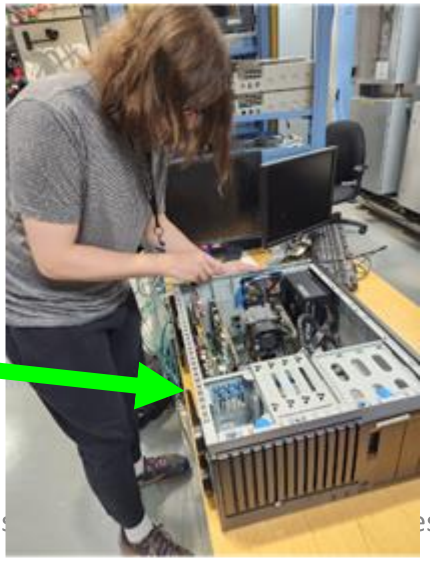
| Module | written | Validated - sim | Validated - test file | Validated - detector |
|---|---|---|---|---|
| PCIe comms | ✓ | ✓ new | ✓ new | ✓ new |
| Optics | ✓ | - | - | ✓ new |
| Decoder | ✓ | ✓ | ✓ new | ✓ new |
| Clusteriser | ✓ | ✓ (C++)<br>Ongoing (VHDL) new | Ongoing | Ongoing |
| Event build and coordinates transform | ✓ new | ✓ new | Ongoing | |
| AI module | ✓ FlowGNN<br>✓ hls4ml new | ✓<br>✓ new | Ongoing | |

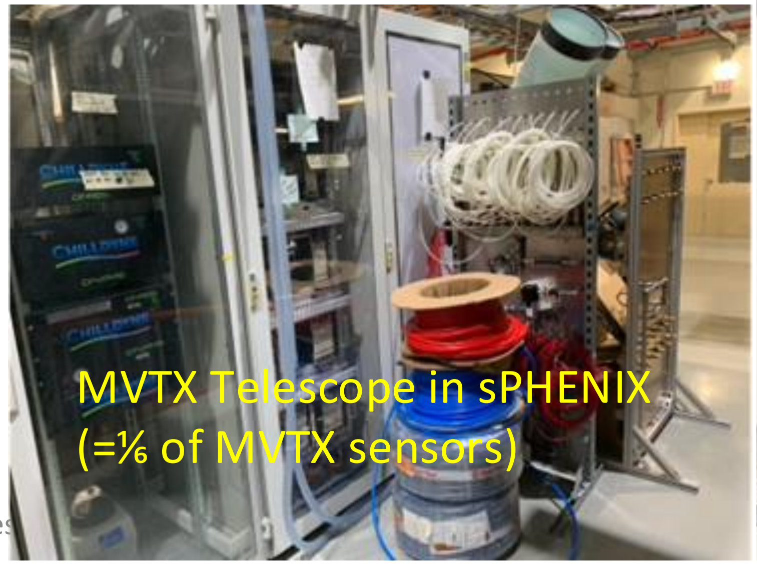
# PCIe and Detector Communication

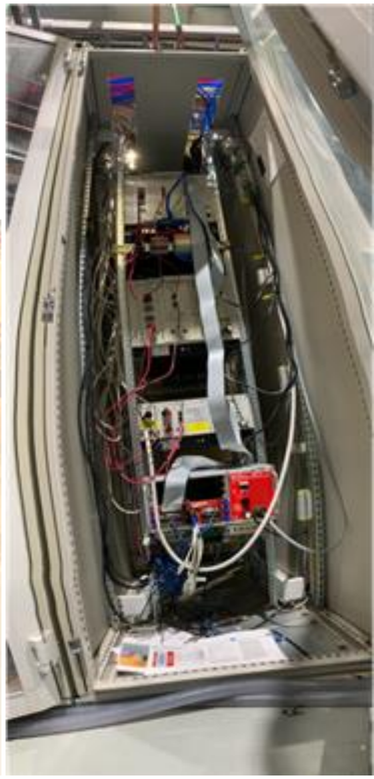
New from last year

- Since FELIX-712 was designed as readout board, the PCIe is used to receive data from the optics
  - We use this to save the timing (Bunch Crossing ID) and trigger decision from the AI
  - We have configured the PCIe uplink (normally used just for configuration) to load real detector data to the board, in order to have a controlled validation environment
- Successfully received and decoded data from single stave of the MVTX 8-stave telescope
- Added ILA via Xilinx virtual cable for additional debugging







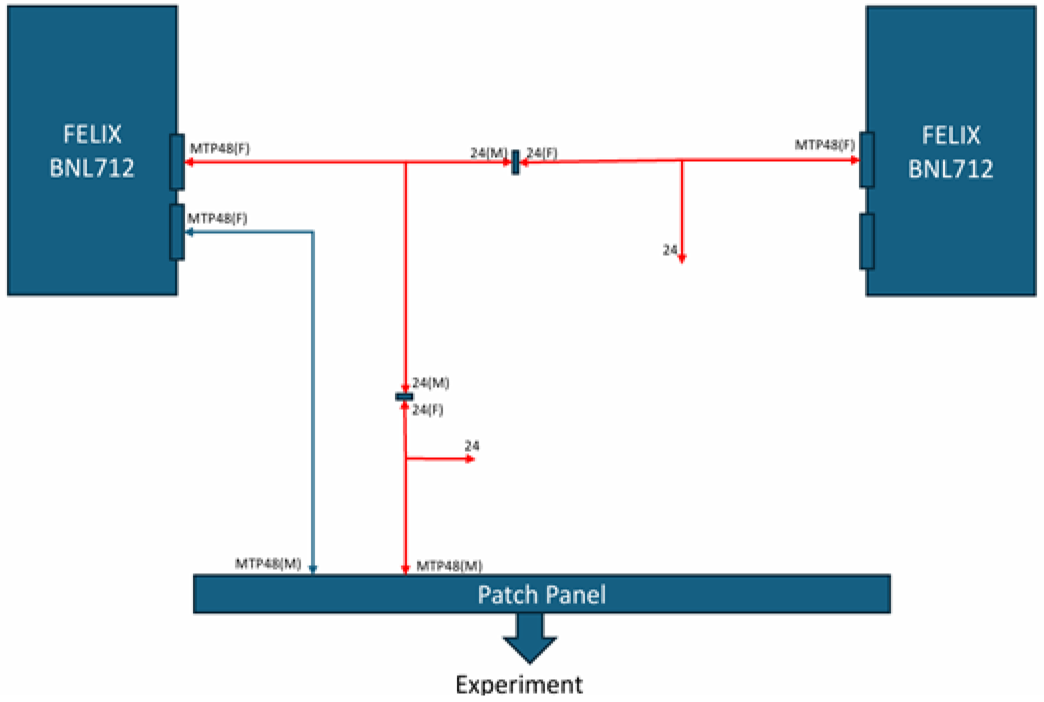


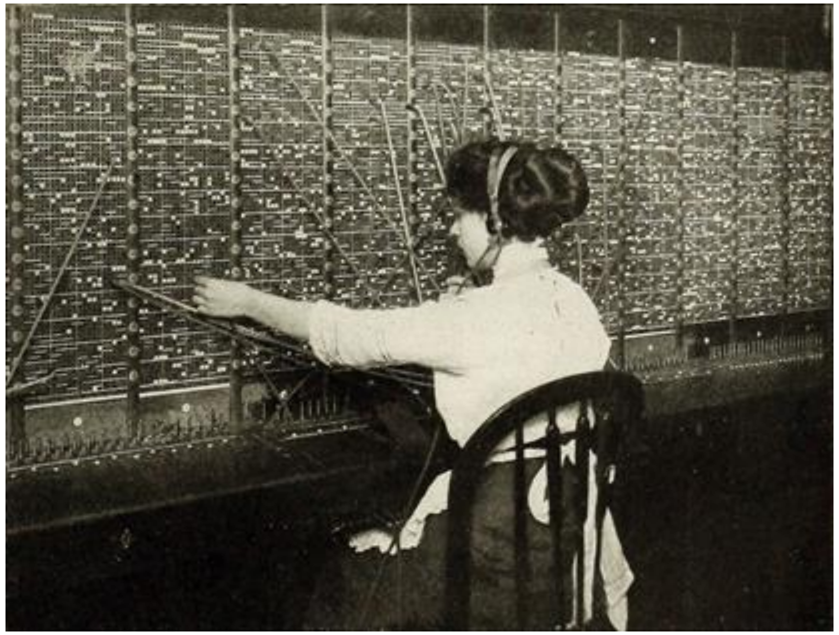
# PCIe and Detector Communication

New from last year

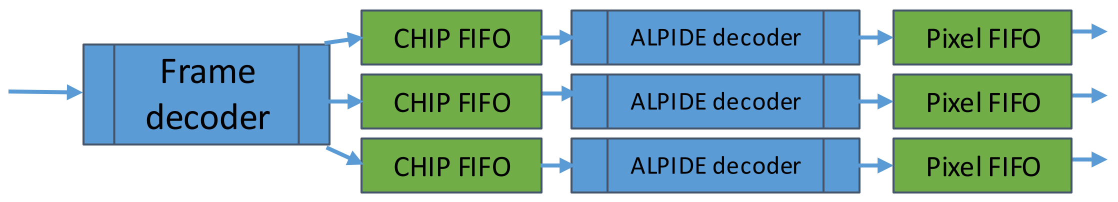
Challenge:

- MVTX has 6 FELIX readout servers with two optical connectors each with 48 links
- We had originally planned to use the 6x 2nd connectors to pass data to AI module
- However, the MVTX is now using both connectors to ease the single connector stress
    - There are still 24 links available on the second connectors, however now we cannot have easy geometry remap in FPGA and board to board connection
    - We **need to create a new optical switchboard**, to connect 288 links between detector and AI, to geometrically separate half-barrels





# Decoder Development



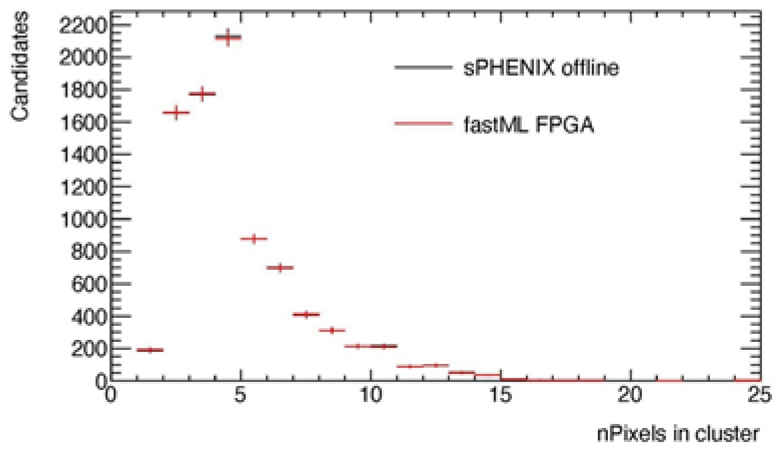
- First FPGA-based decoder for ALPIDE sensors
- New from last year:
  - ➢ The design has been simplified
    - There is only one set of buffers (instead of per event)
  - ➢ The design was validated on simulation, PCIe and Telescope data
    - Several bugs for edge cases have been fixed
    - This also helped to validate the PCIe and Telescope comms
  - ➢ Due to MVTX data compression we need 1 decoder module per detector (FeeID) link

| | LUT (663K) | FF (1.3M) | BRAM (2K) |
|---|---|---|---|
| Frame decoder | 151 | 287 | 0 |
| ALPIDE decoder (x3) | 343 | 256 | 0 |
| FIFOs (x6) | 31 | 36 | 1 |
| Total per FeeID | 1366 | 1271 | 6 |
| **Total per half- barrel** | **98K (14.7%)** | **91K (7%)** | **432 (21%)** |
| Last year's numbers | 189K (28.5%) | 102K (7.7%) | 648 (30%) |

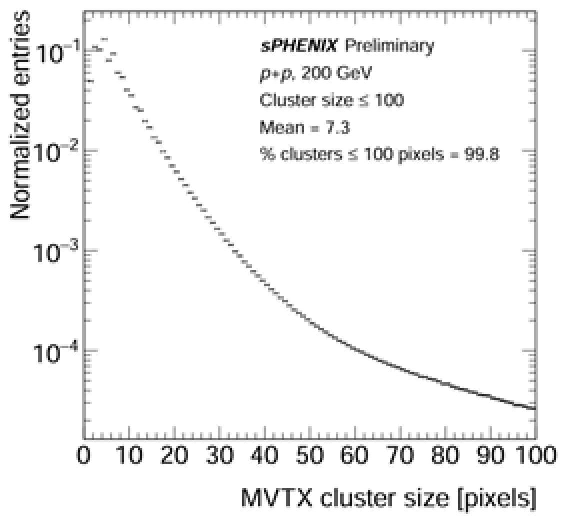
# Clusterizer Development

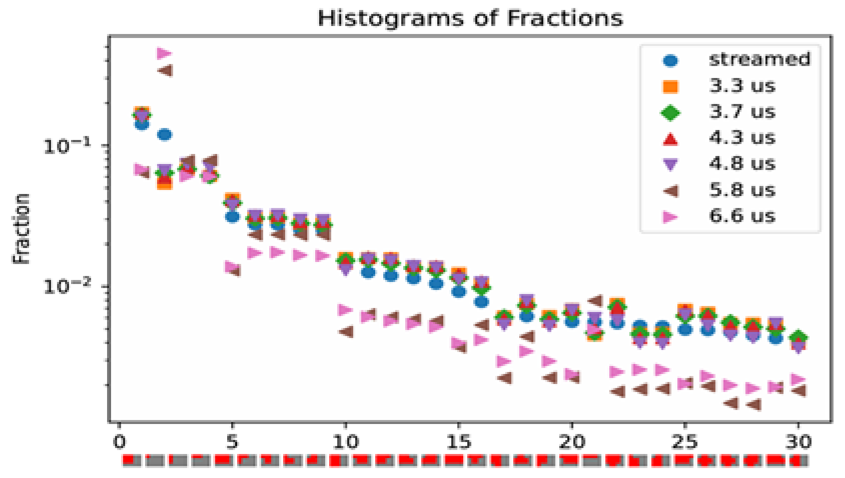
- Clusterizer is utilizing HLS workflow and provides 13.5 um precision
- New from last year:
  - The design has been integrated and monitoring has been added
  - Currently we are creating a test-bench validation framework
    - Will perform parallel on-FPGA validation next week
  - As we have the first pp data from sPHENIX it is found the cluster size is bigger than expected from detector simulation
    - Clusterizer was updated from maximum cluster size of 6 to 15
      - This unfortunately leads to doubling the FPGA resources

Simulation



Reality





# Clusterizer Development

**Challenges**

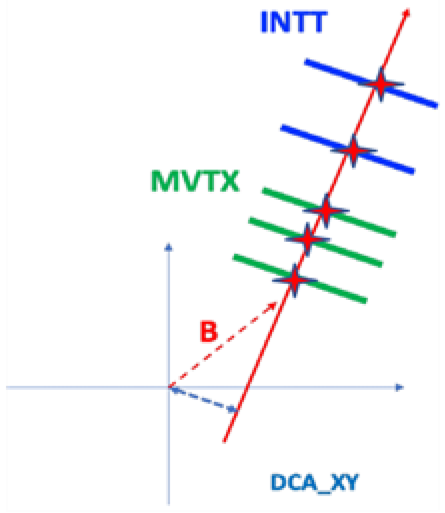
- The ideal design would be 1-cluster per chip (lowest latency); currently for dev we have 1 per stave
  - ➢ Worry is that the latency might be too high and throughput too low
  - ➢ By reusing the clusteriser we always need to wait to cluster 1st event adding 5us latency
- The target is to have one per FeeID, though 40% utilisation is still too much
  - ➢ Need to focus on reducing the utilisation and we have few possible directions how to do that

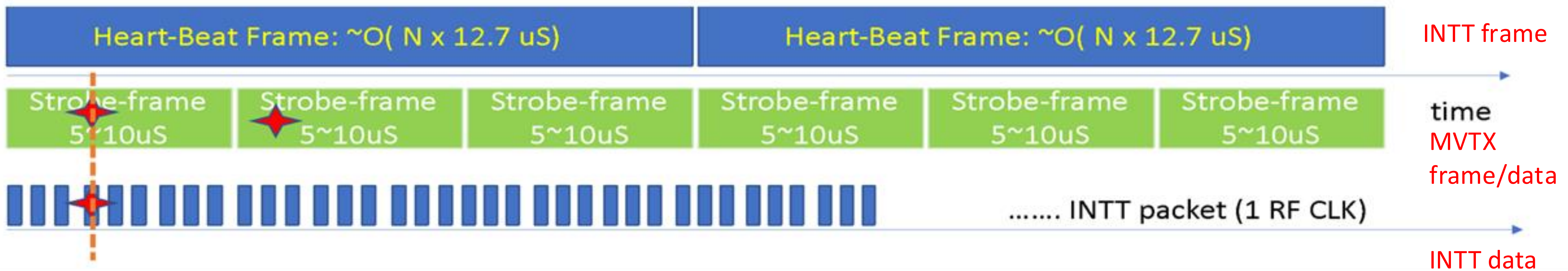
| | LUT (663K) | FF (1.3M) |
|---|---|---|
| Clustering | 3711 + 135 (memory) | 2964 |
| per chip (x216) | 801K (120%) | 640K (49%) |
| **per feeID (x72)** | **267K (40%)** | **213K (16.4%)** |
| per stave (x24) | 89K (13.4%) | 71K (5.4%) |

# Event Building

New from last year

- With the current MVTX-only setup the event building is easy
  - ➢ Since the detector links contain Bunch Crossing ID we can just read event by event link by link
- Challenge: once we add INTT stream this will be much more complicated due to different reading stream lengths and latencies
- Important is to first have the simpler MVTX-only implementation working!





# Coordinate Transformation

New from last year

- The clusterizer provides - layer, stave, chip, row, column
- The AI requires - layer, r, phi, z
- A new transformation module has been created to transform coordinates
- The BRAM usage is quite large, need to look into how to better parametrize the transformation

| | LUT (663K) | FF (1.3M) | BRAM (2K) | DSP (5.5K) |
|---|---|---|---|---|
| Clustering | 347 + 44 (memory) | 310 | 7.5 | 8 |
| per chip (x216) | 75K (11.2%) | 67K (5.1%) | 1620 (81%) | 1728 (31%) |
| **per feeID (x72)** | **25K (3.8%)** | **22K (1.7%)** | **540 (27%)** | **576 (10%)** |
| per stave (x24) | 8.3K (1.2%) | 7.4K (0.5%) | 180 (9%) | 192 (3.5%) |

# Resource Utilization

green: PCIe
purple: decoder
yellow: clusterizer
turquoise: local to global
brown: hsl4ml aggregate
pink: hsl4ml predict

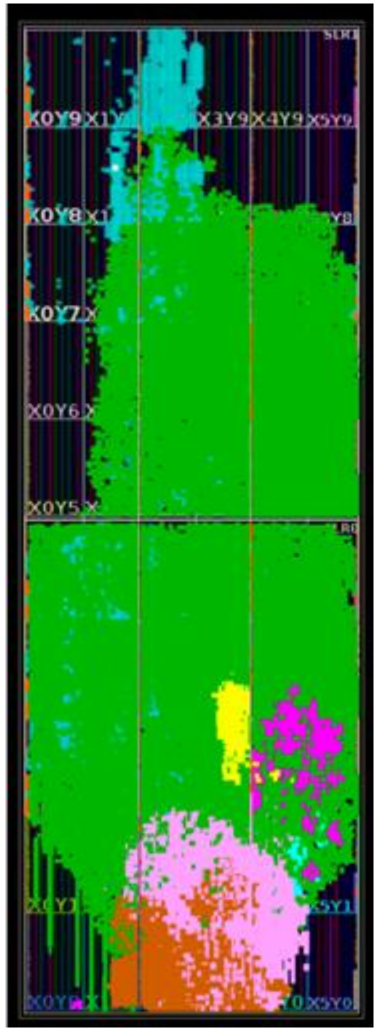
- Currently we have single stave implementation to validate modules
  - ➢ 3 decoders, 1 clusteriser, 1 transformation

| | LUT (663K) | FF (1.3M) | BRAM (2K) | DSP (5.5K) |
|---|---|---|---|---|
| 1-stave | 163K (24.5%) | 359K (27.6%) | 1K (50%) | 525 (9.5%) |
| 8-staves | 232K (35%) | 412K (31.6%) | 1.2K (60%) | 581 (10.5%) |

- Target is 72 decoders, clusterisers, and transformations
  - ➢ Current projection:

| | LUT (663K) | FF (1.3M) | BRAM (2K) | DSP (5.5K) |
|---|---|---|---|---|
| Infrastructure | 87K (13.1%) | 196K (14.8%) | 879 (40%) | - |
| Decoder | 98K (14.7%) | 91K (7%) | 432 (21%) | - |
| Clustering | 267K (40%) | 213K (16.4%) | - | - |
| Transformation | 25K (3.8%) | 22K (1.7%) | 540 (27%) | 576 (10.4%) |
| AI module (FlowGNN) | 194K (29%) | 214K (16.4%) | 406 (20%) | 488 (8.8%) |
| AI module (hls4ml) | 40K (6.1%) | 45K (3.5%) | 31 (1.5%) | 517 (9.4%) |

Need to reduce



# Summary of AI Demonstrator R&D

- Several models developed with various difficulties to implement on FPGA
- The MVTX decoder was optimized, and its size was reduced by ½
- The clusteriser was debugged, more sim validations in progress
  - The size unfortunately increased by factor 2 -> need to look at how to reduce it
- Coordinate transformation and event building is implemented
  - Need to look how parametrize the transformation to lower the BRAM usage
- We have a clear idea of the fiber optics switch box design and need to make it
- PCIe can be used to load actual MVTX pp data to test the AI-engine at around 65% throughput
- Current validation is done on 1-stave implementation
  - Changing it to 24-stave is just single parameter change
- Once MVTX-only chain is validated we will add INTT
  - We have started developing INTT decoder
  - But MVTX only design barely fits, we need to reduce its resource utilization first!
    - or consider analyzing ¼ of detector instead of ½ (need model study)
    - or daisy-chaining two FPGA (increase the latency)
- Plan to deploy the system in sPHENIX during the 2026 p+p (p+Au) runs