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. 5th, 2023
DOE NP AI-ML PI Meeting

DOE Fast-ML projects:
- FY22-FY23; 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)

Today's Presentations

1. Overview, 5'
- **Ming Liu (LANL)/**Gunther Roland(MIT)/Nhan Tran(FNAL)/ Dantong Yu (NJIT)/Callie Hao (GIT)

2. Physics simulation and AI-ML algorithms, 10'
- **Dantong Yu (NJIT)**/Giogian Borca-Tascuiuc(NJIT)/Cameron Dean(MIT)/Zhaozhong Shi(LANL) */Hang Qi(MIT)/Hao-Ren Jheng(MIT)/Beilei Jiang(NTU)/Pan Li(GaTech)*

3. **HLS4ML and firmware implementation**, 8'
- **Hannah Bossi (MIT)/Jovan Mitrevski(FNAL)**/Nhan Tran(FNAL)/Phil Harris(MIT)/Calli Hao (GaTech)

4. Demonstrator implementation, 7'
- **Jakub Kvapil (LANL)**/Yasser Corrales(MIT)/*Noah Wuerfel(LANL)*/Jo Schambach(ORNL)/Kai Chen(CCNU)/*Lang Lei*(CCNU)/*Beilei Jiang(NTU)*

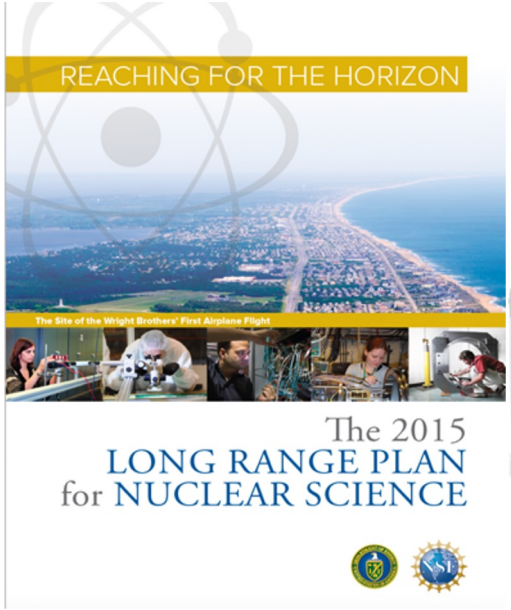
Q & A: *5'*

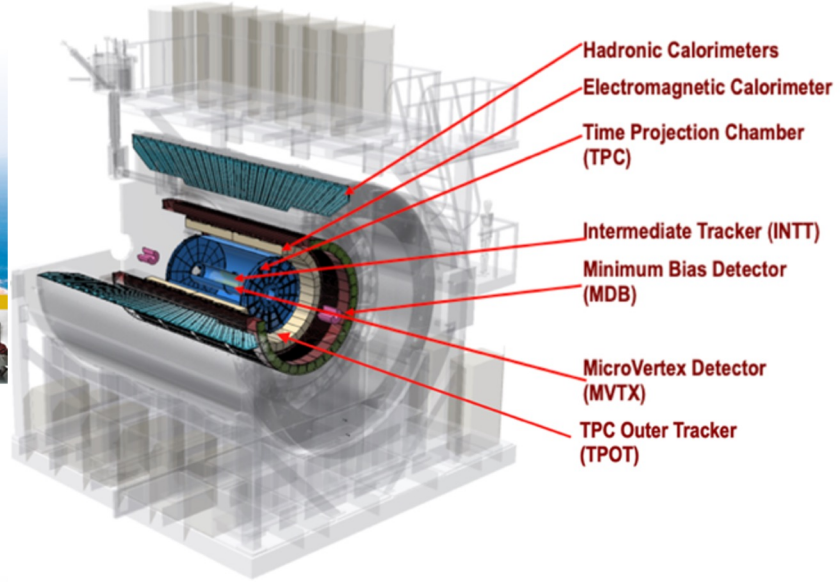
Overview

- Ming Liu (LANL)

Data Taking: 2023 - 2025

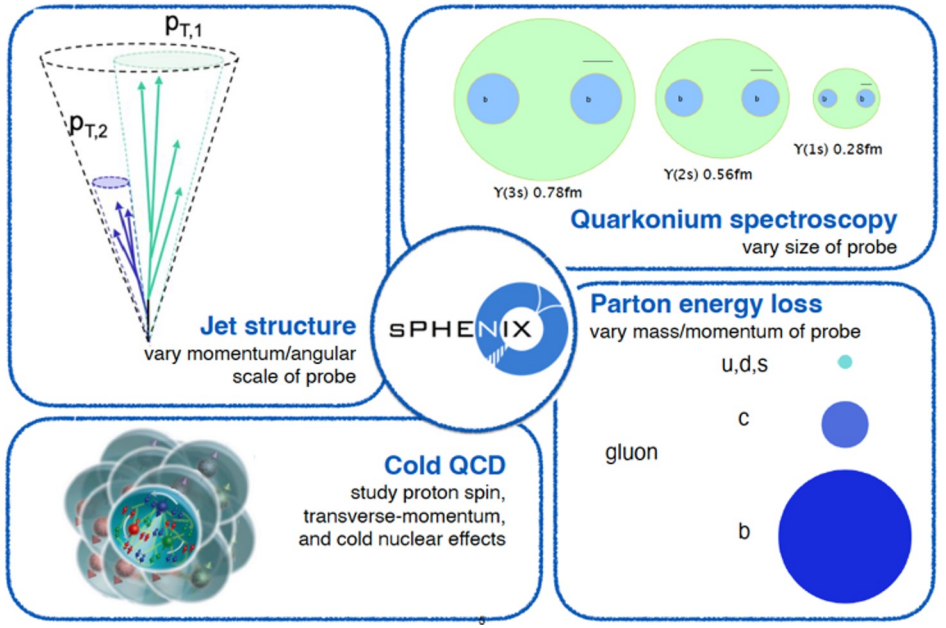
sPHENIX at RHIC





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



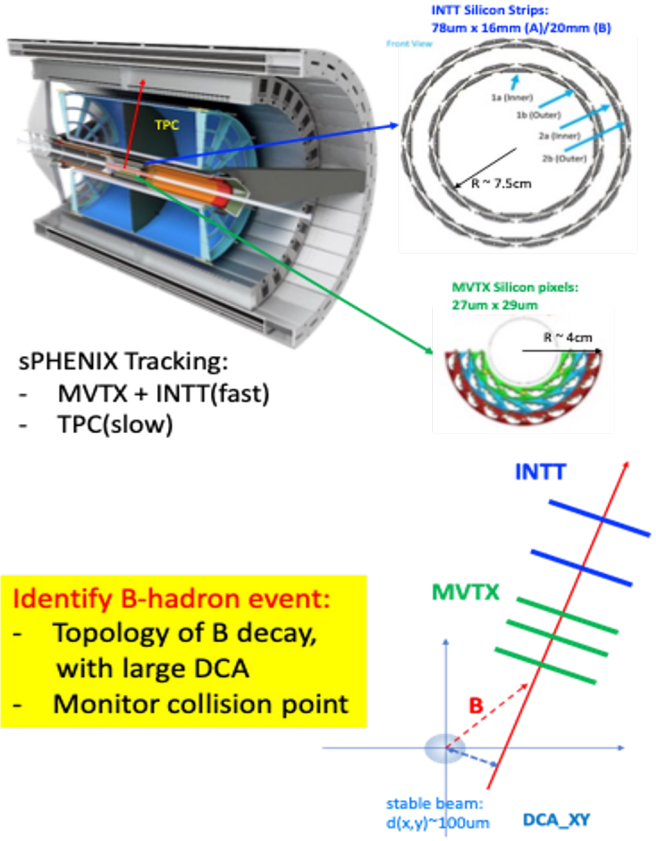
Heavy Quark physics – a key pillar of RHIC science

Project Goals and Deliverables:

- Heavy flavor event AI-trigger demonstrator in p+p

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



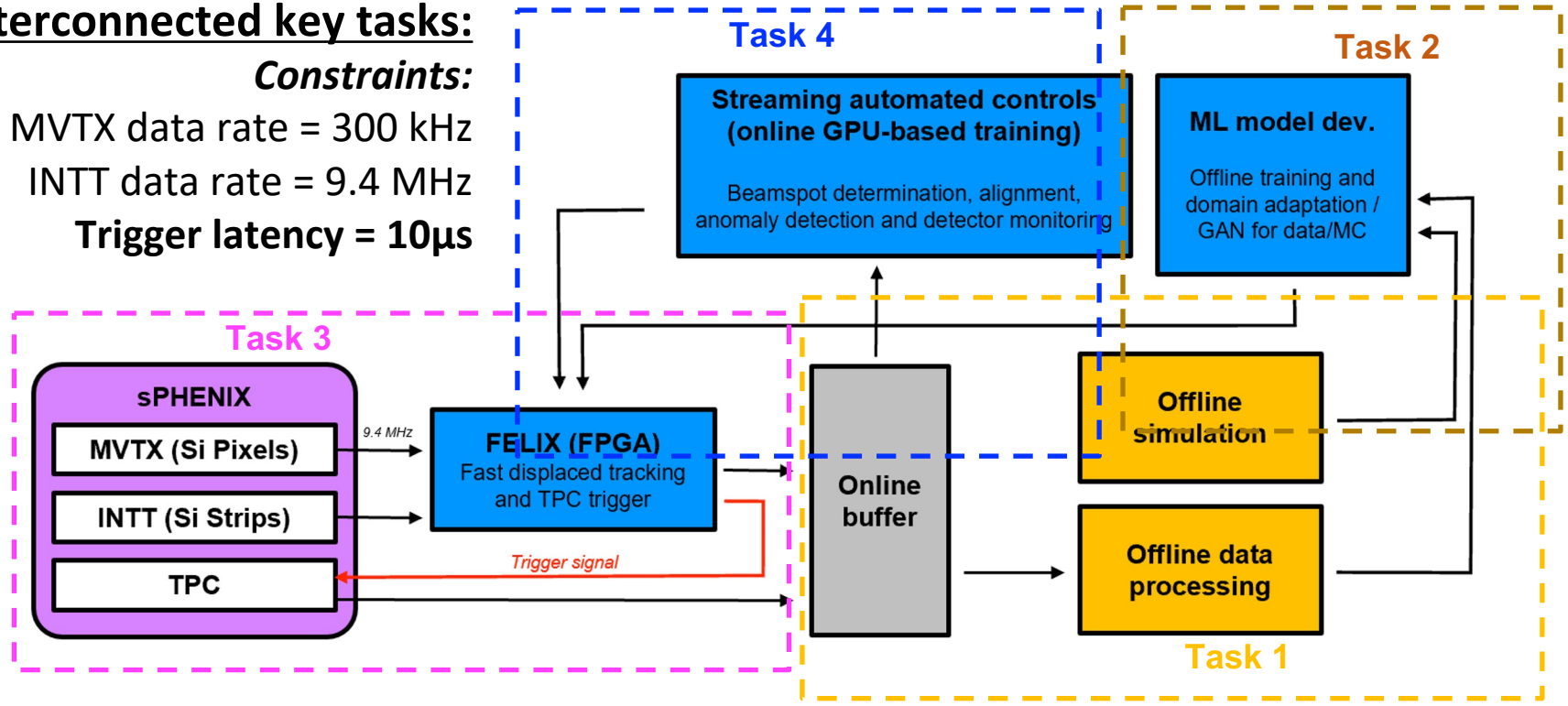
4 interconnected key tasks:

Constraints:

MVTX data rate = 300 kHz

INTT data rate = 9.4 MHz

Trigger latency = 10µs



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)**

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.

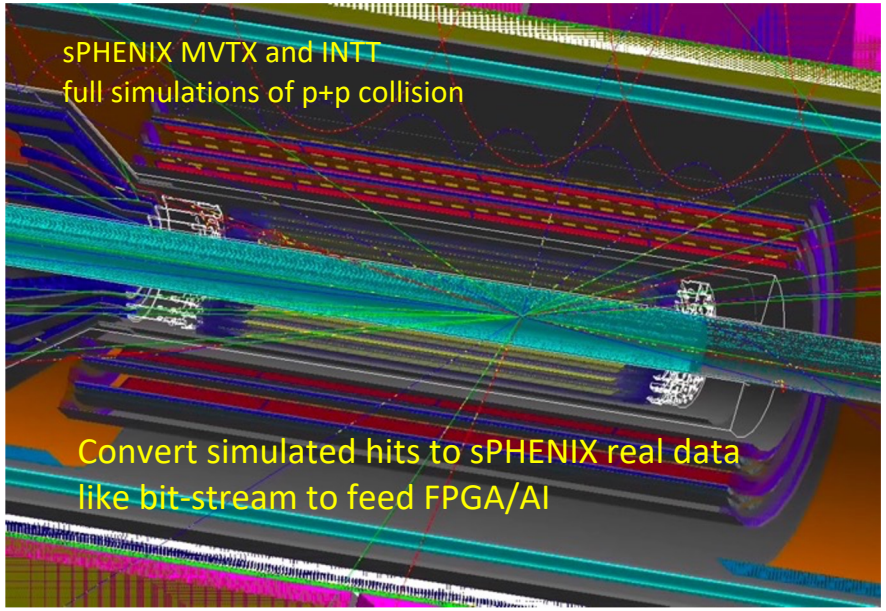
New teams joined later in 2022/2023:

- Dr. Jo Schambach, ORNL, sPHENIX/EIC readout integration, sPHENIX MVTX and EIC/ePIC readout lead
- Dr. Kai Chen, CCNU, FELIX-AI-Trigger hardware integration, FELIX developer at BNL for ATLAS, also sPHENIX
- Prof. Song Fu, NTU, data acceleration in ML
- Prof. Callie Hao, GaTech, AI algorithm/Firmware in ML

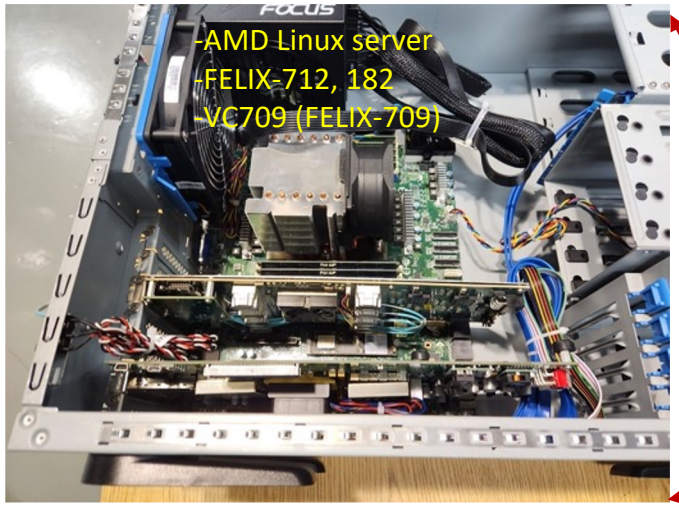
Technical Approaches and Highlights - I

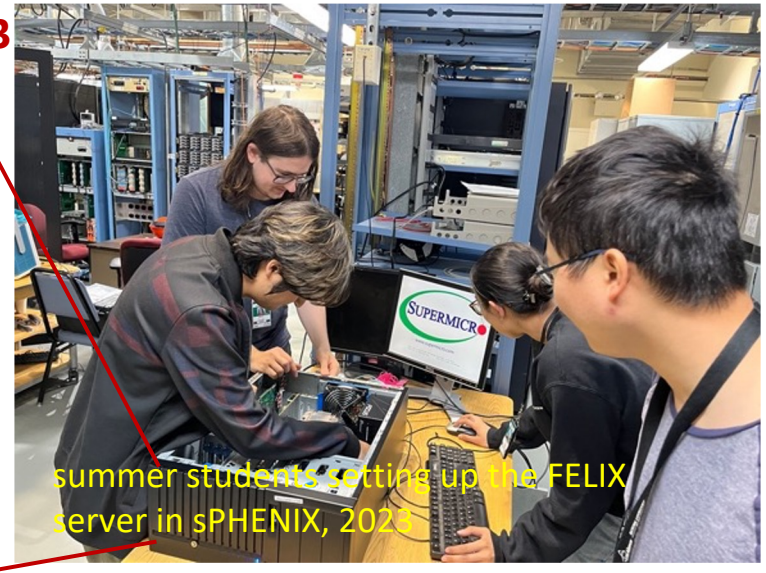
Jakub

- Objective 1 – **Design, build, simulate, and benchmark a prototype streaming readout system with AI-based fast online data processing and autonomous detector control system that meets the physics and engineering requirements.** To support this objective, we first aim to generate a large volume of simulation data for heavy flavor decay events. We plan to design a prototype in the simulated and the real sPHENIX experimental environment and later apply the technology in the high luminosity EIC experiments at RHIC. Our objective is to create a working prototype that serves as a baseline and template for future upgrades. With this prototypical working solution, we target to improve the heavy flavor samples from the current 0.05% yield to more than 10+%. (**Task 1**)



FastML Demonstrator setup at BNL sPHENIX DAQ rack room, summer 2023



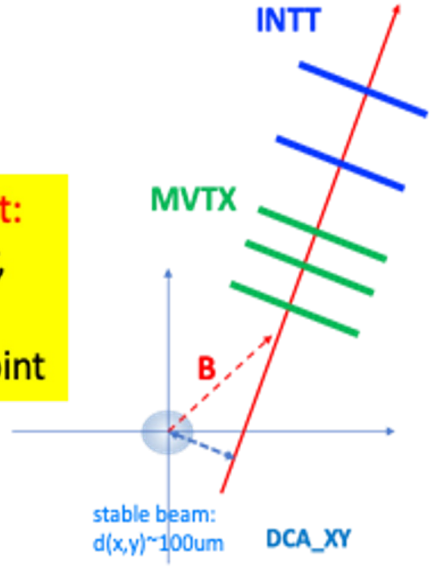


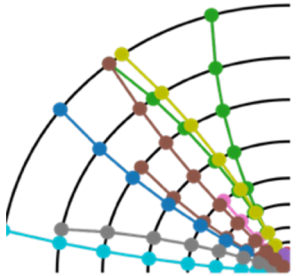
Technical Approaches and Highlights - II

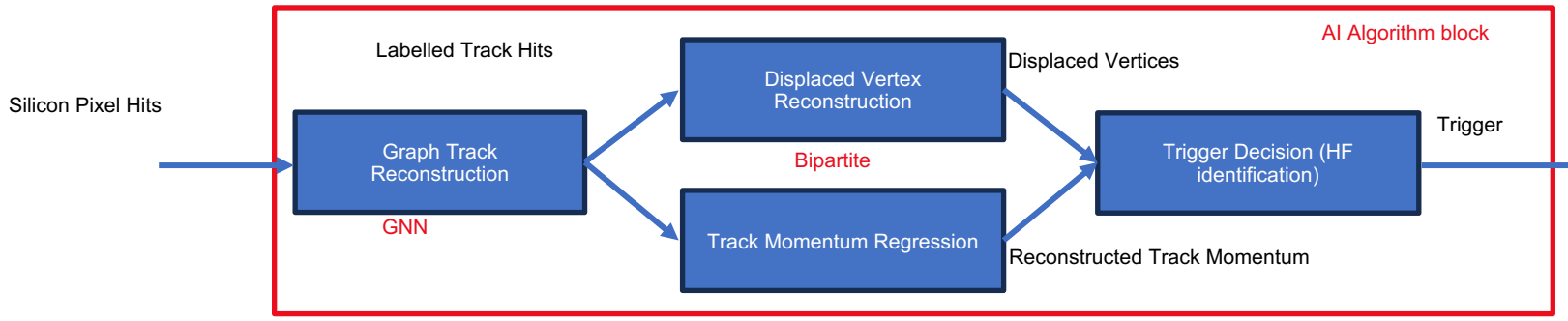
Dantong

- Objective 2 – **Design advanced deep neural networks commensurate with sPHENIX/EIC streaming data requirements.** We aim to design deep neural networks with the following goals: (a) network size: neuron weights that fit in the FPGA block RAMs (BRAMs) of the FELIX cards in sPHENIX/EIC experiments, (b) handling the extremely low signal-to-noise ratio of hit images due to the sparse readout of the high-resolution MVTX and INTT detectors, (c) performance improvements: 10% improvement over state-of-the-art triggering algorithms, and (d) minimal performance gap between simulated data and real experiment readouts, and outstanding generalization capability. (**Task 2**)

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



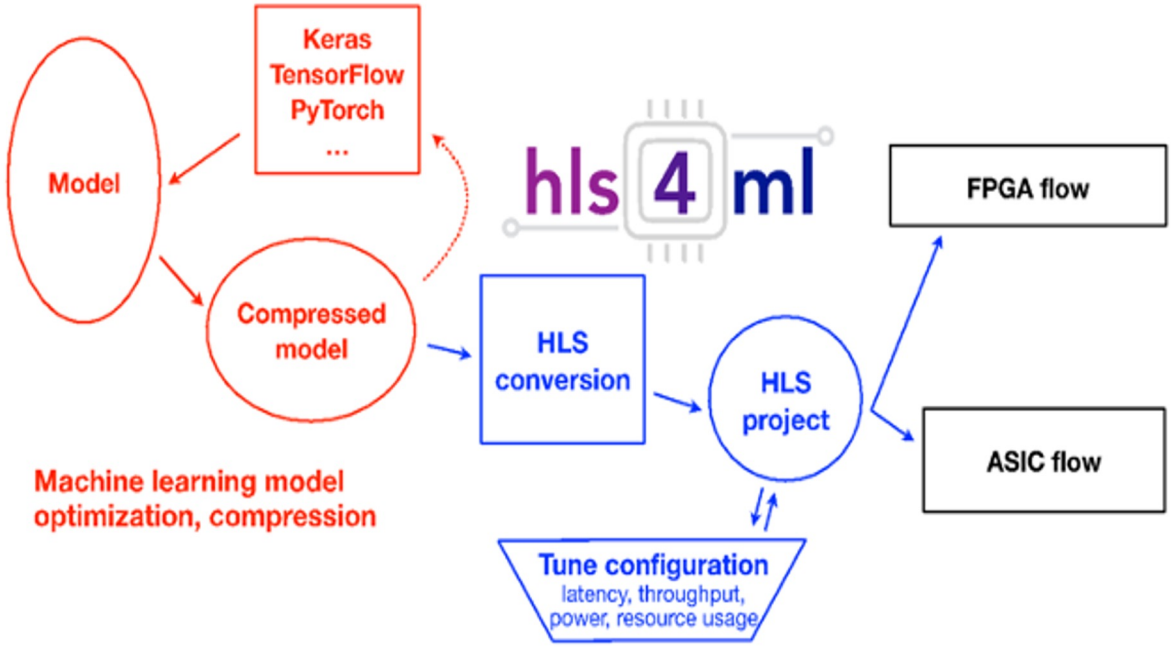




Technical Approaches and Highlights - III

Hannah/Jovan

- Objective 3 – **Deploy advanced deep neural networks within the FELIX system that are capable of real-time reconstruction of heavy flavor events at high throughput.** With the development of advanced deep neural networks, a parallel strategy is needed to ensure that these networks can be designed to operate at low latency and high throughput on the FELIX FPGA cards. This challenge involves detailed AI/hardware co-design to ensure that the desired algorithms can be fit within existing resources, and can achieve full throughput. (**Task 3**)



Also manual model implementation with primary focus:

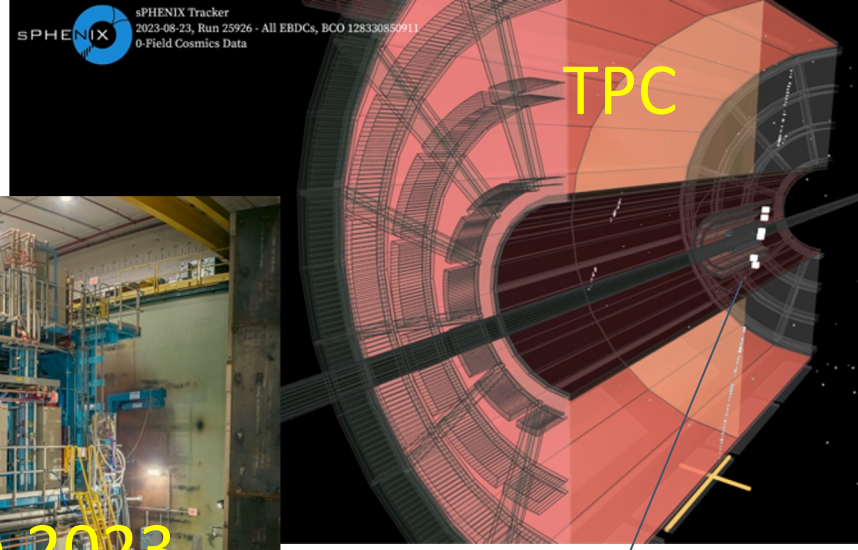
- *achieving low latency*
- *real-time processing of data*
- *deployment of algorithms with high efficiency*

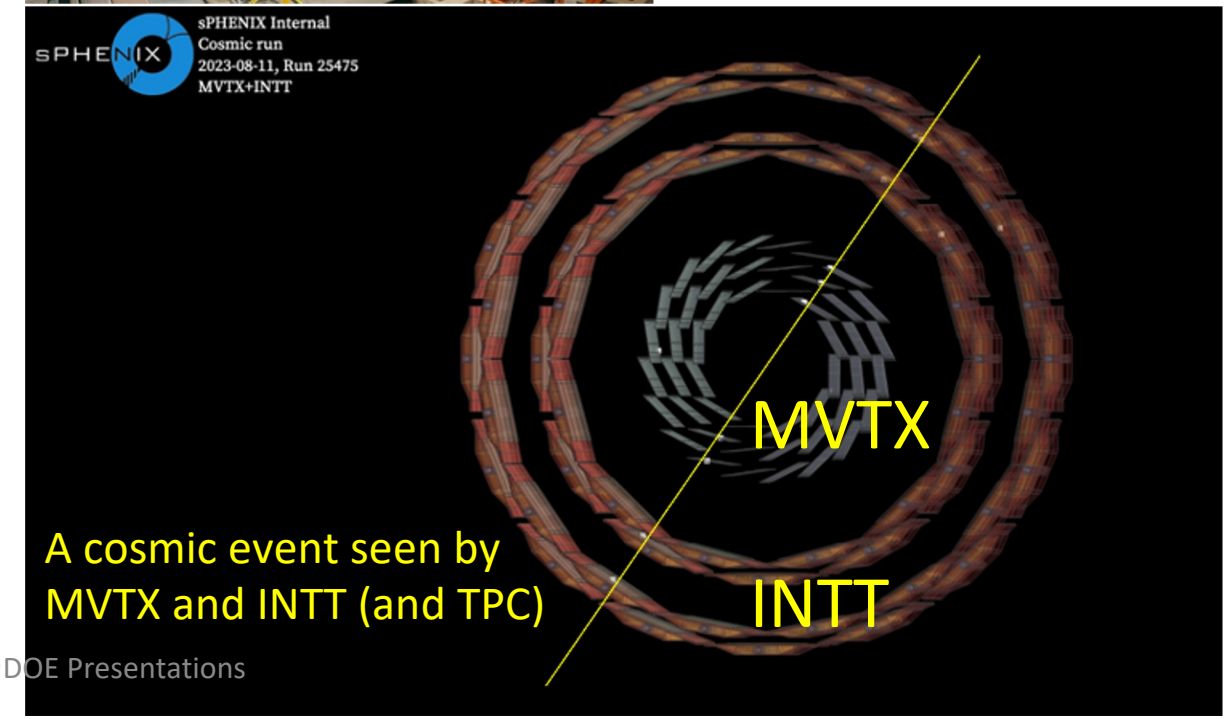
- **Task:** Take predefined algorithm in PyG to implement on FPGA with minimized latency
  - Use manual approach within FPGA and latency constraints (< 10 us)
  - Measured end-to-end on FPGA board (as opposed to simulation)
- **Approach:** Use architecture based on Flow GNN (https://arxiv.org/pdf/2204.13103.pdf)
  - **Synthesizable C++ via High Level Synthesis (HLS)**: translate the Pytorch model into C++ without pointers, recursive, or dynamic memory
  - **Optimized C++ in HLS**: apply hardware optimization techniques to reduce execution latency and resource usage

Successes and Challenges

- **sPHENIX status**
  - installed and commissioned in summer 2023
    - Run23 ended prematurely due to accelerator hardware failure
    - delayed FastML-sPHENIX DAQ integration
  - MVTX installed on time and partially commissioned
    - successfully took cosmic ray data with other sPHENIX detectors (INTT, TPOT, TPC)
    - successfully commissioned SRO
  - INTT installed and partially commissioned
    - successfully took data with other subsystems (MBD, TPOT) in triggered mode
    - successfully demonstrated SRO
  - Run 2024 will start ~March, 2024
    - Au+Au and p+p
    - DAQ and AI-Trigger integration
- **Work in progress and challenges**
  - improve algorithms
    - optimize FPGA resource usage
  - MVTX and INTT SRO integration into FPGA/AI-Trigger
    - sPHENIX DAQ system integration
    - *Beam backgrounds*
  - remote computer/hardware access for non-sPHENIX collaborators
    - demonstrator setup in sPHENIX at BNL
    - setup a standalone at MIT for small testings and development
- **Summary of expenditures**
  - total budget, $1,500K (FY22-FY23), arrived late in 2022;
  - no-cost extension
  - Stage-II project funded, $1,600K (FY24-25), funding received

	FY 20XX	FY 2022	FY 2023
a) Funds allocated		750K	750K
b) Actual costs to date		50K/LANL 55K/MIT 82K/FNAL $44K/NJIT	$410K/LANL $400K/MIT $150K/FNAL $99K/NJIT

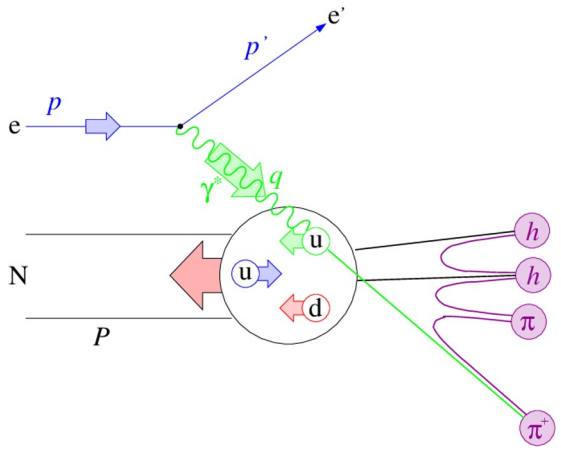




Electron Tagging - DIS Event Identification

Selective streaming readout for AI-Engine:

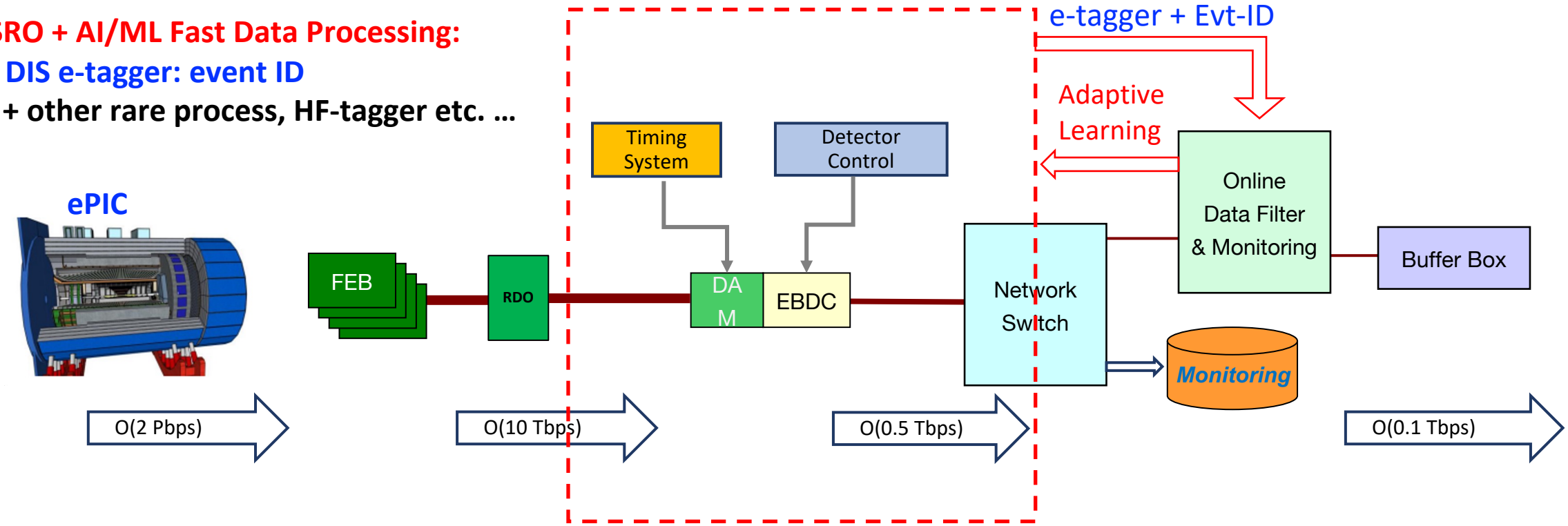
- tag DIS-electron to define DIS event ID
  - EMCal + Trker + PID
- tag other rare must-keep physics signals
  - HF with Trker etc.



SRO + AI/ML Fast Data Processing:

- DIS e-tagger: event ID

+ other rare process, HF-tagger etc. …



Summary and Outlook

- **Produced full sPHENIX physics and detector simulations of heavy quark and QCD backgrounds**
- **Successfully developed preliminary AI-algorithms for sPHENIX HF triggers**
- **sPHENIX installed and commissioned in summer 2023**
  - **Completed MVTX SRO**
  - **Demonstrated INTT SRO**
- **Successfully implemented a toy AI-algorithm in HLS4ML in FELIX**
- **Work in progress to implement full sPHENIX HF AI-trigger in a simplified hardware**

Future plan:

- Implement the demonstrator for sPHENIX p+p run in 2024
- Further develop EIC/ePIC SRO with AI/ML for EIC CD2(2025) and CD3(2025) based on our work
- Funded to continue R&D in FY24-25

Note: completed; in progres

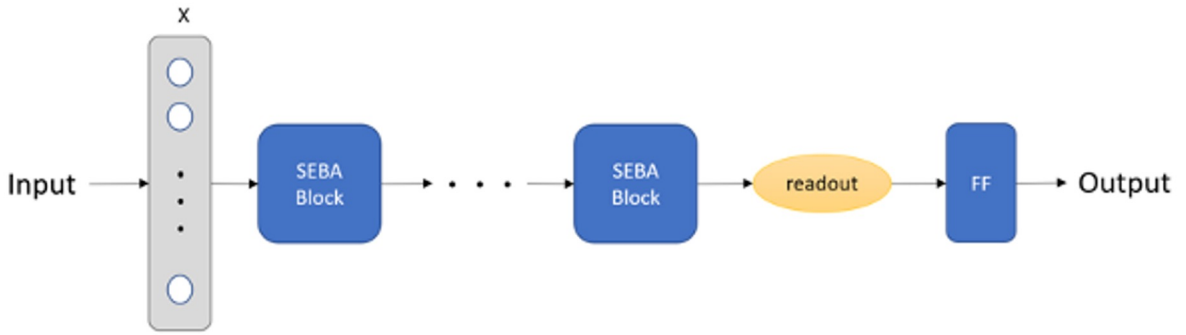
TASK I	TASK II	TASK III	TASK IV
Year 1			
1. Software development and system design. We will first perform detailed sPHENIX physics and detector simulations to design a real-time fast data processing and autonomous detector control and calibration system. In the meantime, we will survey currently available AI models and design a system for offline training and domain adaptation for data and MC. The physics and detector simulation results and the performance of hardware are used to tune the AI algorithms. 2. Hardware development and system integration. We will take advantage of the streaming readout capability of the sPHENIX tracking system to implement continuous readout of two fast silicon tracking subsystems, MVTX and INTT. A FPGA based fast tier-1 AI system will be developed to identify heavy flavor (HF) events in $p+p$ collisions, and generate a fast trigger to initiate the readout of TPC.			
By Q2			
▶ Generate open heavy flavor and QCD background events for simulations (LANL, MIT)	▶ HF trigger algorithm development for FPGA (FNAL, LANL, NJIT)	▶ MVTX streaming readout (LANL) ▶ INTT streaming readout (MIT)	▶ Beamspot interaction and readout simulation (FNAL) ▶ Displaced tracks and anomaly simulation (FNAL, MIT)
By Q3			
▶ Develop fast tracking algorithms using MVTX and INTT hit information (LANL, MIT, NJIT)	▶ Design real-time GPU training machine (MIT, NJIT)	▶ `hls4ml` implementation and algorithm development (FNAL, MIT)	▶ Preliminary design of streaming and automated controls of online GPU-based training system (MIT, NJIT)
By Q4			
▶ Complete a preliminary design of HF trigger AI offline (FNAL, LANL, NJIT)	▶ ML, Graph NN training, by NJIT and MIT	▶ FPGA implementation of HF trigger with MVTX and INTT (All)	▶ Simulation and training (MIT, NJIT)
Year 2			
We will focus on the system integration and continue to improve and benchmark the performance of software, firmware, and hardware system.			
By Q5 & Q6			
▶ Interface between AI system and MVTX detector Data Input by (FNAL, LANL) ▶ Interface between AI system and TPC Readout Control (FNAL, LANL)	▶ Design new GNNs (Encoder, Attention, Particle-Net) algorithms with `hls4ml` (MIT, NJIT)	▶ `hls4ml` customization for FELIX board (FNAL, LANL) ▶ FPGA, GPU system integration and evaluation (MIT, NJIT)	▶ GPU deployment for autoencoder and training (FNAL, MIT)
By Q7			
▶ Continue to improve algorithms for HF tagging (LANL)	▶ Improve algorithm with `hls4ml` on FPGA (FNAL, NJIT)	▶ Multi-FELIX Board Integration (LANL) ▶ Validation and test with FELIX boards (All)	▶ ML model and domain adaption update (MIT, NJIT)
By Q8			
▶ Benchmark system performance with sPHENIX or test beam data (All)			

Physics simulation and AI-ML algorithms

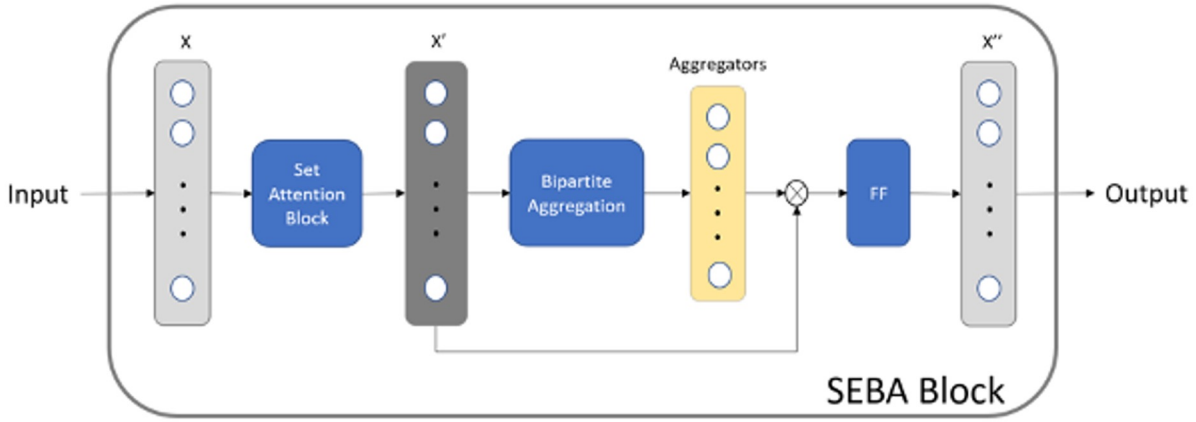
- Dantong Yu and Giorgian Borca-Tasciuc (NJIT)

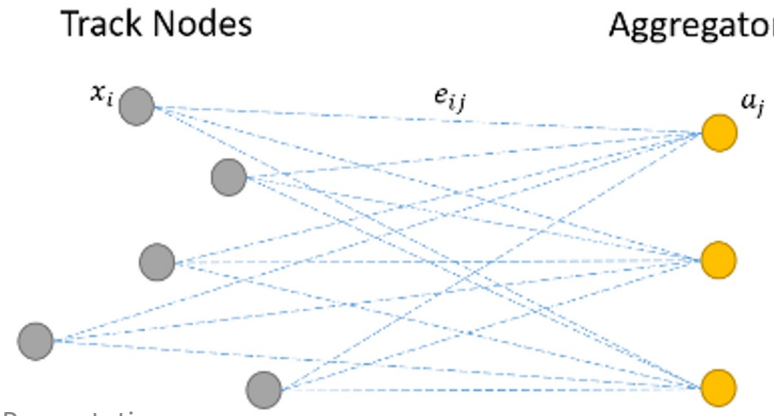
Progress

- Robustness and Algorithm Accuracy Improvement
- Latency Improvements in Tracking Algorithm
- Track-Based Trigger Prediction Algorithm
- Hits-Based Trigger Prediction Algorithm
- Pileup Handling
- Robustness Verification



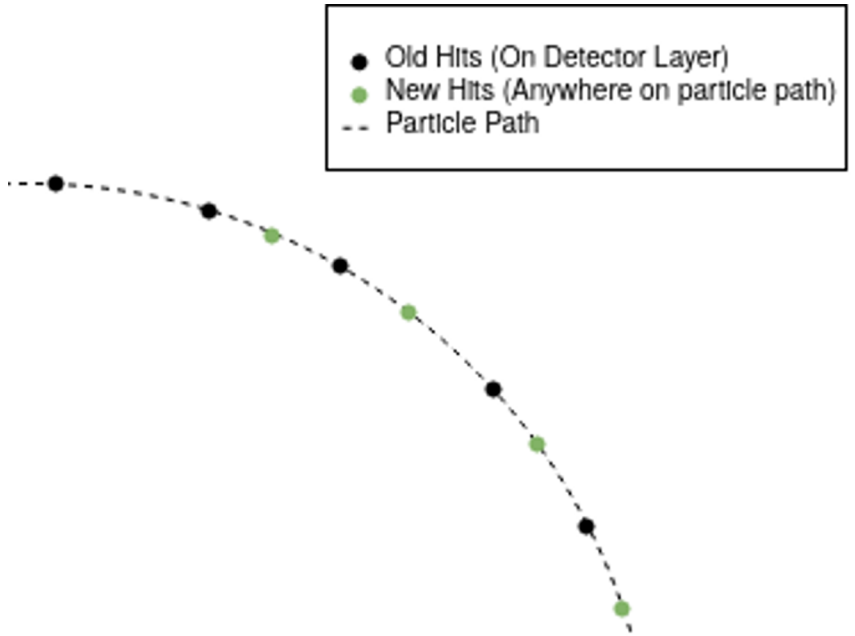
Set Encoder with Bipartite Aggregator (SEBA) Blocks





Robustness: Data Augmentation

- Hits belonging to a track are perturbed to different points on the particle's trajectory
- Model learns more general classification function based on trajectory instead of fixed layer locations
- Helps improve both robustness and final accuracy
- Robust to detector alignment.



Robustness:
- Multi-Task Learning in Trigger Prediction

- Multi-Task Learning: Track embeddings used to predict whether two tracks come from the same parent particle
  - Additional adjacency-matrix component added to the loss function:
  - $\mathcal{L}$ = LCE($\text{trigger}_{\text{pred}}$, $\text{trigger}_{\text{true}}$) + LCE($A_{\text{pred}}$, A_{true})

- Regularize the model with additional physical structural information about the event.

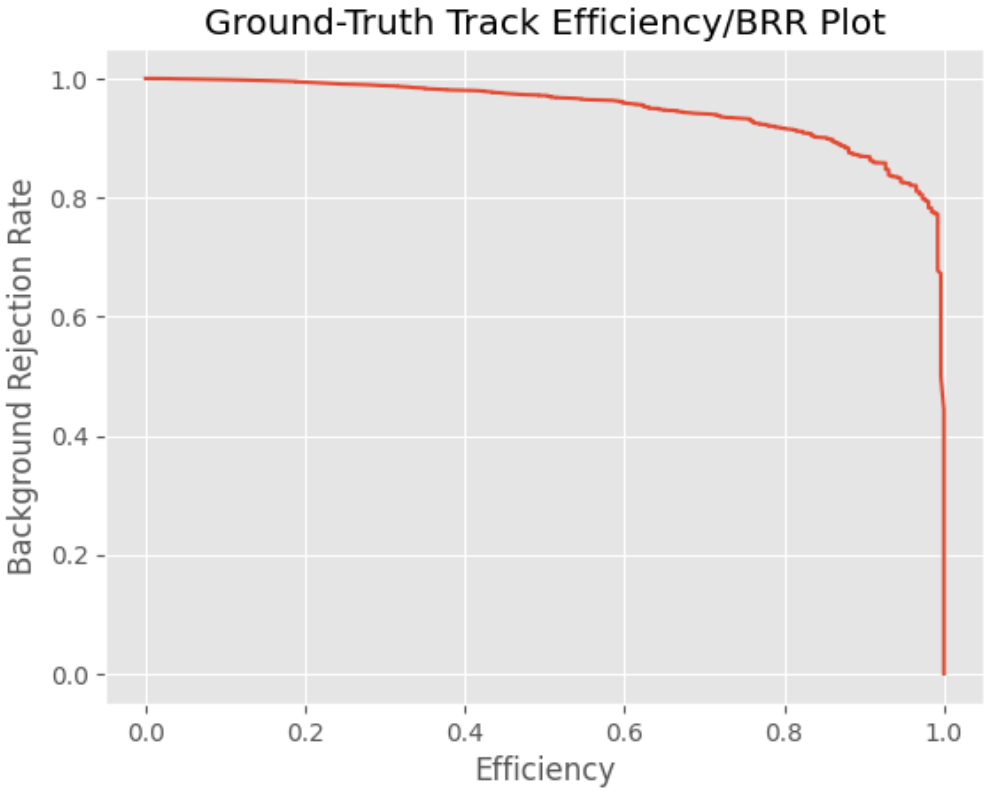
Algorithm Accuracy Improvements

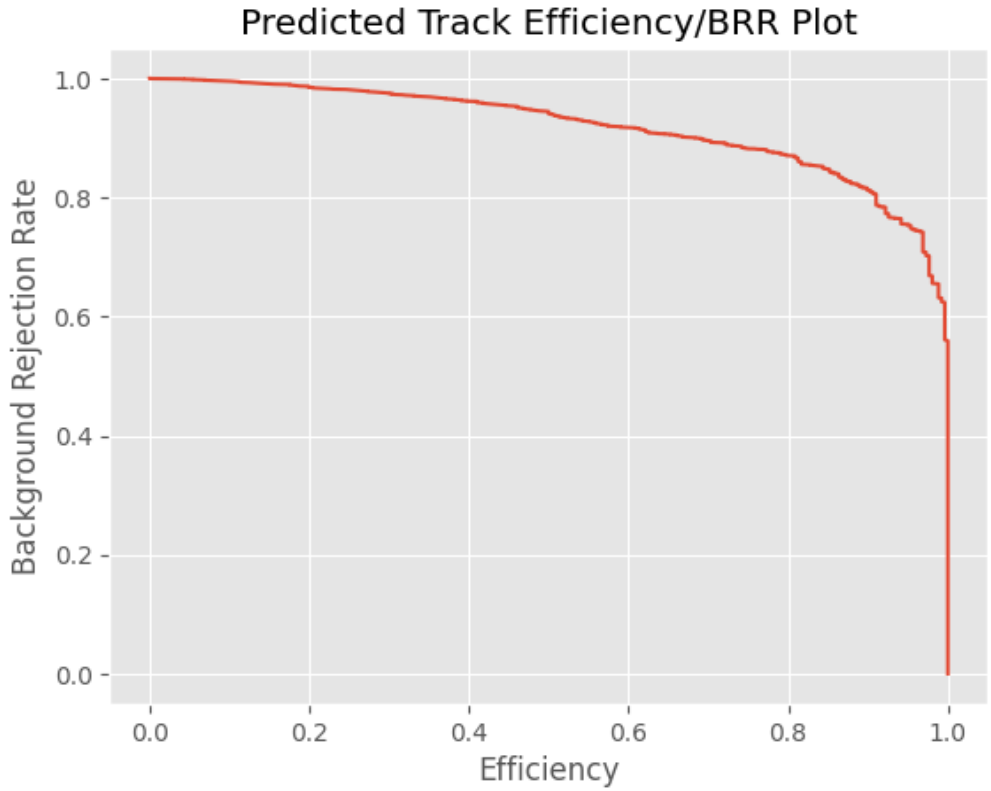
- Applying previous techniques, accuracy for both ground truth and predicted tracks (fully end-to-end) improved. Improved Model (in Bold) v.s. the old model in italic without data augmentation.
  - +3.7% for GT Tracks
  - +2.5% for predicted tracks

Data	Year	Accuracy	Efficiency	Purity	BRR
GT Tracks	**2023**	**90.2%**	**96.1%**	**6.20%**	**85.4%**
Predicted Tracks (also includes tightened constraints)	**2023**	**86.5%**	**92.6%**	**4.62%**	**80.7%**
Predicted Tracks	2022	84.0%	-	-	
GT Tracks	2022	87.5%	-	-	

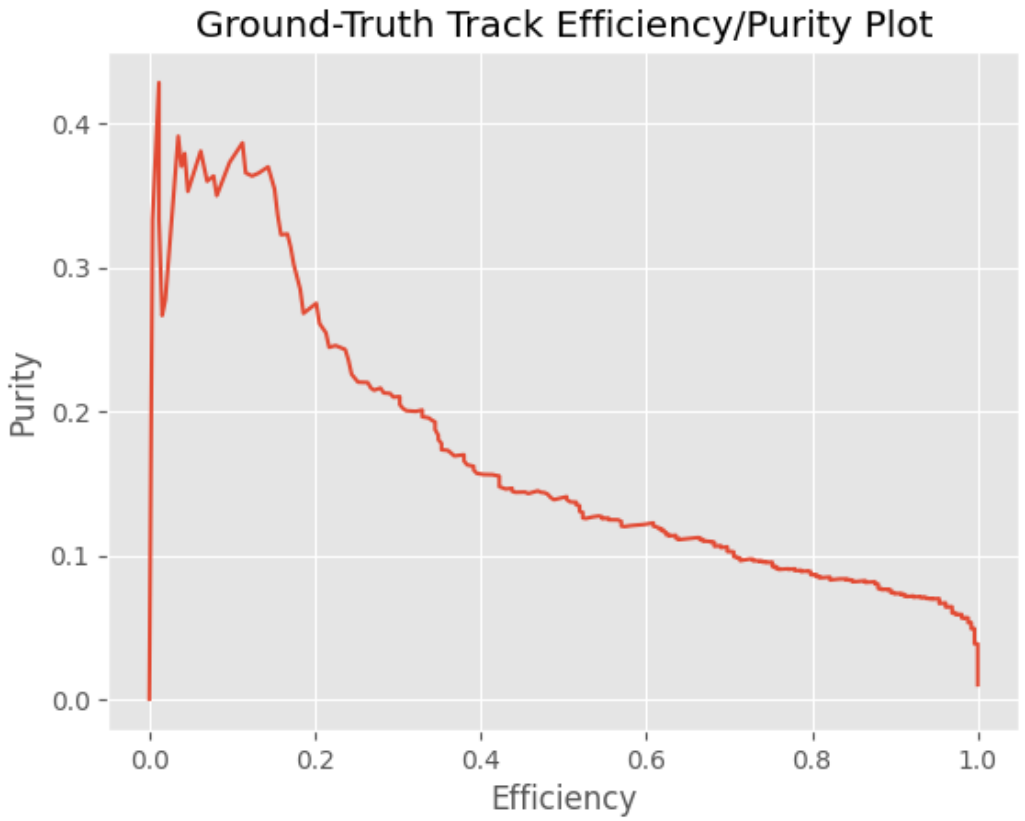
Note: predicted probability cutoff of 0.5 is used for efficiency/purity, and accuracy calculations

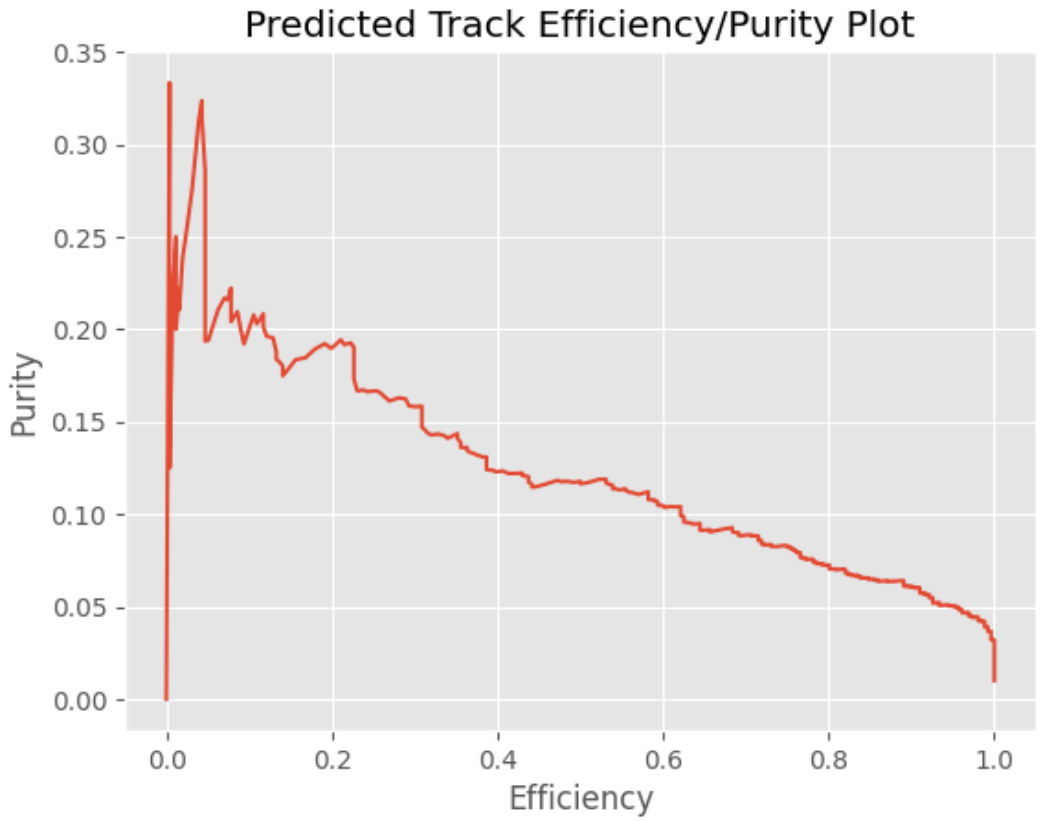
Efficiency/BRR with ROC





Efficiency/Purity Under 1% Signal Rate





Latency Improvements in Tracking Algorithm

- Geometric constraints are used to determine edge candidates for tracking algorithm
- Latency-Accuracy tradeoff exists:
  - Permissive geometric constraints allow more true edges to be captured in the edge candidate set→Improves Accuracy
  - Restrictive geometric constraints reduces the number of edge candidates → Improves Latency
- Run experiments to determine effect of geometric constraints on final **trigger** performance, assuming a perfect tracking algorithm
- We can reduce the maximum edge candidates by ~50% with little accuracy penalty

	$d\phi_{max}$	dz_{max}	accuracy	Maximum Edge Candidates
0	0.025005	102.000000	0.885895	1030.0
1	0.014881	16.000000	0.885360	548.0
2	0.011599	155.000000	0.884555	638.0
3	0.026555	113.000000	0.884320	1077.0
4	0.024582	178.000000	0.883860	1022.0
5	0.010320	48.000000	0.882630	556.0
6	0.012193	14.220353	0.881850	463.0
7	0.030000	200.000000	NaN	1171.0

Hits-Based Trigger Prediction Algorithm

- Co-design means trade-off between accuracy and latency. If we skip the tracking and directly predict trigger from hit graph, we will significantly reduce the latency and cost on FPGA acceleration.
- Implemented End-To-End trigger pipeline that removes intermediate tracking step
- Some performance loss, but large improvement in parameter count and latency

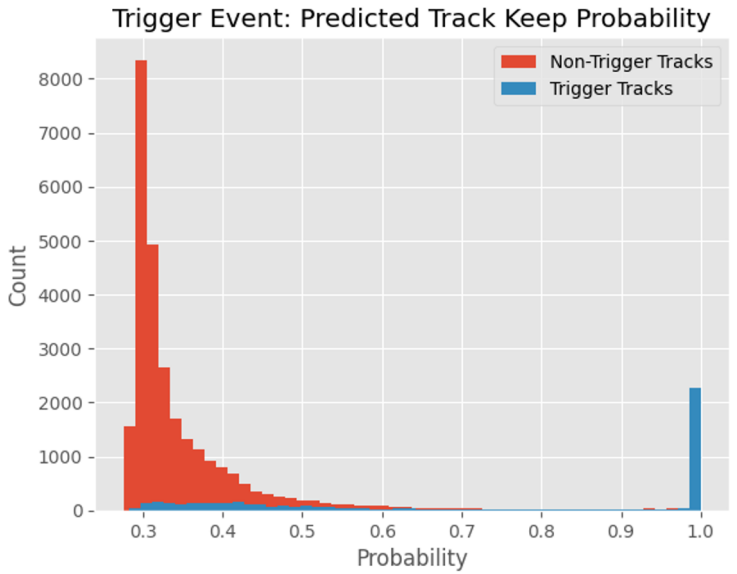
Input Type	**Parameter Count**	**Accuracy**
Single-Event Hits	776	72.58%

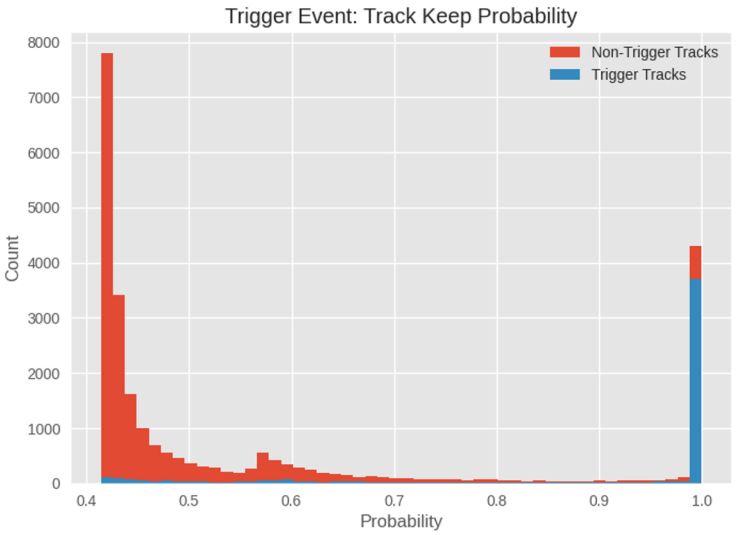
Pileup Handling

- As-implemented, tracking and trigger algorithm need to deal with pile-up
- retrain the tracking algorithm with the hits of 20 MVTX events and 1 INTT events. Target is the tracks of the single event within the INTT.
- Tracking algorithm takes the 20 events and produces the tracks from a single event
- Use 20-event pile-up, determine end-to-end accuracy
- We achieve an end-to-end accuracy of **78%**. Less than a <10% reduction in accuracy despite a large increase the complexity of the problem

Robustness Verification with Explainable AI

- Ensure our trigger models are making decisions on a physically-sound basis and thus will work outside the simulation
- Use Bernoulli LRI technique (bLRI) to probe which tracks the model is using to make decisions
  - Model is penalized for not dropping tracks, thus it will only keep tracks important to the final decisions
- For both end-to-end and ground-truth tracking models, the model chooses to drop non-trigger tracks and keep trigger tracks: physically sound!





hls4ml translation and firmware implementation

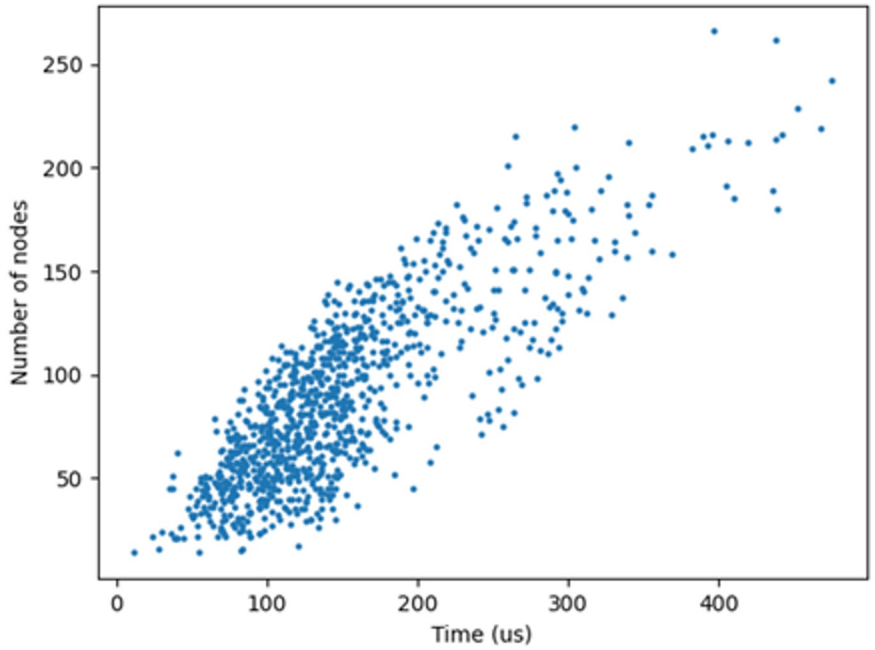
- Hannah Bossi (MIT) and Jovan Mitrevski (FNAL)

Model Implementation on FPGA

- **Task:** Take TrackGNN (implemented with PyG) to put on FPGA with minimized latency
  - Use approach within FPGA and latency constraints (< 10 us)
  - Measured end-to-end on FPGA board (as opposed to simulation)
- **Approach # 1:** Manual approach for model instance based on FlowGNN (https://arxiv.org/pdf/2204.13103.pdf)
  - **Synthesizable C++ via High Level Synthesis (HLS)**: translate the Pytorch model into C++ without pointers, recursive, or dynamic memory
  - **Optimized C++ in HLS**: apply hardware optimization techniques to reduce execution latency and resource usage
- **Approach #2:** Automated firmware generation with hls4ml
  - Python package for machine learning inference on FPGAs
  - Recent progress to implement GNNs in hls4ml

Detailed Solutions – Implementation 1

- **TrackGNN model**
  - 5 layers; node/edge embedding MLP: 64 dimensions, 4 layers
- **Implementation Results**
  - Tested with: 100 nodes, 140 edges
  - 150 us per graph (Freq. 130 MHz)
  - 130 us per graph (Freq. 180 MHz)
- **Utilization (Alveo U280)**
  - LUT: 308K (23.7%), FF: 378K (14.5%)
  - BRAM: 1,025 (50.8%), DSP: 1,426 (15.8%)



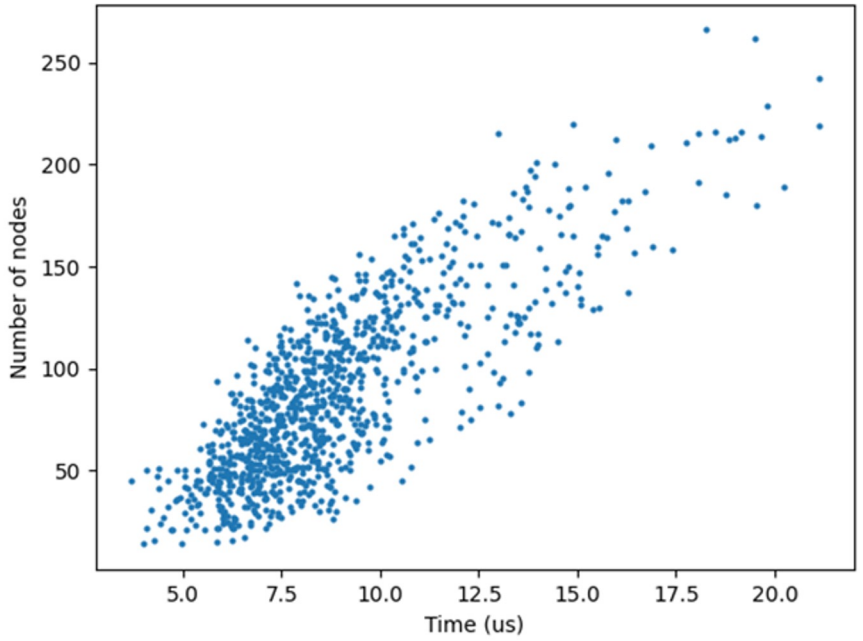
Detailed Solutions – Implementation 1

- **TrackGNN model**
  - 5 layers; node/edge embedding MLP: 64 dimension, 4 layers
- **Implementation Results**
  - Tested with: 100 nodes, 140 edges
  - 150 us per graph (Freq. 130 MHz)
  - **130** us per graph (Freq. **180** MHz)
- **Utilization (Alveo U280)**
  - LUT: 308K (23.7%), FF: 378K (14.5%)
  - BRAM: 1,025 (50.8%), DSP: 1,426 (15.8%)

★ **Too slow!! Target is < 10 us**
★ **Hardware has almost reached its maximum capability**
★ **Needs to redesign the algorithm**

Detailed Solutions – Implementation 2

- **Simplified TrackGNN model by software team**
  - 1 layers; node/edge embedding MLP: 8 dimension, 4 layers
- **Implementation Results**
  - Tested with: 92 nodes, 142 edges
  - 8.82 us per graph (Freq. 285 MHz)
  - **14.7x speedup!**
- **Utilization (Alveo U280)**
  - LUT: 194K (14.9%), FF: 214K (8.2%)
  - DSP: 488 (5.4%), BRAM: 406 (20.2%)



Detailed Solutions – Implementation 2

- **Simplified TrackGNN model by software team**
  - 1 layers; node/edge embedding MLP: 8 dimension, 4 layers
- **Implementation Results**
  - Tested with: 92 nodes, 142 edges
  - 8.82 us per graph (Freq. 285 MHz)
  - **14.7x speedup!**
- **Utilization (Alveo U280)**
  - LUT: 194K (14.9%), FF: 214K (8.2%)
  - DSP: 488 (5.4%), BRAM: 406 (20.2%)

★ **Smaller model improves both latency (14.7x) and clock frequency (1.58x)**
★ **More aggressive quantization is expected to be helpful**

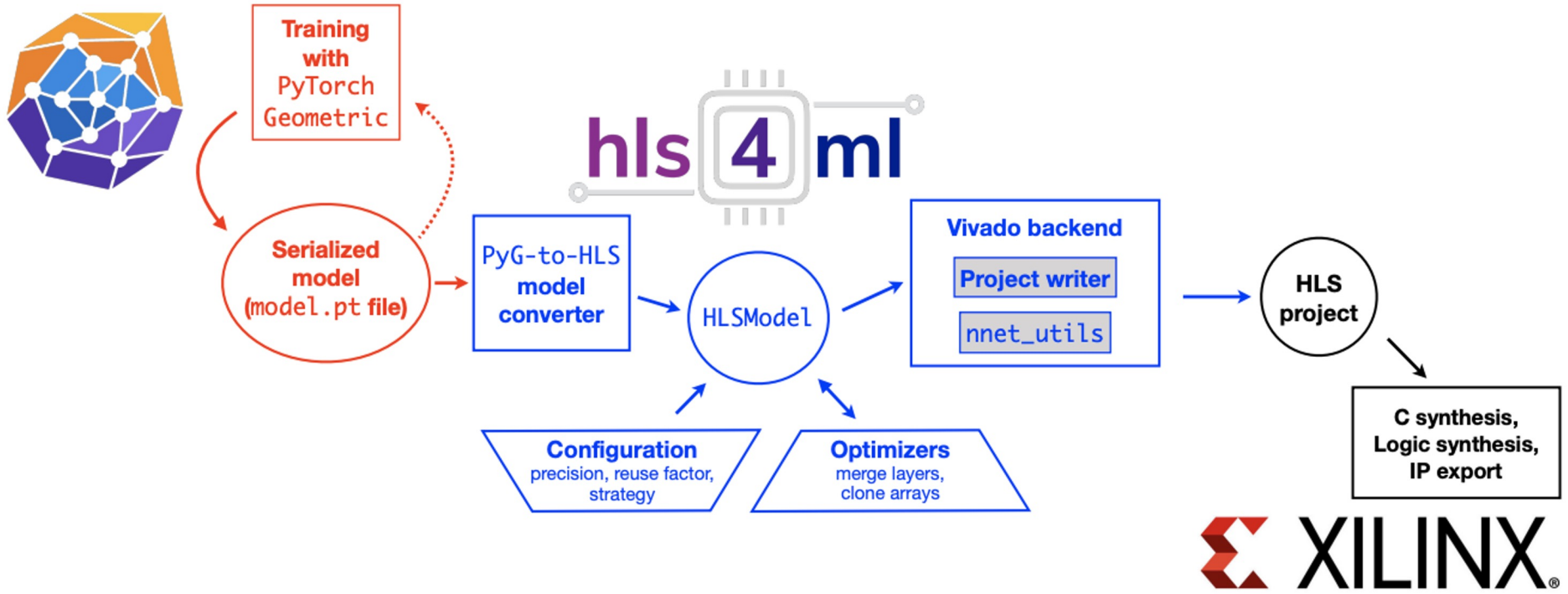
Detailed Solutions – Tracking + Triggering

- **Trigger model is needed to be included, following TrackGNN**
  - The latency limit is still 10 us but TrackGNN itself is already 8.82 us
- **Algorithm innovation: combined tracking and triggering**
  - Modified TrackGNN + graph level aggregation for triggering
- **Implementation Results**
  - Tested with: 92 nodes, 142 edges
  - **9.2** us per graph (Freq. **180** MHz) → **still within 10 us!**
- **Utilization (Alveo U280)**
  - LUT: 241K (18.5%), FF: 236K (9.04%), DSP: 969 (10.7%), BRAM: 594 (29.5%)

Firmware Implementation with hls4ml at a Glance

https://fastmachinelearning.org/hls4ml/

https://arxiv.org/pdf/2112.02048.pdf



hls4ml is a compiler taking AI models in TF/Keras, PyTorch, or ONNX, producing HLS for deployment on FPGAs and ASICs.

hls4ml for Real-time AI Deployment

- Origin: to deploy neural networks in the Level-1 trigger on the CMS Experiment at the Large Hadron Collider (LHC) at CERN
- Widely used open source software
  - GitHub: https://github.com/fastmachinelearning/hls4ml
  - 950+ stars, 350+ forks, 45+ contributors, almost 30k PyPI downloads
- Experienced hls4ml developers are part of this effort, including Nhan Tran, Philip Harris, Vladimir Lončar, Jovan Mitrevski
  - Can extend the software to suit the needs of the project

GNN Implementation in hls4ml

- Graphs are natural representations for physics purposes - shows data points (nodes) and the relation between them (edges).
  - Ex: charged particle tracking
- To fully realize GNNs for tracking, need implementation on FPGA due to strict latency constraints

- Common software used is pytorch geometric (PyG) - first step is to implement prototype for PyG model
- ***Already some progress towards this end!***
- See slides from London workshop for more details

PyTorch

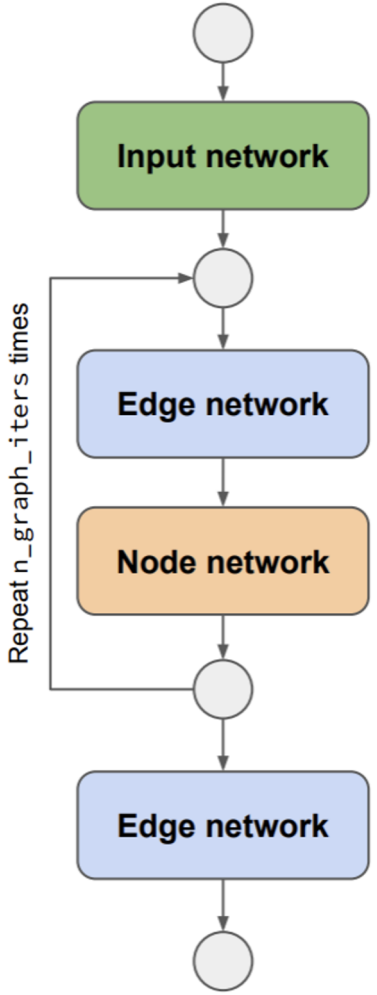
```
class GNNSegmentClassifier(nn.Module):
    def __init__(self, input_dim=3, hidden_dim=8, n_graph_iters=1,
                 hidden_activation='relu', layer_norm=False):
        super(GNNSegmentClassifier, self).__init__()
        self.n_graph_iters = n_graph_iters
        self.input_network = make_mlp(input_dim, [hidden_dim],
                                      output_activation=hidden_activation,
                                      layer_norm=layer_norm)
        self.edge_network = EdgeNetwork(hidden_dim, hidden_dim,
                                        hidden_activation, layer_norm=layer_norm)
        self.node_network = NodeNetwork(hidden_dim, hidden_dim,
                                        hidden_activation, layer_norm=layer_norm)

    def forward(self, inputs):
        x, edge_index = inputs
        x = self.input_network(x)
        for i in range(self.n_graph_iters):
            x0 = x
            e = torch.sigmoid(self.edge_network(x, edge_index))
            x = self.node_network(x, e, edge_index)
            x = x + x0

        return self.edge_network(x, edge_index)
```

Preliminary Results

- **Preliminary benchmark!**
- Network inputs: 90 nodes, 140 edges (same as simplified TrackGNN)
- Input network:
  - Latency ~ 0.2 us
- Edge network
  - Latency ~ 0.7 us
- Node network
  - fine estimation not yet done
  - Latency ~ 4 us
- Total latency estimation
  - latency = input_network + layers * (edge_network + node_network + ~overhead) + edge_network = 0.2 + 1*(0.7+4.0 + ~0.1) + 0.7 ~ **5.7 us (approximately)**

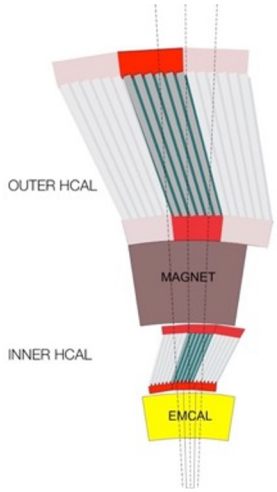


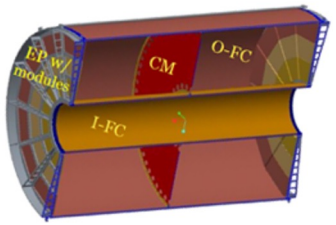
Repeat for each layer

Demonstrator Implementation

- Jakub Kvapil (LANL)

sPHENIX Readout and AI-ML HF Trigger Integration





On Detector **Rack Room**

Front-End Module/Electronics

Data Collection Module
- Zero suppress, packing

SubEvent Buffer (x20)
- Data collector

Calorimeters

FEM
FEM
FEM

DCM2
DCM2
DCM2

SEB
SEB
SEB

Tracking detectors

FEE
FEE
FEE

FELIX EBDC
FELIX EBDC
FELIX EBDC

100+ Gigabit Crossbar Switch

~ 1 PB each

Buffer Box
Buffer Box
Buffer Box
Buffer Box
Buffer Box
Buffer Box

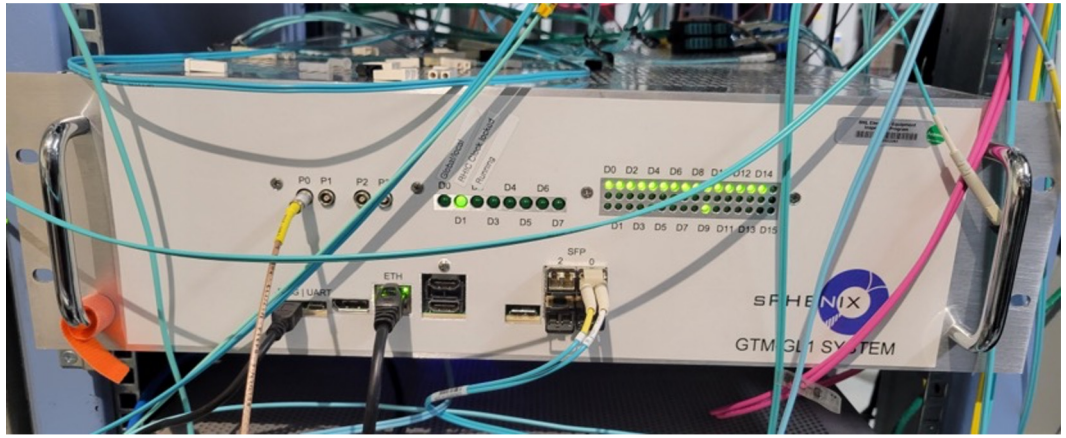
To Computing Centre

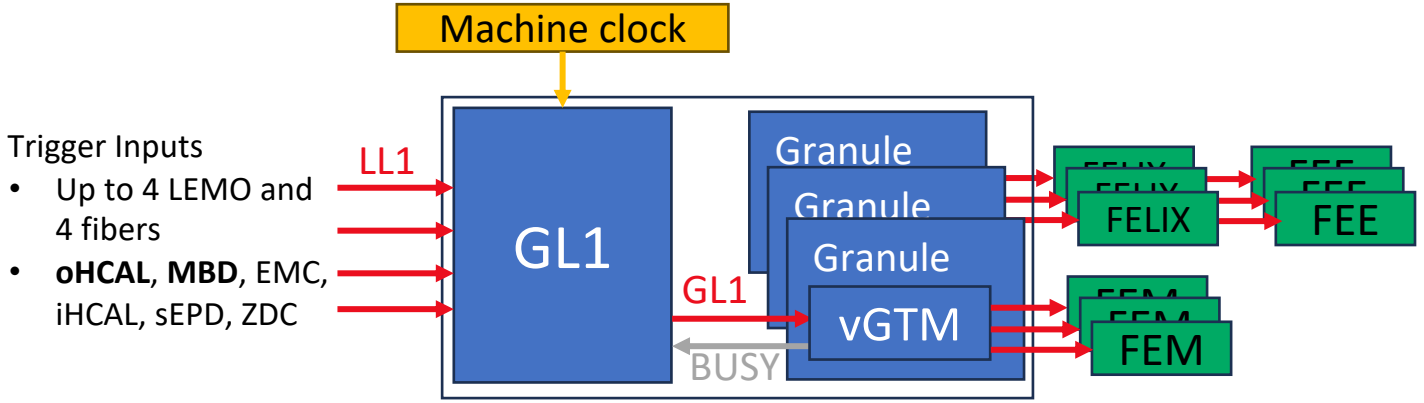
FELIX (MVTX+INTT) -> AI/ML -> Trigger

EBDC: Event Buffer and Data Compressor (x40)
- 6x MVTX, 8x INTT, 24x TPC

The timing and trigger distribution

- The Global Level 1 Trigger (GL1) and the accelerator clock is distributed via Granule Timing Module (GTM)
  - GL1 trigger is used by calorimeters and the TPC
  - GL1 transmits clock and trigger to the vGTM, which then transmits it to the FEE
    - vGTM is the adapter to a given detector
    - GL1 is maintaining the BUSY received from vGTM





The Latency Constraints for the Trigger Delivery

- The TPC buffers can hold up to 30 μs of data
  - The goal of this project is to aim for 10 μs collision-trigger latency to capture the TPC stream
- The Calorimeter buffers can hold up to 6.4 μs of data
  - Can we improve the latency down to 5 μs to also capture the calorimeter stream?

- **The latency breakdown**
  1. MVTX readout window 5 μs – not fixed interaction-readout latency!
  2. IR -> Counting house ~ 0.3 μs (81 m fibres)
  3. FELIX -> AI data forward, decoder buffers, clustering ~ 0.6 μs (@240 MHz)
  4. Tracking + Trigger decision (currently 8.82 μs for TrackGNN model!)
  5. AI -> GTM -> TPC FELIX (negligible, all three sits in Counting house)

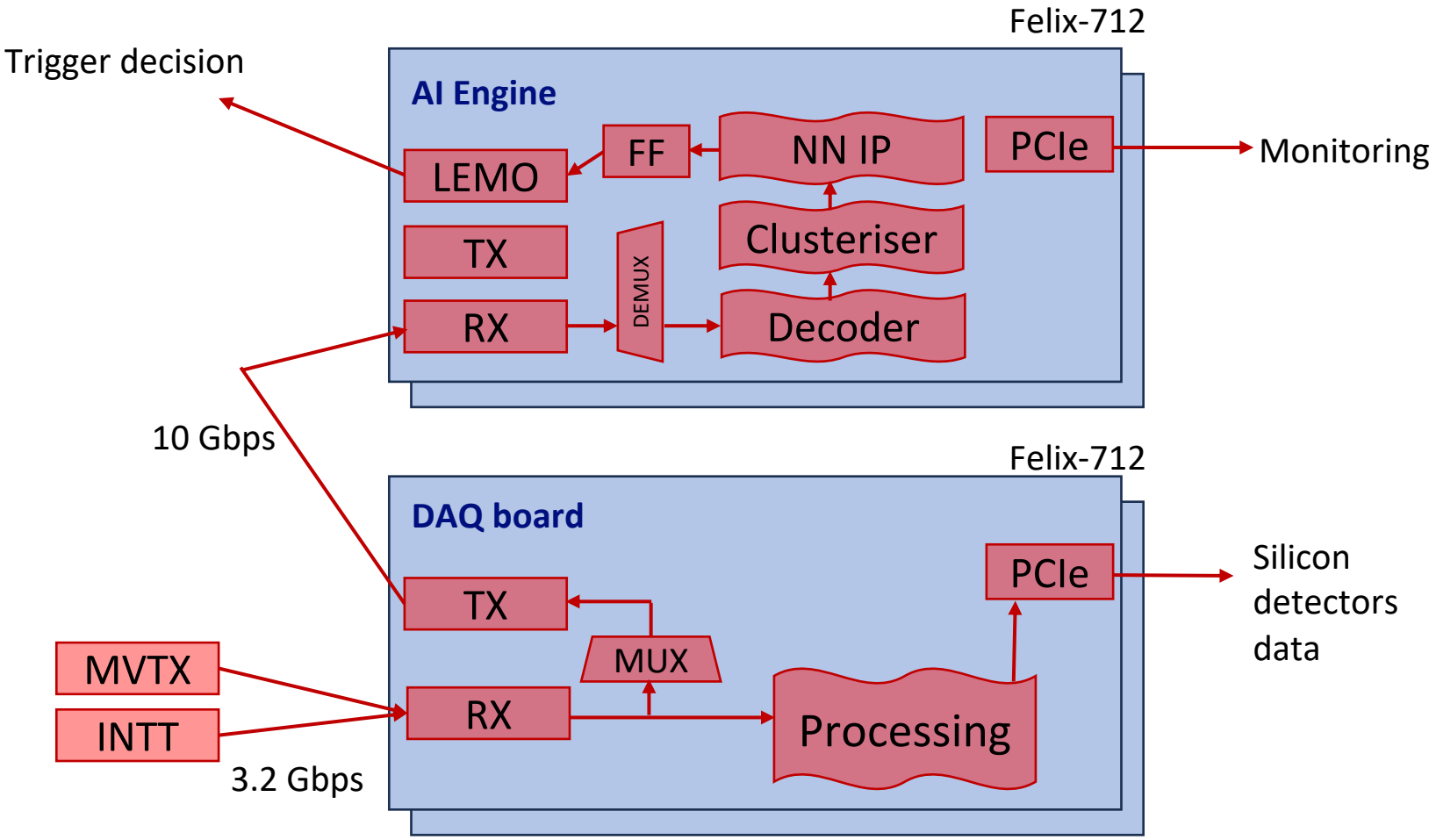
The DAQ–AI Data Flow

- Motivation to use FELIX board:
  - Use the sPHENIX FELIX infrastructure provided to the sPHENIX tracking detectors
  - To reuse the PCIe implementation (16-lane Gen-3) and software tools provided by the FELIX developers
  - Large amount of optical IO, on-board FPGA is a Kintex Ultrascale XCKU115FLVF1924-2E (half the size of Alveo)
- The decision signal of heavy flavor event from the AI-Engine will be sent out via the LEMO connectors to the sPHENIX GTM/GL1 system to initiate the TPC readout in the triggered mode

- MVTX Readout FELIX Firmware is the best starting point -> could even lighten it more by removing MVTX data processing

Post-Implementation	LUT (663K)	FF (1.3M)	BRAM (2K)
MVTX Readout FELIX	87K (13.1%)	196K (14.8%)	879 (40%)

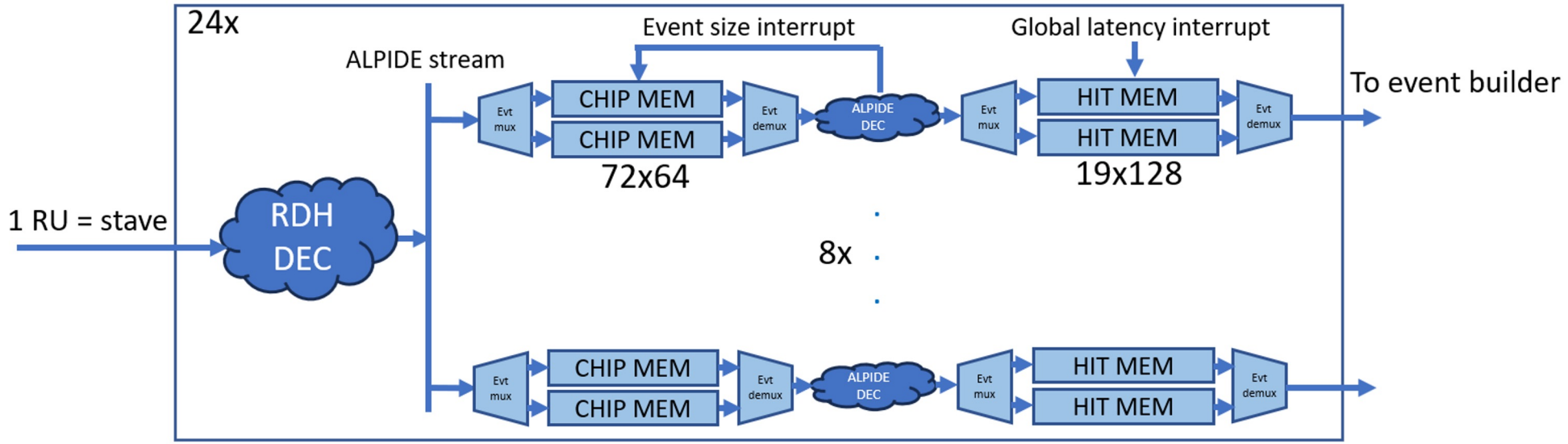
The Firmware Design - Data Flow



- MVTX 144 links @ 3.2 Gbps and INTT raw data stream will feed two AI engines (one for each hemisphere)
  - 24 links for MVTX and 24 links for INTT per AI engine
  - 8b10b protocol with links driven @ 10Gbps
    - tested up to 14 Gbps, with external loopback measurement at FELIX with BER < 10^(-16)
- Raw MVTX and INTT data packets:
  - 1 MVTX packet @5 us strobe
    - ~10 pp collisions (MB events) @2MHz pp collisions
  - 50 INTT packets @ 100 ns strobe
    - Need to run GraphGNN 50 times!
- Data needs to be decoded, clustered, time aligned and feed the neural network IP

Very challenging project to fit in the FPGA resources!

MVTX Decoder

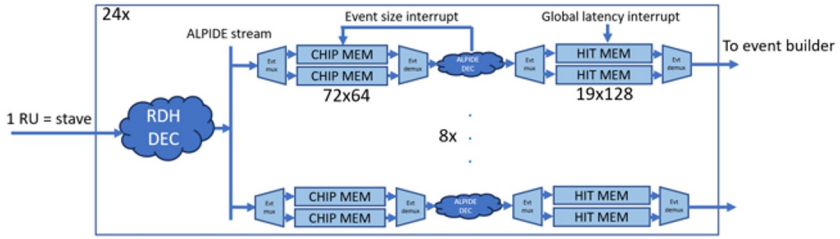


- Initial implementation of the FPGA-based MVTX decoder
- Max 128 hits per chip stored (expected physics ~50, issues with beam background?)
  - Maximum latency 532 ns @ 240 MHz (given by the buffer depth)
- The MVTX data latency depends on the actual collision time and hit occupancy
  - To provide a fixed latency to the GTM a BC information from INTT is used
  - An interrupts to event size/processing time are in place not so exceed the maximum latency
    - Separate memory per MVTX event to fast clear the data

MVTX Decoder - Implementation

- First implementation based on simulation and synthesis data

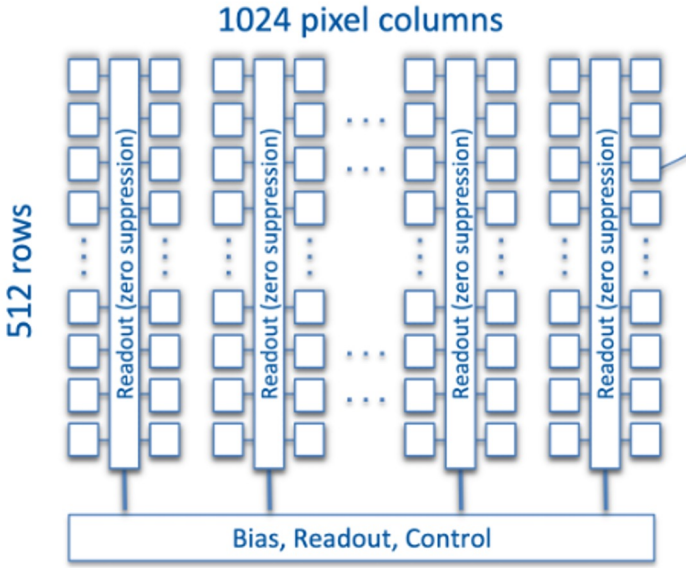
	LUT (663K)	FF (1.3M)	BRAM (2K)	Needed
Readout decoder (RDH DEC)	431	701	0	per stave (x24)
Chip Memory	37	34	1	per chip (x216), per event (x2)
Chip decoder (ALPIDE DEC)	677	252	0	per chip (x216)
Hit (pixel) memory	39	38	0.5	per chip (x216), per event (x2)
Total	189K (28.5%)	102K (7.7%)	648 (30%)	

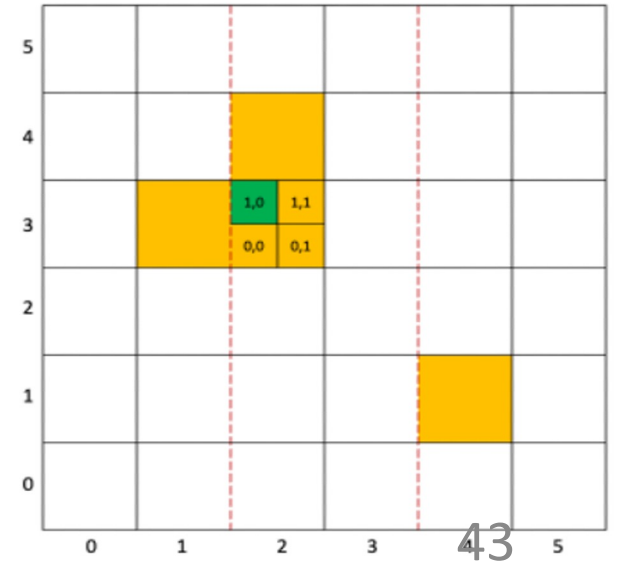


- **Latency**
  - RDH decoder is data distributor = only 1 clock latency (4.16 ns @ 240 MHz)
  - CHIP decoder has read-to-pixel latency of 6 cycles (25 ns @ 240 MHz), total latency depends on occupancy -> 2-3 cycles per pixel. It is expected to have around 50 pixels = 0.5 us
- **Utilization**
  - A bit high, we still need to fit in the FELIX infrastructure, INTT decoder, and model
  - The above design is fully parallelised, could reuse decoders and memory at the cost of latency

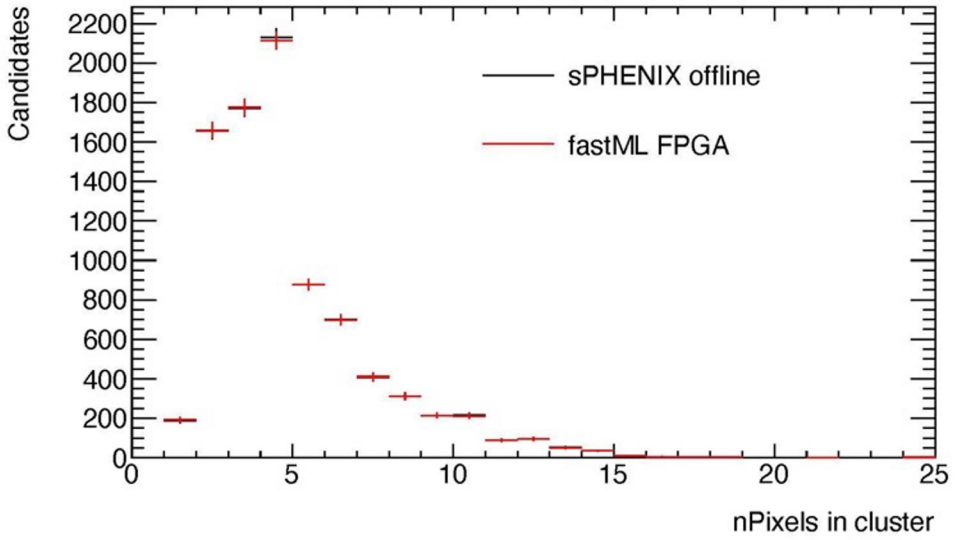
MVTX Clusterizer

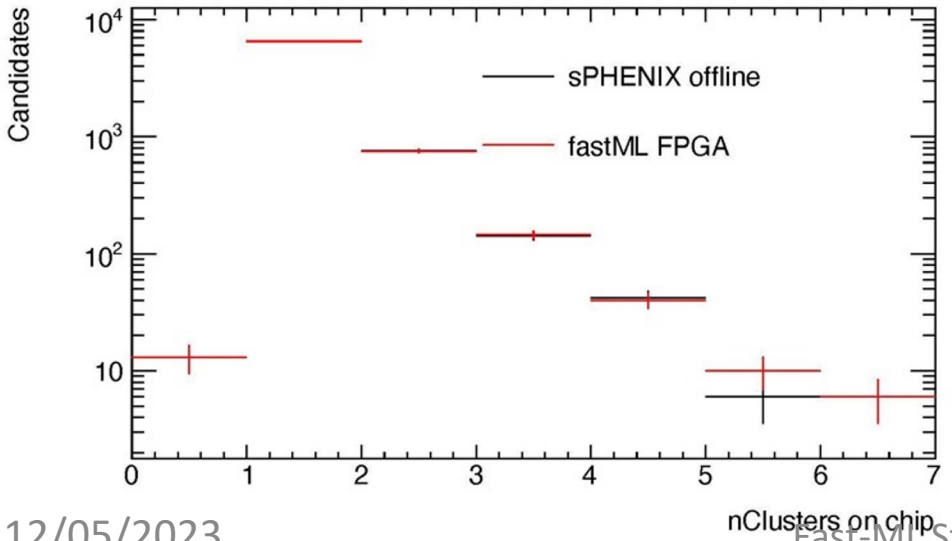
- ALPIDE reads data out in double columns from 0 to 1023
  - Decoded hits thus arrive double column-by-double column
- Clusters can be assembled as they arrive
  - No hits in the next columns three adjacent pixels means cluster is ready to be sent out
- After finding pixel with centroid, pixel can be divided into grids to improve resolution using only 2 more bits
- Can get 13.5 μm cluster resolution at the global level from 31 bits
  - 6 bits to define layer and sensor number
  - 4 bits to define chip number on the sensor
  - 21 bits for cluster position on chip (9 for row, 10 for column, 2 for quadrant)
- After changing to global cluster position, detector layout has become abstracted



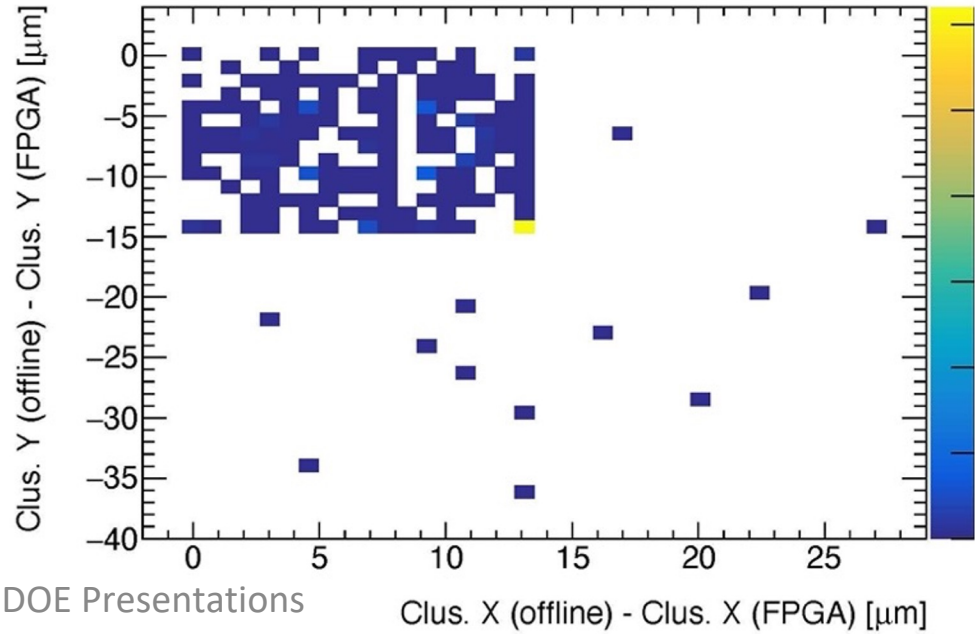


MVTX Clusterizer - WIP Results



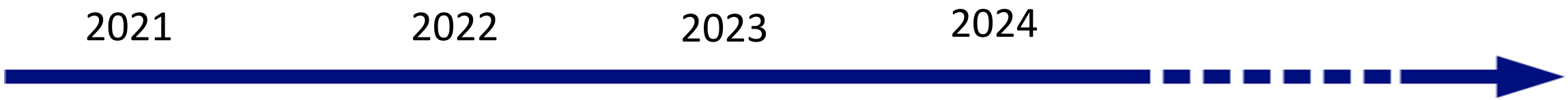


- Preliminary results show good agreement between sPHENIX algorithm and our mock FPGA code.
- Good validation that algorithm logic is working, next step is to translate to VHDL
  - vitis hls can be used for this!



Outlook and Timeline

- Assemble each components together and iterate until FPGA resources and timing is met
- Integrate MVTX and INTT SRO
  - MVTX will be re-installed at sPHENIX in February
  - First beam expected in March - time to test



- Project funded by DOE FOA
- Initial simulations constructed
- First data for algorithm training

- Initial FPGA bitstream synthesis

- Refine interface between system and detectors

We are here!

- Deploy device at sPHENIX pp/pA run
- EIC preliminary TDR (CD2)

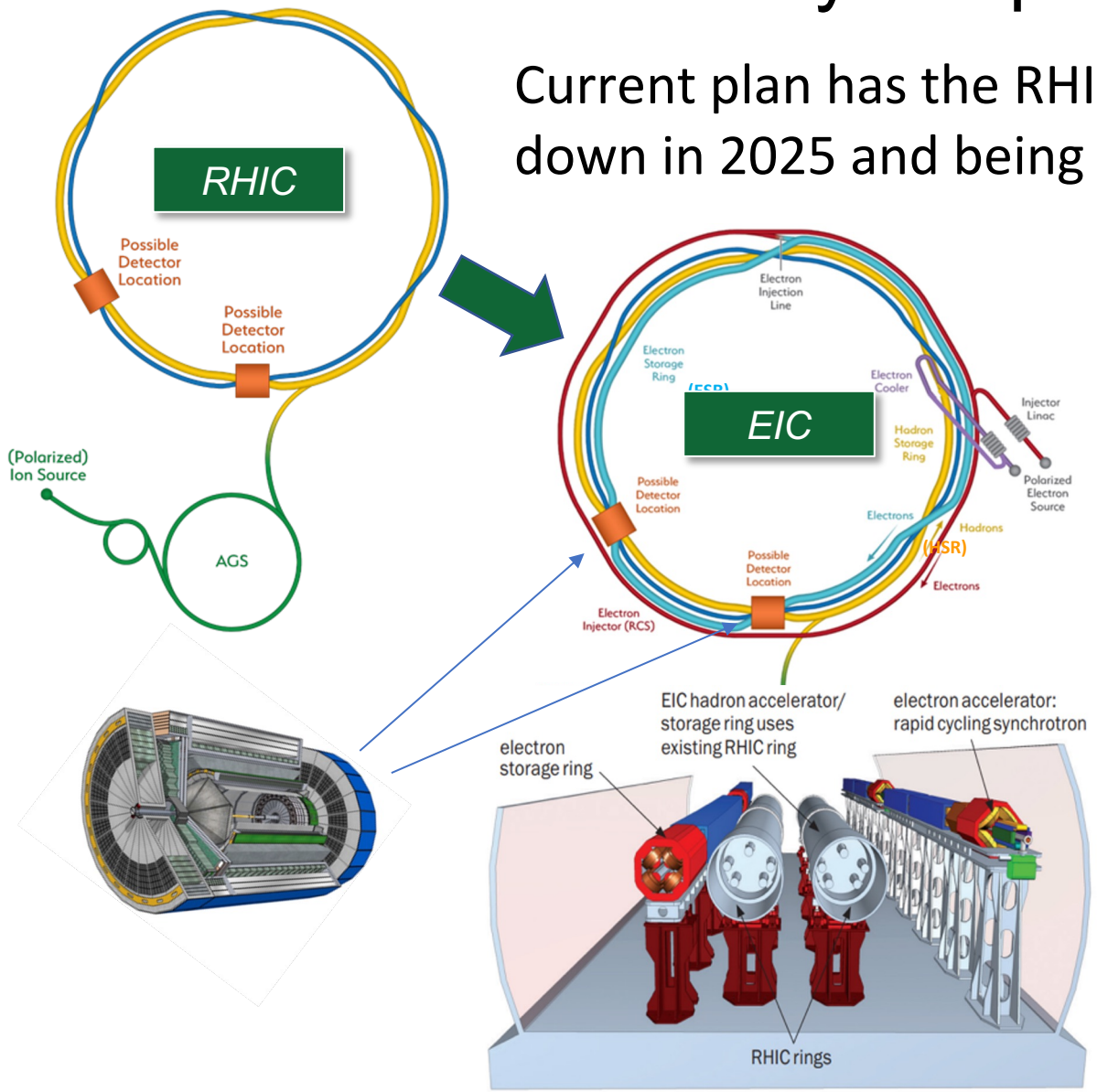
- Final design for EIC TDR (CD3)
- Take advantage of new technology if required

- Deploy device at EIC

Backup Slides

EIC Preliminary Scope Overview

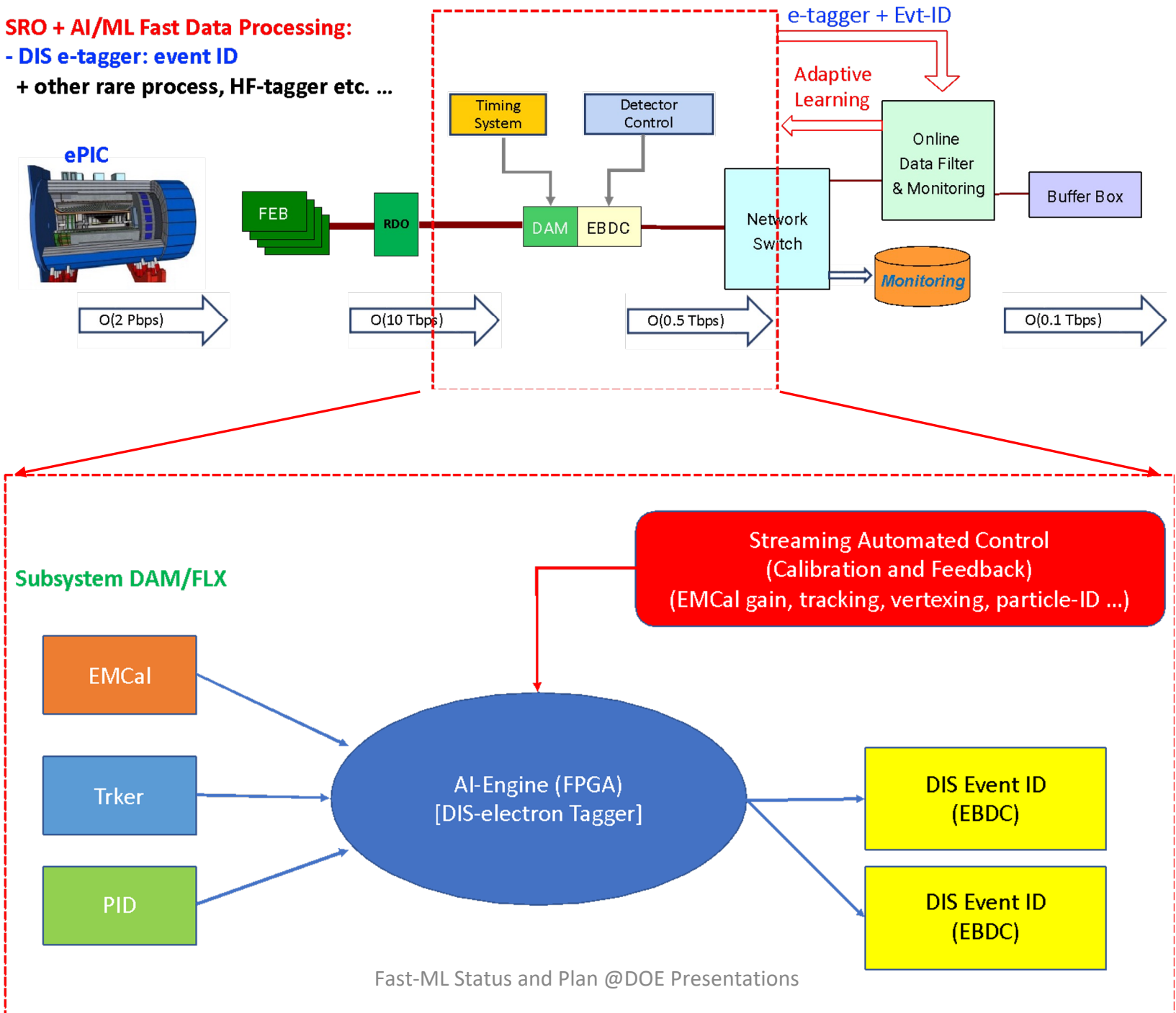
Current plan has the RHIC facility shutting down in 2025 and being modified for the EIC.

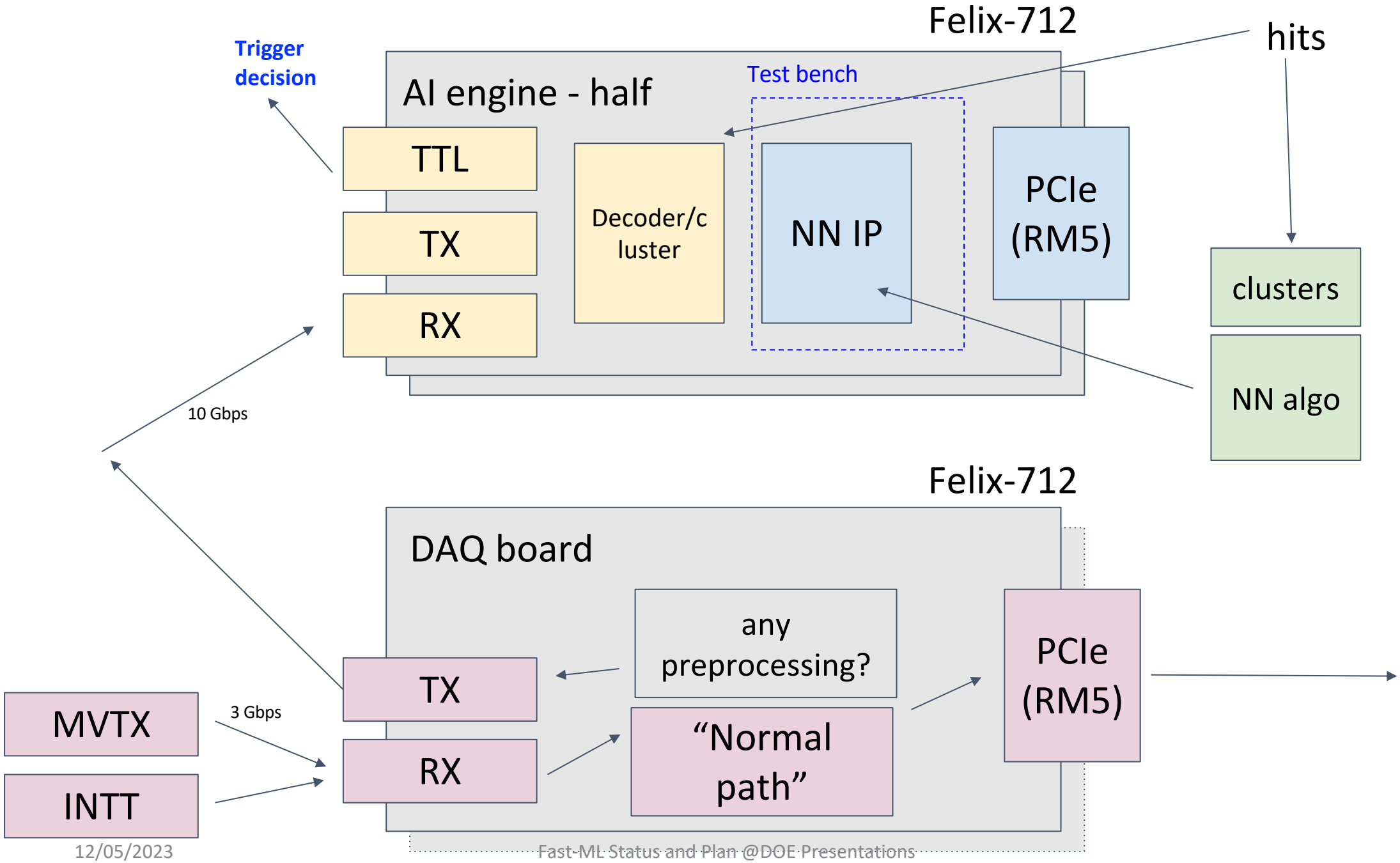


New systems include:

- Polarized electron source,
- Injector linac,
- Electron cooler complex,
- Rapid Cycling Synchrotron(RCS)
- Electron storage ring (ESR),
- Interaction region (IR) with 1 detector,
- Capability for implementing 2 IRs
- Infrastructure improvements.

SRO + AI/ML Fast Data Processing:
- DIS e-tagger: event ID
+ other rare process, HF-tagger etc. …




Felix-712
Trigger decision
AI engine - half
Test bench
hits
TTL
TX
RX
Decoder/cluster
NN IP
PCIe (RM5)
clusters
NN algo
10 Gbps
Felix-712
DAQ board
any preprocessing?
TX
RX
"Normal path"
PCIe (RM5)
MVTX
INTT
3 Gbps