Jefferson Lab
Thomas Jefferson National Accelerator Facility

Particle identification and tracking in real time using Machine Learning on FPGA

Sergey Furletov
Jefferson Lab

F. Barbosa,[a], L. Belfore,[b], C. Dickover,[a], C. Fanelli,[a,c], S. Furletov, [a], Y. Furletova,[a], L. Jokhovets,[d], D. Lawrence,[a], and D. Romanov[a]

[a]Jefferson Lab, U.S.A.
[b]Old Dominion University, U.S.A.
[c]William & Mary, U.S.A.
[d]Juelich Research Centre, Germany

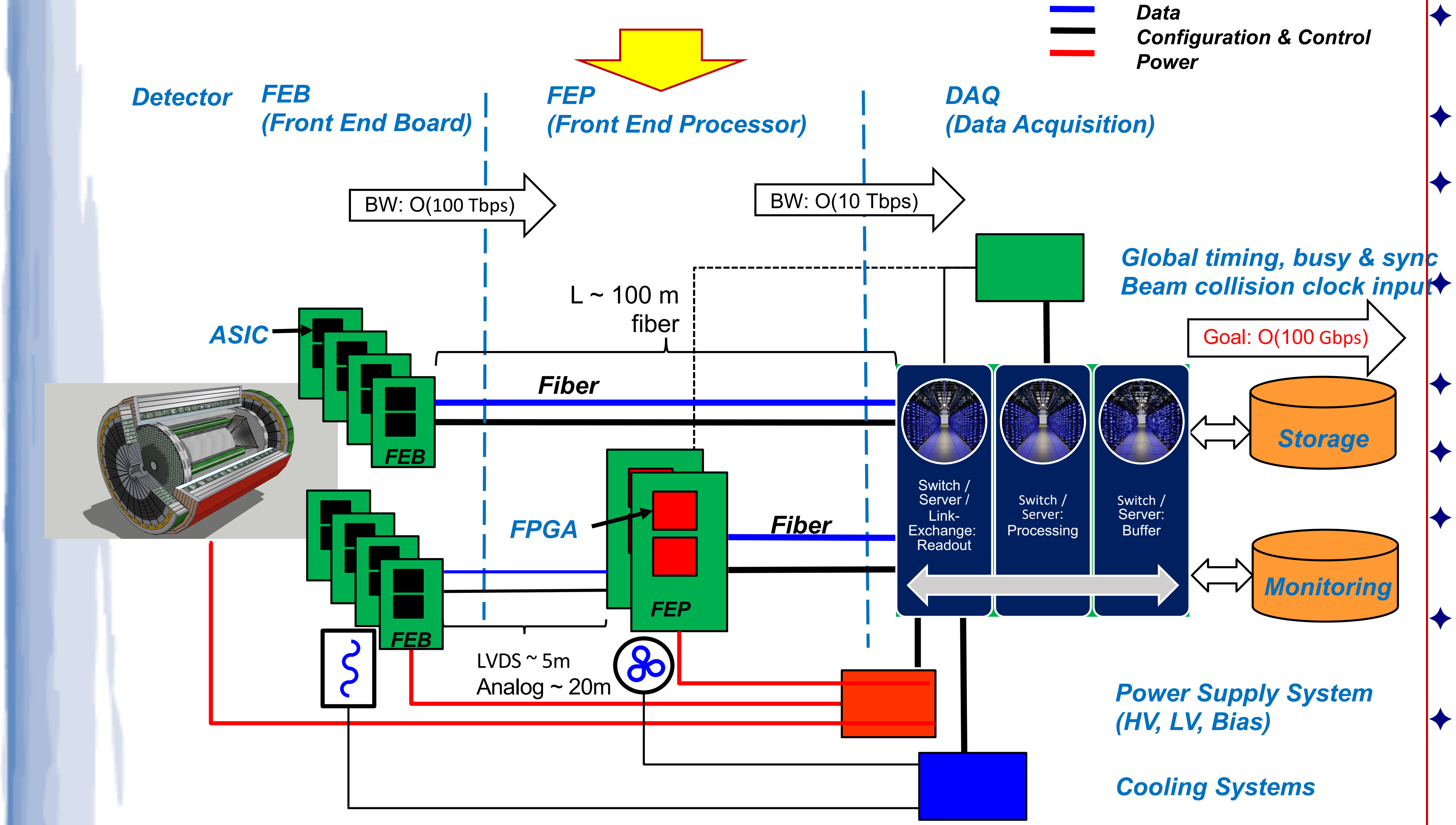
EIC generic R&D

16 Nov 2022

Motivation

Jefferson Lab
Thomas Jefferson National Accelerator Facility

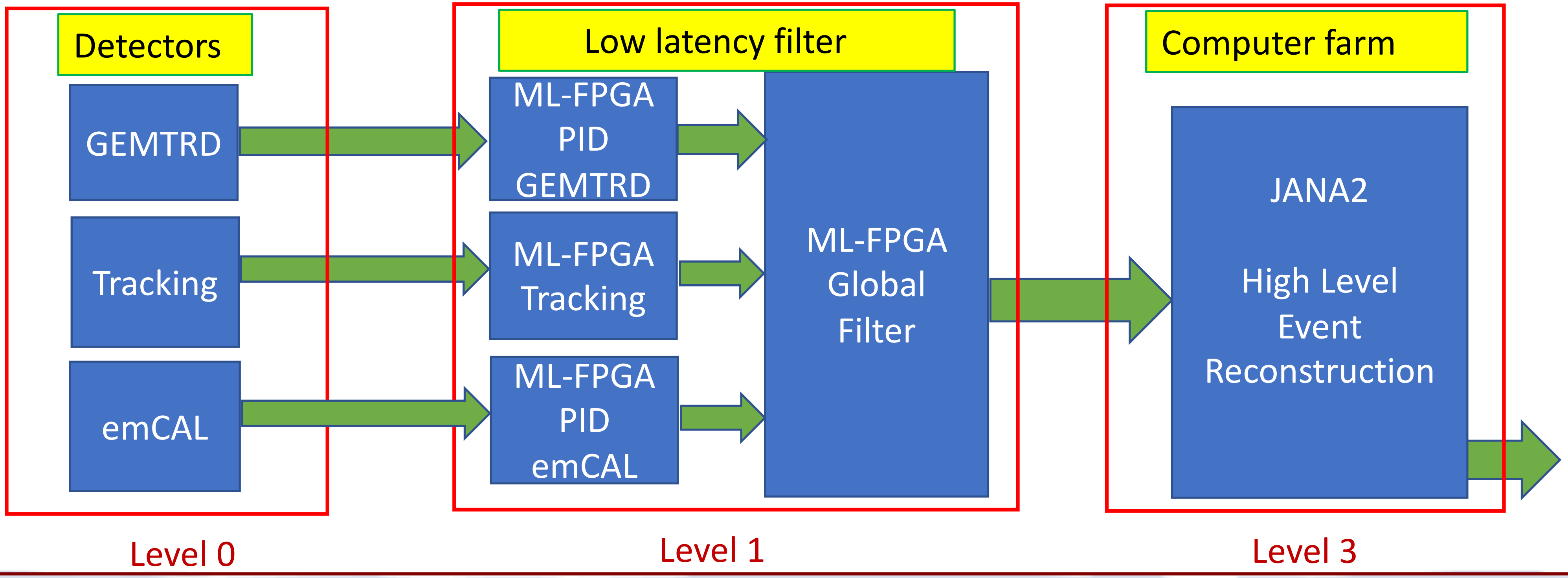
- *The growing computational power of modern FPGA boards allows us to add more sophisticated algorithms for real-time data processing.*
- *Many tasks, such as tracking and particle identification, could be solved using modern Machine Learning (ML) algorithms which are naturally suited for FPGA architectures.*



- *The correct location for the ML on the FPGA filter is called "FEP" in this figure.*
- *This gives us a chance to reduce traffic earlier.*
- *Allows us to touch physics: ML brings intelligence to L1.*
- *However, it is now unclear how far we can go with physics at the FPGA.*
- *Initially, we can start in pass-through mode.*
- *Then we can add background rejection.*
- *Later we can add filtering processes with the largest cross section.*
- *In case of problems with output traffic, we can add a selector for low cross section processes.*
- *The ML-on-FPGA solution complements the purely computer-based solution and mitigates DAQ performance risks.*

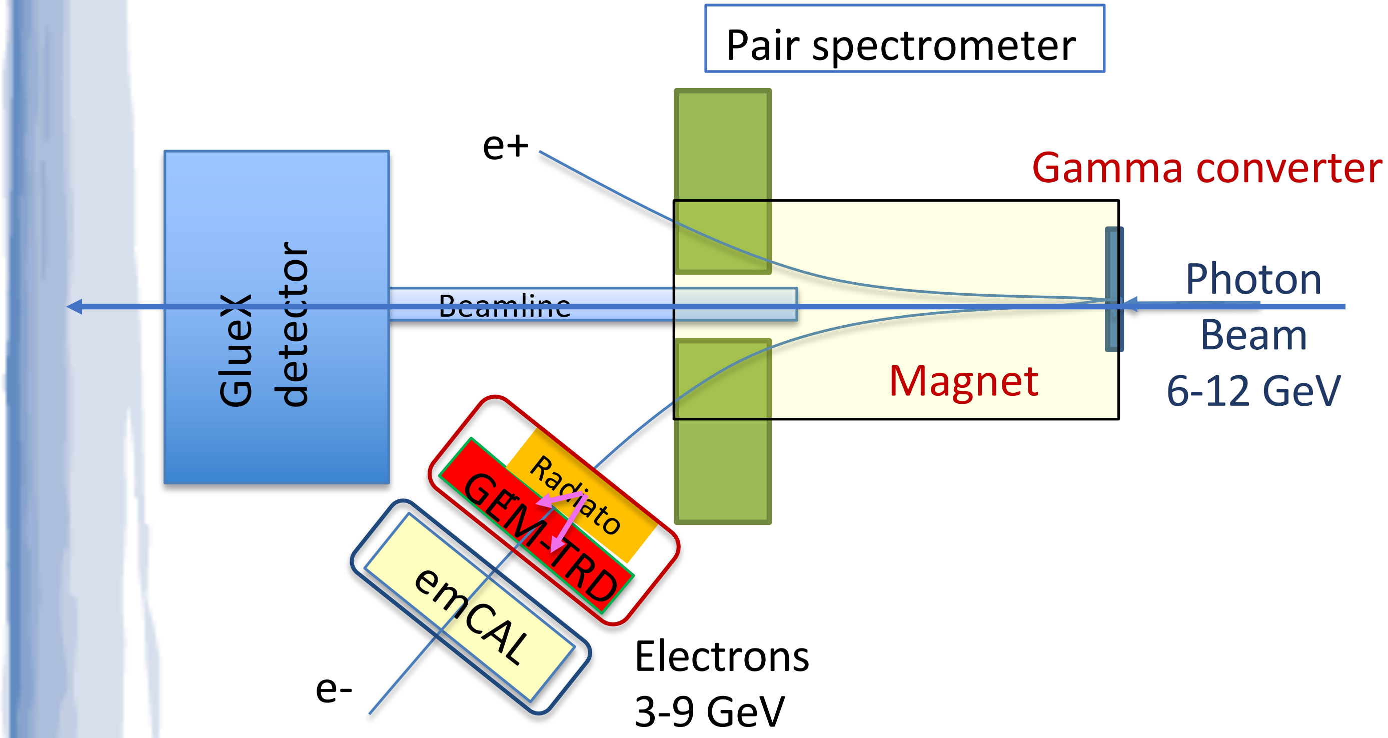
Processing chain / Global PID

- ❑ *Usually, several PID detectors are used in an experiment.*
- ❑ *For example, the GEM-TRD and e/m-calorimeter, both provide separation of electrons and hadrons.*
- ❑ *Summation and processing of joint data from both detectors at the early stages will increase the identification power of these detectors compared to independent identification.*
- ❑ *To test the "global PID" performance we work on integration of the EIC calorimeter prototype (3x3 modules) into the ML-FPGA setup.*
- ❑ *Preprocessed data from both detectors including decision on the particle type will be transferred to another ML-FPGA board with neural network for global PID decision.*

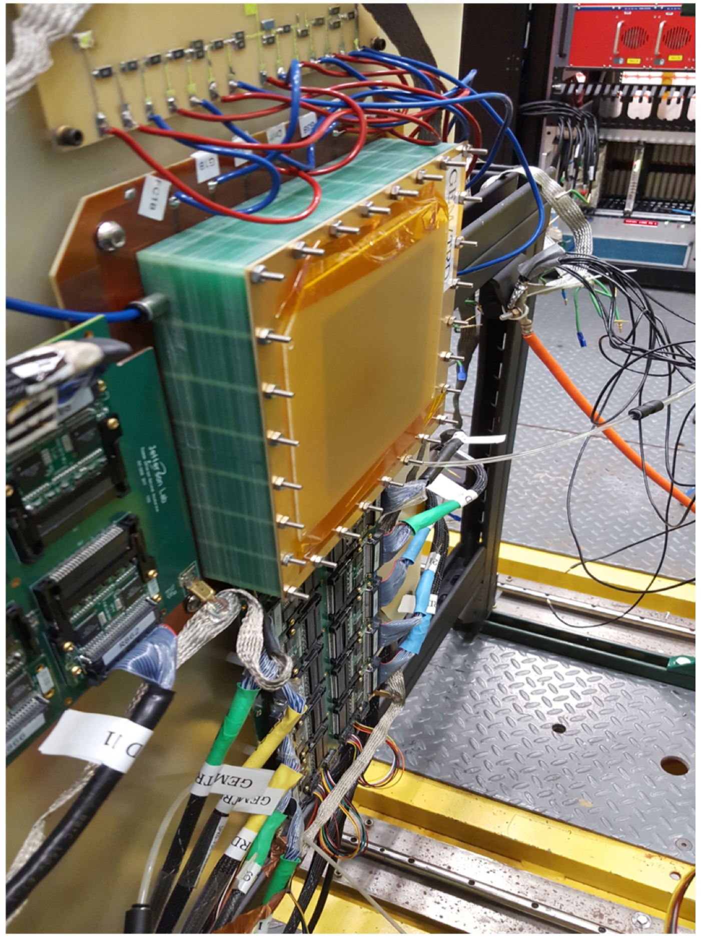


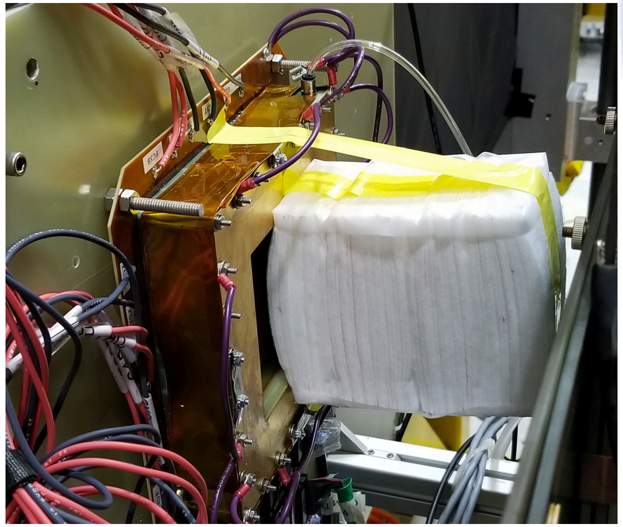
Beam setup at JLab Hall-D

- *Tests were carried out using electrons with an energy of 3-6 GeV, produced in the converter of a pair spectrometer at the upstream of GlueX detector.*



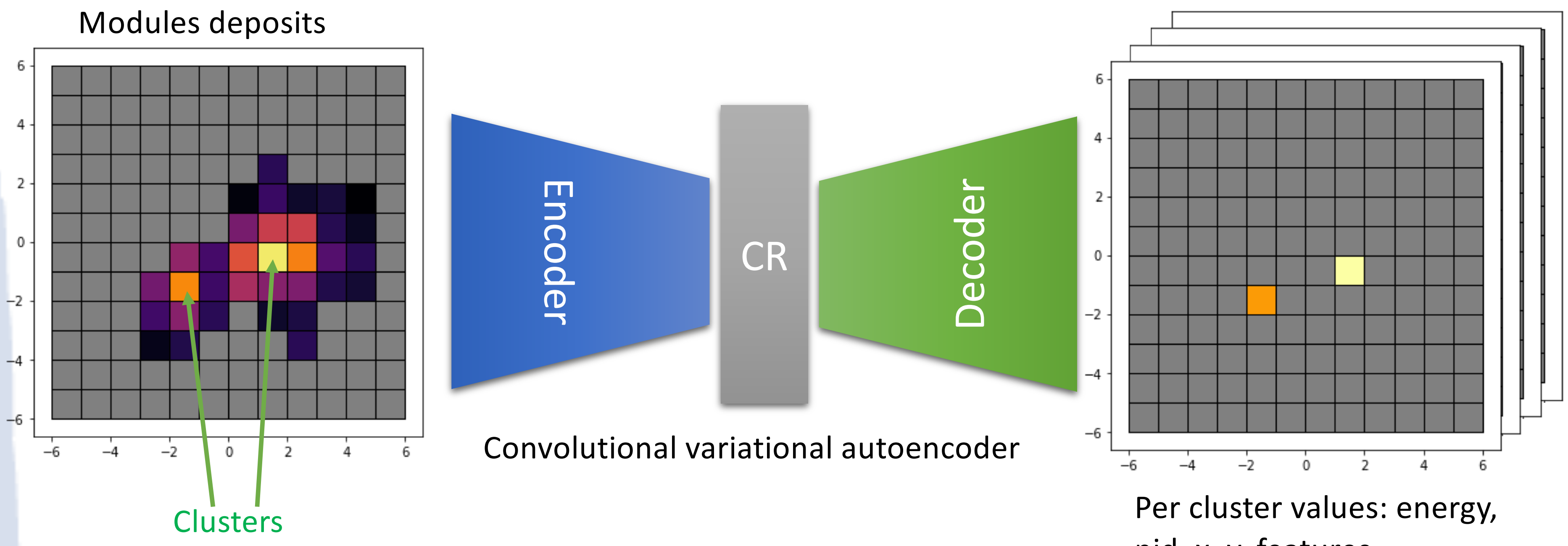
GEMTRD prototype





Calorimeter parameters reconstruction

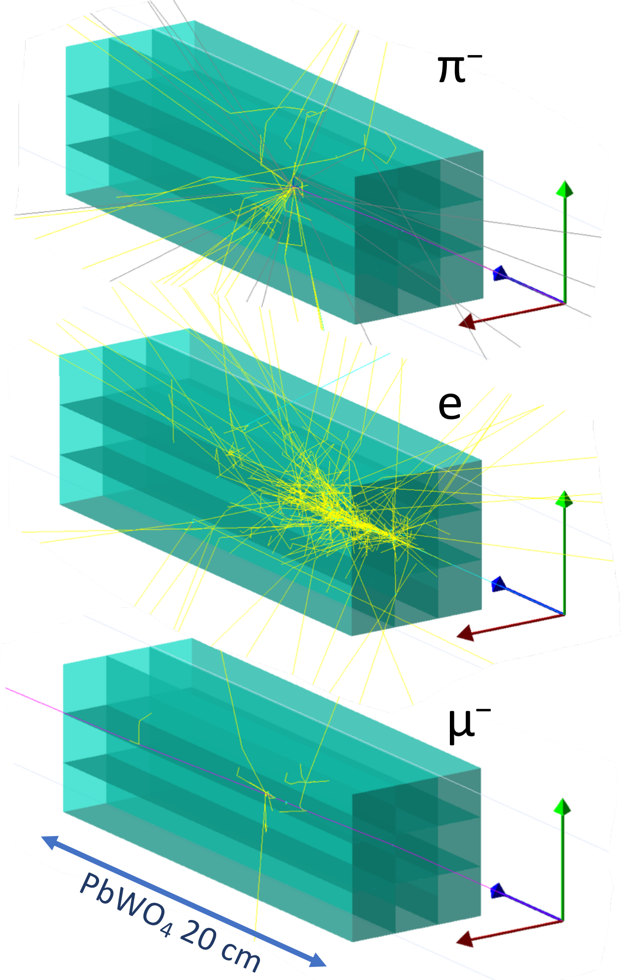
By Dmitry Romanov



Convolutional variational autoencoder

Per cluster values: energy, pid, x, y, features

Geant 4 simulation

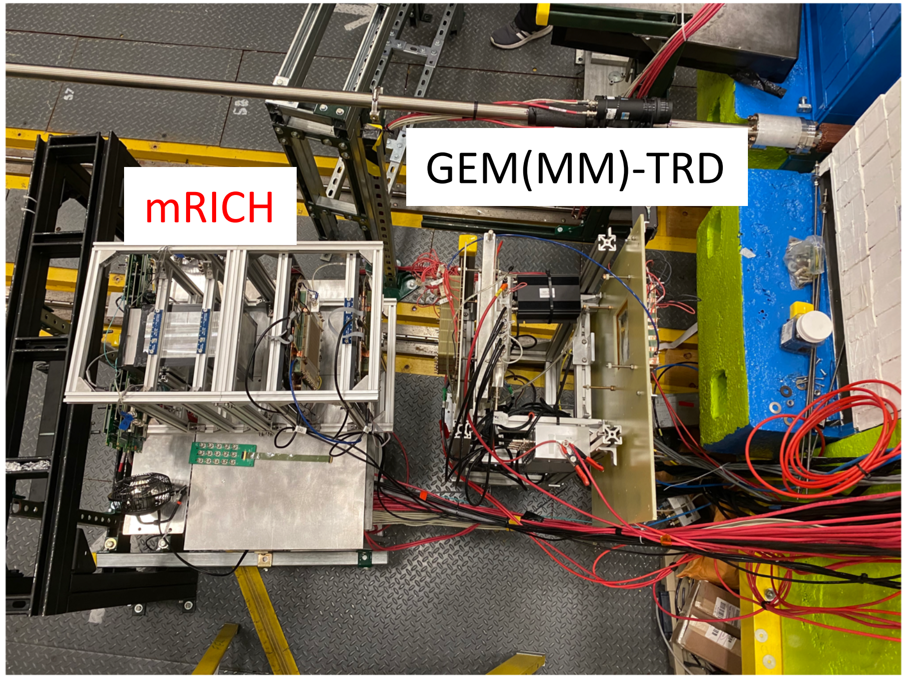


Examples of events with e and π^- showers and μ^- passing through.

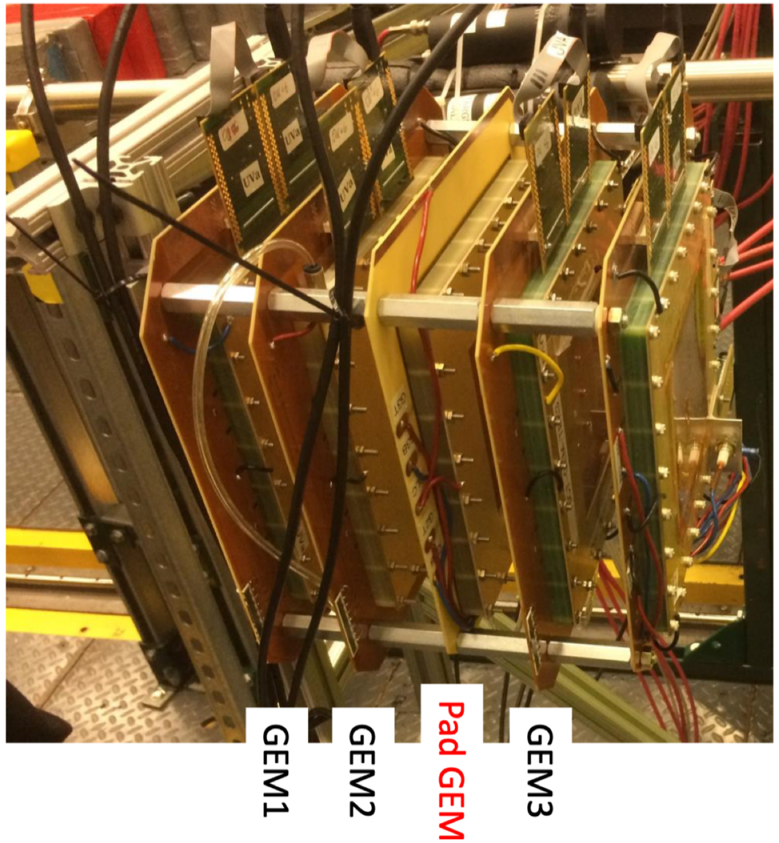
- Convolutional VAE as a backbone
- Modules deposits as inputs
- Per cluster output of multiple values:
- Energy, e/ π, coordinates, features

EIC detectors prototypes in Hall-D test beam

mRICH and GEM MM



uRWELL, pad-GEM with capacitive-sharing readout — Kondo



Summary

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- ❑ *An FPGA-based Neural Network application would offer online event preprocessing and allow for data reduction based on physics at the early stage of data processing.*
- ❑ *The ML-on-FPGA solution complements the purely computer-based solution and mitigates DAQ performance risks.*
- ❑ *FPGA provides extremely low-latency neural-network inference.*
- ❑ *Open-source HLS4ML software tool with Xilinx® Vivado® High Level Synthesis (HLS) accelerates machine learning neural network algorithm development.*

- ❑ *The ultimate goal is to build a real-time event filter/tagger based on physics signatures.*

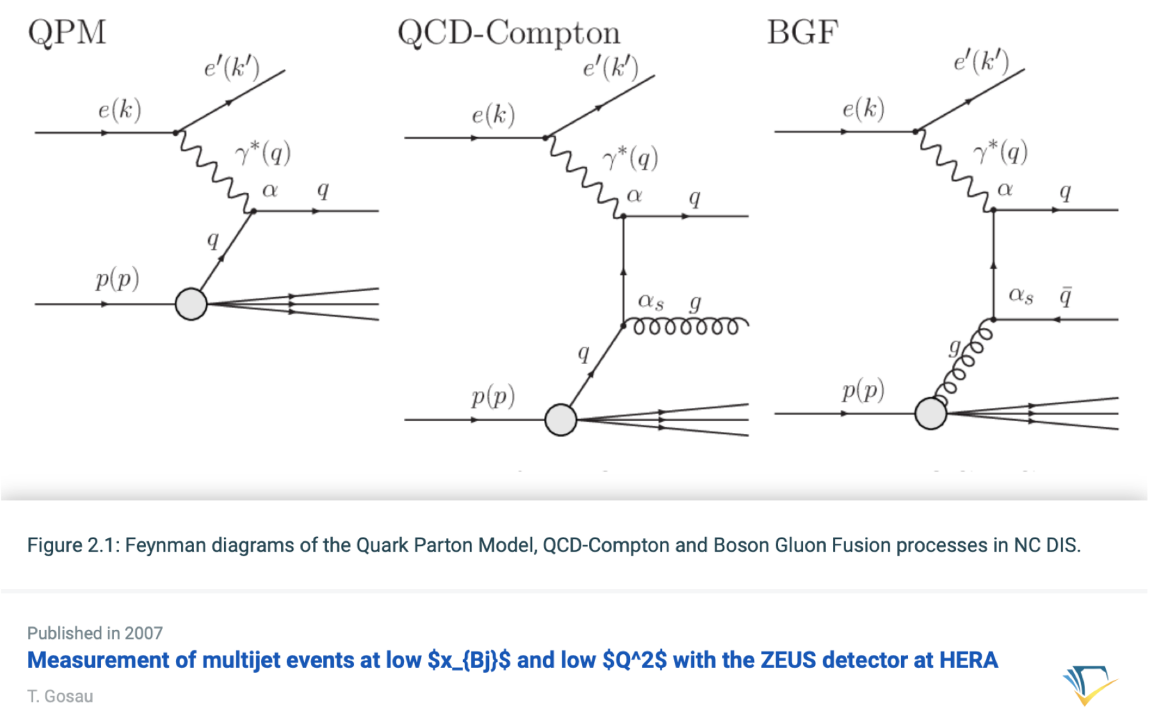
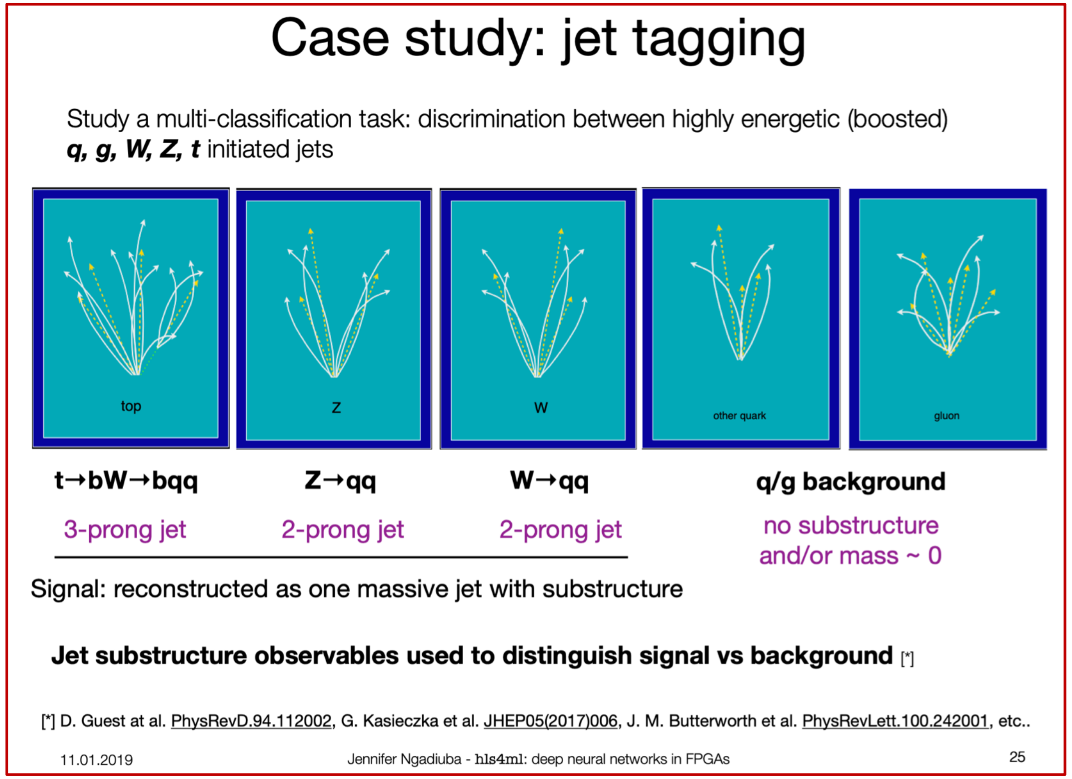


Figure 2.1: Feynman diagrams of the Quark Parton Model, QCD-Compton and Boson Gluon Fusion processes in NC DIS.

Published in 2007

Measurement of multijet events at low x_{Bj} and low Q^2 with the ZEUS detector at HERA

T. Gosau



Case study: jet tagging

Study a multi-classification task: discrimination between highly energetic (boosted) ***q, g, W, Z, t*** initiated jets

t→bW→bqq	Z→qq	W→qq	q/g background
3-prong jet	2-prong jet	2-prong jet	no substructure and/or mass ~ 0

Signal: reconstructed as one massive jet with substructure

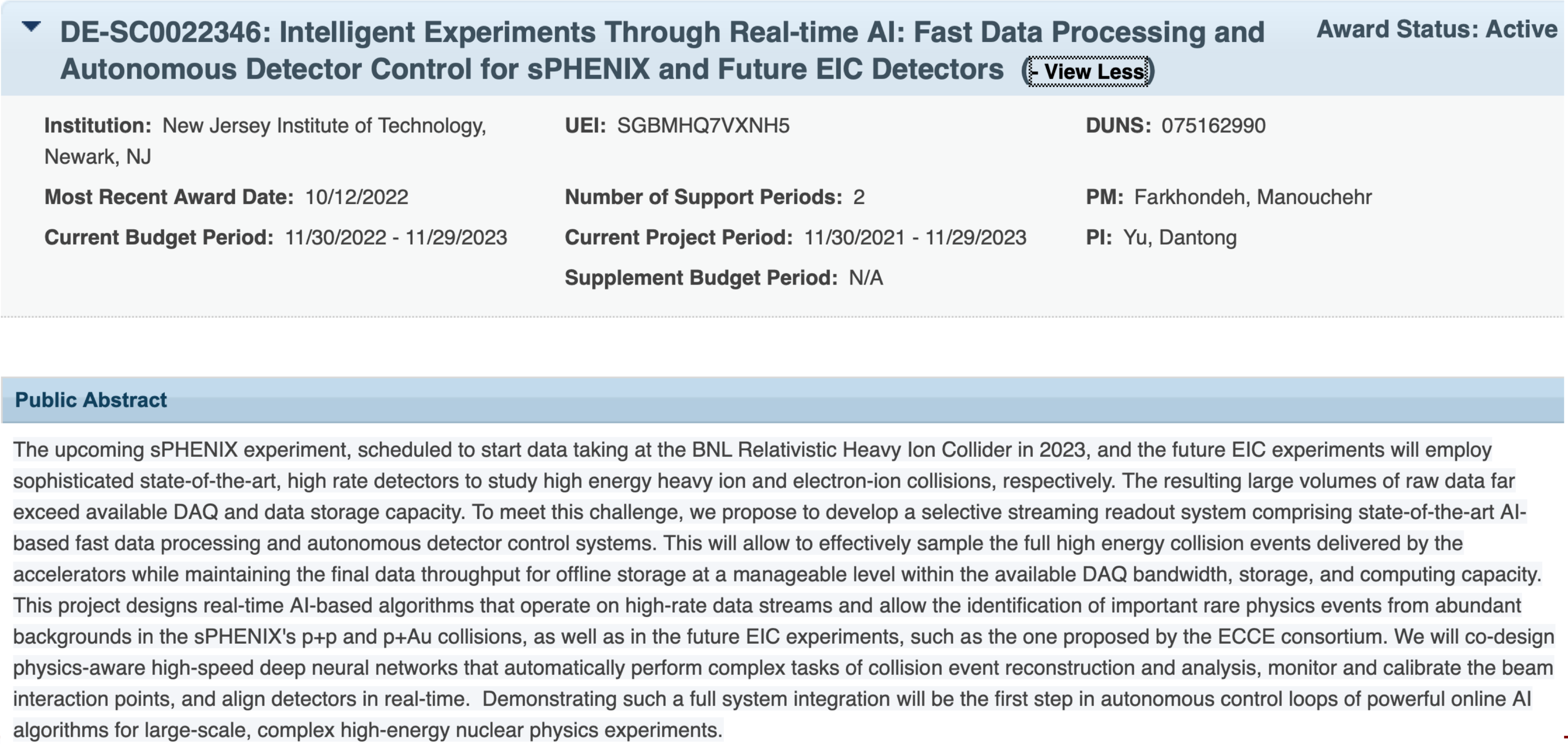
Jet substructure observables used to distinguish signal vs background [*]

[*] D. Guest at al. PhysRevD.94.112002, G. Kasieczka et al. JHEP05(2017)006, J. M. Butterworth et al. PhysRevLett.100.242001, etc..

11.01.2019 Jennifer Ngadiuba - hls4ml: deep neural networks in FPGAs 25

Question 1

There appears to be overlap in the proposed research with the current DOE-funded project on FPGA-ML tracking/full event tagging for RHIC/EIC under DE-FOA-0002490 by LANL-MIT-FNAL-NJIT . Can you comment on how this proposal will complement that effort?



DE-SC0022346: Intelligent Experiments Through Real-time AI: Fast Data Processing and Autonomous Detector Control for sPHENIX and Future EIC Detectors (- View Less)

Award Status: Active

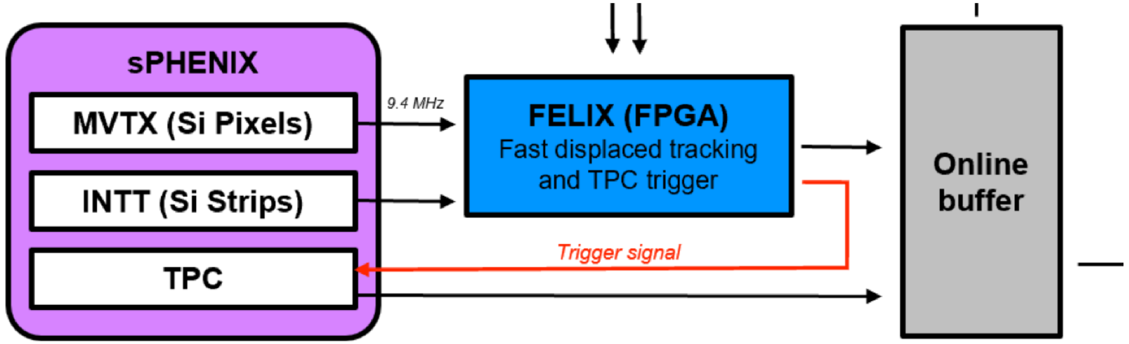
Institution: New Jersey Institute of Technology, Newark, NJ	**UEI:** SGBMHQ7VXNH5	**DUNS:** 075162990
Most Recent Award Date: 10/12/2022	**Number of Support Periods:** 2	**PM:** Farkhondeh, Manouchehr
Current Budget Period: 11/30/2022 - 11/29/2023	**Current Project Period:** 11/30/2021 - 11/29/2023	**PI:** Yu, Dantong
	Supplement Budget Period: N/A	

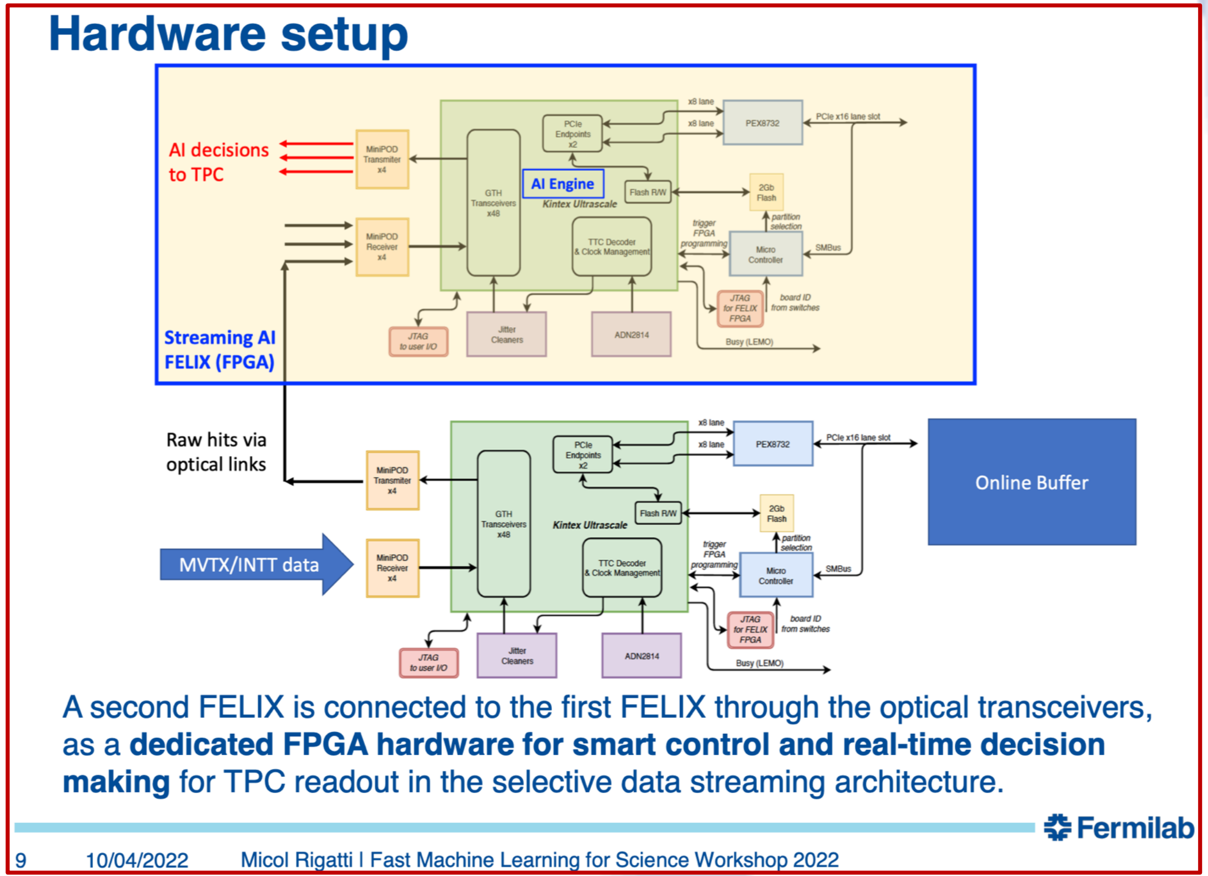
Public Abstract

The upcoming sPHENIX experiment, scheduled to start data taking at the BNL Relativistic Heavy Ion Collider in 2023, and the future EIC experiments will employ sophisticated state-of-the-art, high rate detectors to study high energy heavy ion and electron-ion collisions, respectively. The resulting large volumes of raw data far exceed available DAQ and data storage capacity. To meet this challenge, we propose to develop a selective streaming readout system comprising state-of-the-art AI-based fast data processing and autonomous detector control systems. This will allow to effectively sample the full high energy collision events delivered by the accelerators while maintaining the final data throughput for offline storage at a manageable level within the available DAQ bandwidth, storage, and computing capacity. This project designs real-time AI-based algorithms that operate on high-rate data streams and allow the identification of important rare physics events from abundant backgrounds in the sPHENIX's p+p and p+Au collisions, as well as in the future EIC experiments, such as the one proposed by the ECCE consortium. We will co-design physics-aware high-speed deep neural networks that automatically perform complex tasks of collision event reconstruction and analysis, monitor and calibrate the beam interaction points, and align detectors in real-time. Demonstrating such a full system integration will be the first step in autonomous control loops of powerful online AI algorithms for large-scale, complex high-energy nuclear physics experiments.

Question 1 (cont 1)

- ❑ *The project mentioned above has a broad title that allows them to work on any topic of Real-Time AI applications for experiments and detectors, but narrowed down to the sPHENIX and EIC experiments.*
- ❑ *The last reports of this group at the workshops reveal some details about the direction of their work:*
  - ➢ A fast search for displaced tracks will be performed by AI-trained FPGA to identify tracks from heavy quark decays that are pointing away from the nominal beam center.
- ❑ *In other words, they're going to make a trigger for displaced vertices based on AI in RealTime.*
- ❑ *Work will be done for the sPHENIX experiment and will therefore build on existing DAQ hardware.*





Hardware setup

A second FELIX is connected to the first FELIX through the optical transceivers, as a **dedicated FPGA hardware for smart control and real-time decision making** for TPC readout in the selective data streaming architecture.

Fermilab

9 10/04/2022 Micol Rigatti I Fast Machine Learning for Science Workshop 2022

- ❑ *This field is new and there are no ready-made solutions yet, so any information from other groups working in this direction will be interesting and useful.*
  - ➢ i.e. despite the fact that our program also has track finding, the methods used for this can be different.

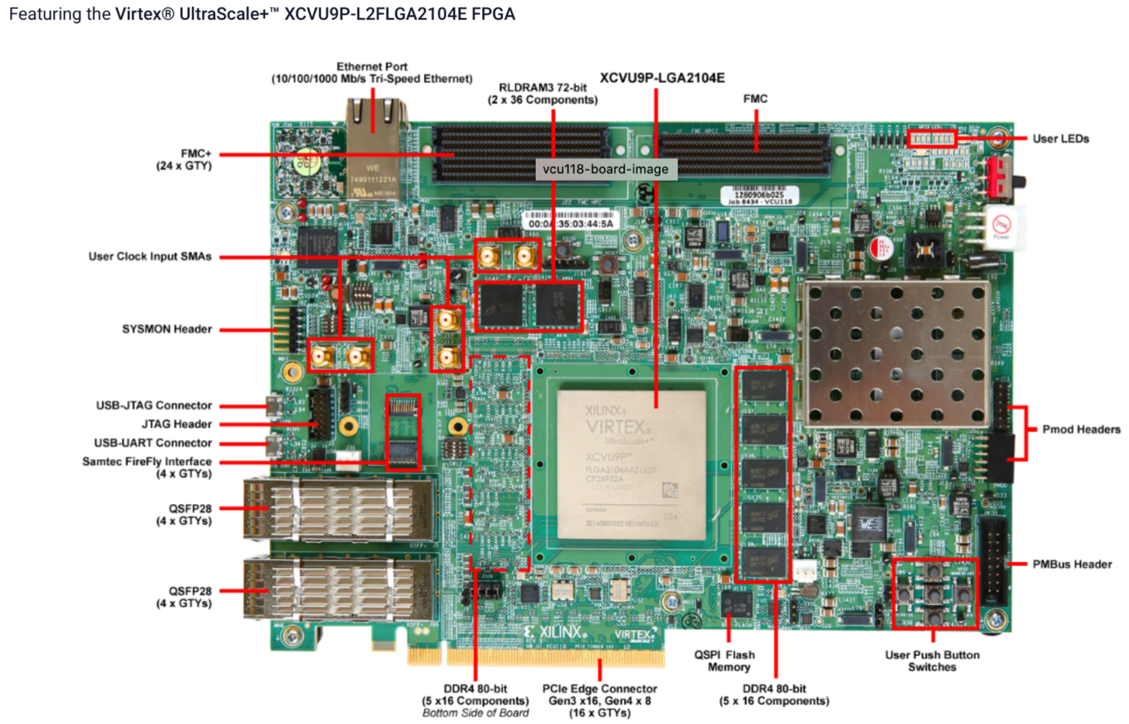
Question 1 (cont 2) FPGA test board for ML

- At an early stage in this project, as hardware to test ML algorithms on FPGA , we use a standard Xilinx evaluation boards rather than developing a customized FPGA board. These boards have functions and interfaces sufficient for proof of principle of ML-FPGA.

- The Xilinx evaluation board includes the Xilinx XCVU9P and 6,840 DSP slices. Each includes a hardwired optimized multiply unit and collectively offers a peak theoretical performance in excess of 1 Tera multiplications per second.

- Second, the internal organization can be optimized to the specific computational problem. The internal data processing architecture can support deep computational pipelines offering high throughputs.

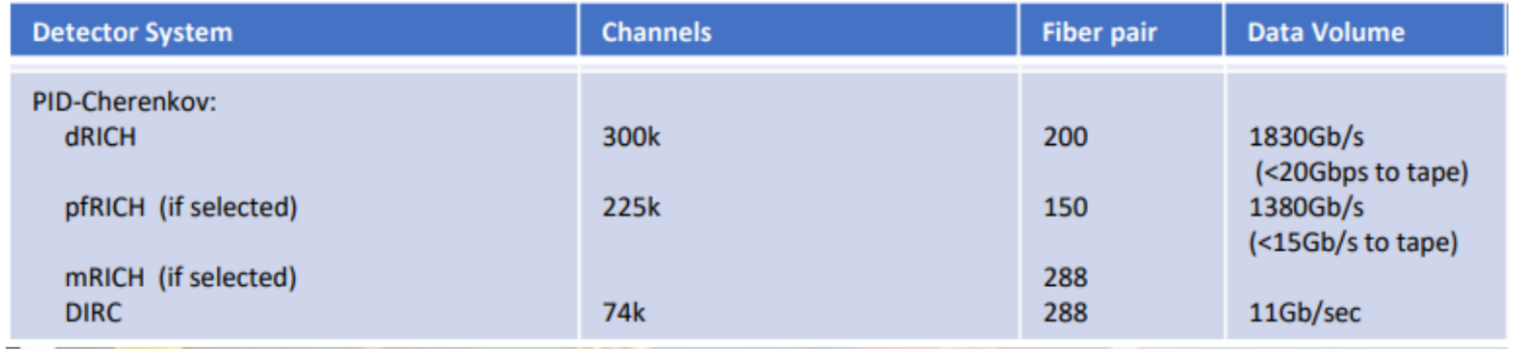
- Third, the FPGA supports high speed I/O interfaces including Ethernet and 180 high speed transceivers that can operate in excess of 30 Gbps.


Featuring the Virtex® UltraScale+™ XCVU9P-L2FLGA2104E FPGA
Ethernet Port (10/100/1000 Mb/s Tri-Speed Ethernet)
RLDRAM3 72-bit (2 x 36 Components)
XCVU9P-LGA2104E
FMC
User LEDs
FMC+ (24 x GTY)
vcu118-board-image
User Clock Input SMAs
SYSMON Header
USB-JTAG Connector
JTAG Header
USB-UART Connector
Samtec FireFly Interface (4 x GTYs)
QSFP28 (4 x GTYs)
QSFP28 (4 x GTYs)
Pmod Headers
PMBus Header
QSPI Flash Memory
User Push Button Switches
DDR4 80-bit (5 x16 Components) Bottom Side of Board
PCIe Edge Connector Gen3 x16, Gen4 x 8 (16 x GTYs)
DDR4 80-bit (5 x 16 Components)

Xilinx Virtex® UltraScale+™

Question 2

Can you make a clear case that the proposed R&D addresses problems that are currently projected to be bottlenecks for EIC Detector 2 or an upgrade of Detector 1? For example, is it a mere detail that Detector 1 tracking technology is based on silicon whereas this proposal would use GEMs? Are the noise and track reconstruction challenges similar? The EIC-related generic R&D program cannot support overly generic R&D.



Detector System	Channels	Fiber pair	Data Volume
PID-Cherenkov:			
dRICH	300k	200	1830Gb/s (<20Gbps to tape)
pfRICH (if selected)	225k	150	1380Gb/s (<15Gb/s to tape)
mRICH (if selected)		288	
DIRC	74k	288	11Gb/sec

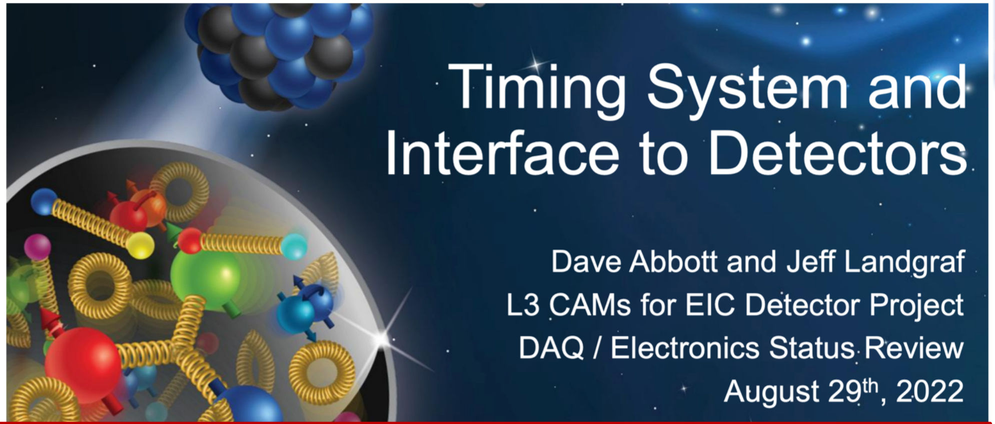
- ❑ *One known bottleneck is data traffic from dRICH. dRICH in its current design is based on SiPM readout and can produce up to 1.8 Tb/s of data including noise.*
- ❑ *One of the methods for cleaning dRICH data from noise hits could be the reconstruction of tracks in dRICH using other tracking detectors before and/or after dRICH, followed by reconstruction of the rings.*
- ❑ *Track reconstruction using ML is quite general and is usually based on 2D or 3D hits in space.*

  Of course, the amount of noise can affect tracking performance and especially scalability.

  However, the detector technology itself - GEM vs silicon should not be a problem.

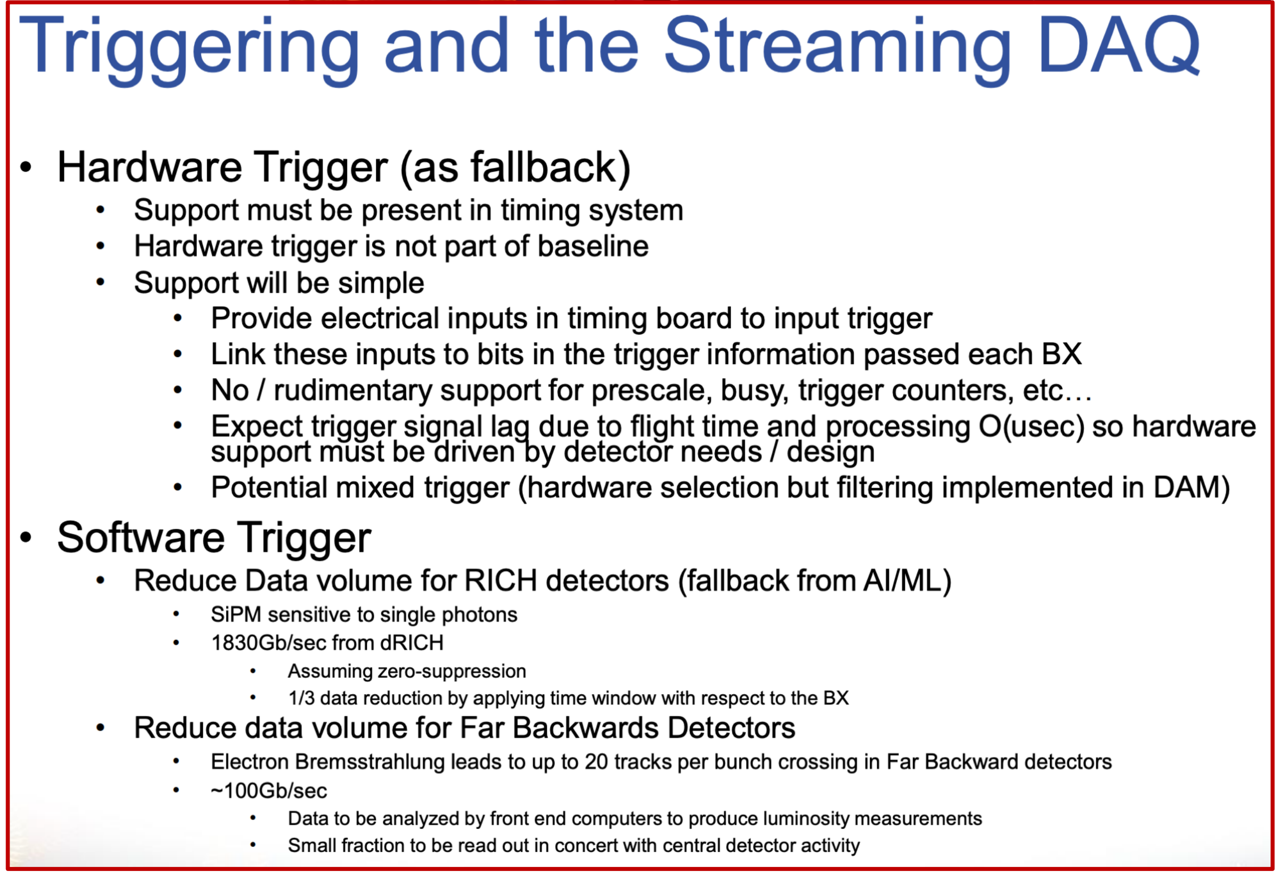
Question 2 (cont)

- ❑ *The current EIC/DAQ design also considers the use of a hardware trigger as a fallback solution.*
- ❑ *The growing computational power of modern FPGA boards allows us to add more sophisticated algorithms for real-time data processing.*
- ❑ *Many tasks, such as tracking and particle identification, could be solved using modern Machine Learning (ML) algorithms which are naturally suited for FPGA architectures.*
- ❑ *Performing a physics event reconstruction at Level 1 can provide a more efficient and clean trigger.*
- ❑ *The ML-on-FPGA solution complements the purely computer-based solution and mitigates DAQ performance risks.*



Timing System and Interface to Detectors

Dave Abbott and Jeff Landgraf
L3 CAMs for EIC Detector Project
DAQ / Electronics Status Review
August 29th, 2022



Triggering and the Streaming DAQ

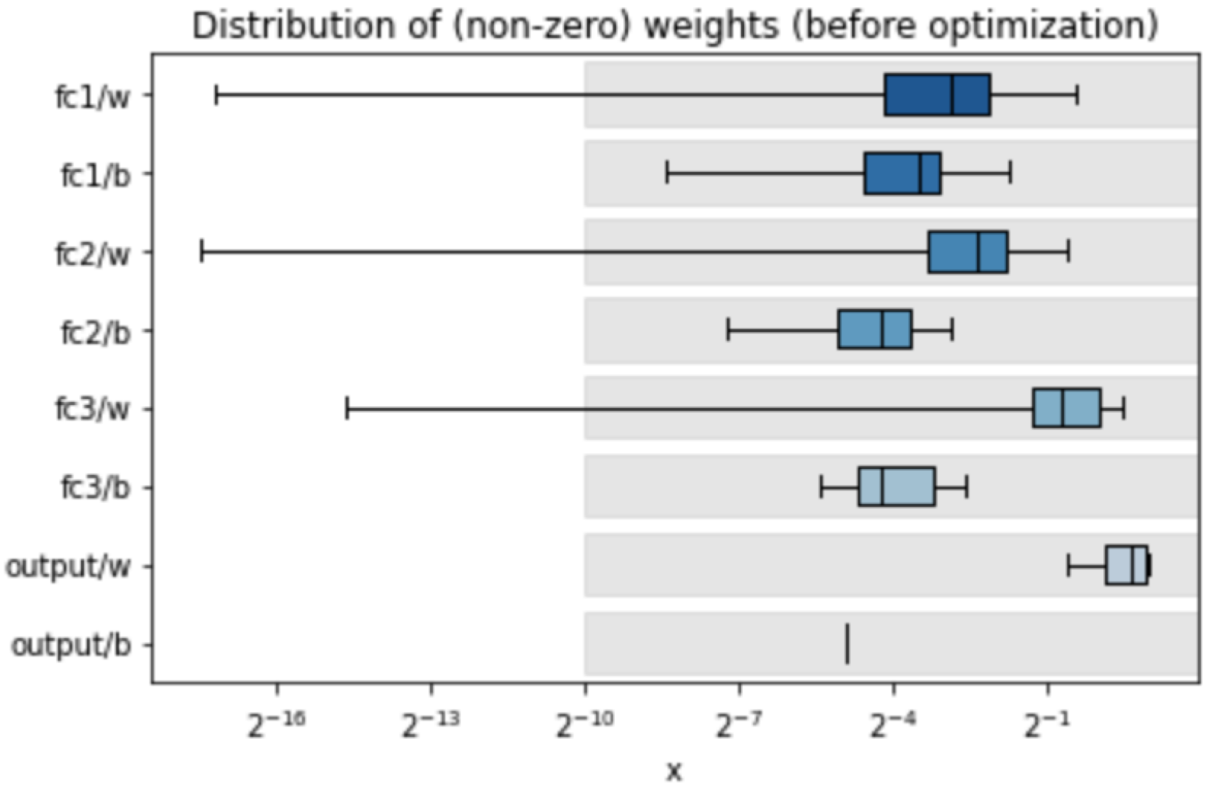
- Hardware Trigger (as fallback)
  - Support must be present in timing system
  - Hardware trigger is not part of baseline
  - Support will be simple
    - Provide electrical inputs in timing board to input trigger
    - Link these inputs to bits in the trigger information passed each BX
    - No / rudimentary support for prescale, busy, trigger counters, etc…
    - Expect trigger signal lag due to flight time and processing O(usec) so hardware support must be driven by detector needs / design
    - Potential mixed trigger (hardware selection but filtering implemented in DAM)
- Software Trigger
  - Reduce Data volume for RICH detectors (fallback from AI/ML)
    - SiPM sensitive to single photons
    - 1830Gb/sec from dRICH
      - Assuming zero-suppression
      - 1/3 data reduction by applying time window with respect to the BX
  - Reduce data volume for Far Backwards Detectors
    - Electron Bremsstrahlung leads to up to 20 tracks per bunch crossing in Far Backward detectors
    - ~100Gb/sec
      - Data to be analyzed by front end computers to produce luminosity measurements
      - Small fraction to be read out in concert with central detector activity

Question 3

Please discuss the reliability of the proposed methods, in particular for the following areas:

- What amount of by-passing data is required to study the systematic uncertainty on the acceptance efficiency for the proposed full event ML trigger/tagger?
- Please quantify the performance difference for the inference network between the training environment and FPGA implementation (e.g. via emulation with QONNX), as different numerical precisions are used in these two environments.
- Please comment on the following factors in the algorithm design: competence awareness, quantization aware training, and whether/how calibration is used.

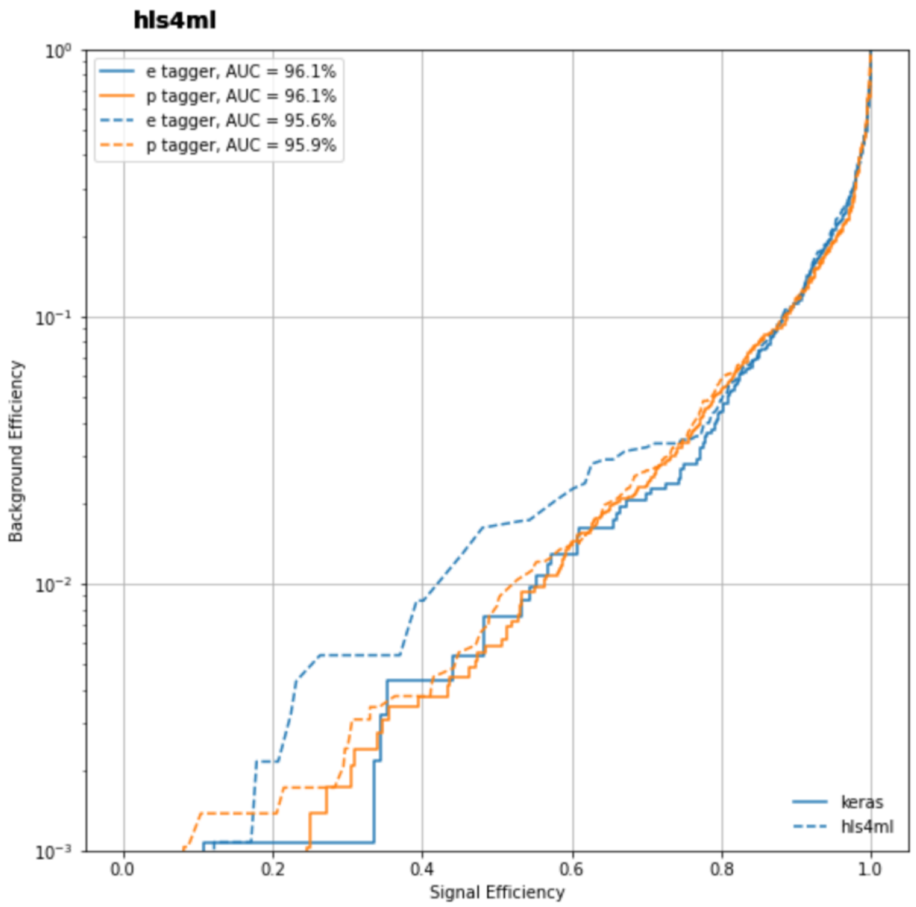
1. *The amount of by-passing data depends on the specific variables provided by the neural network. If it's measurable values like tracks, clusters, PID then it's pretty fast, 100Hz should be enough. For complex parameters such as the charmed meson invariant mass, this depends on the stability of the detectors and calibration.*
2. *In development, we actively use the open source HLS4ML software. It provides tools for analyzing and optimizing a neural network before implementing it into an FPGA.*
   1. - compression: reduce number of synapses or neurons
   2. - quantization: reduces the precision of the calculations (inputs, weights, biases)
   3. - parallelization: tune how much to parallelize to make the inference faster/slower versus FPGA resources



Question 3 (cont)

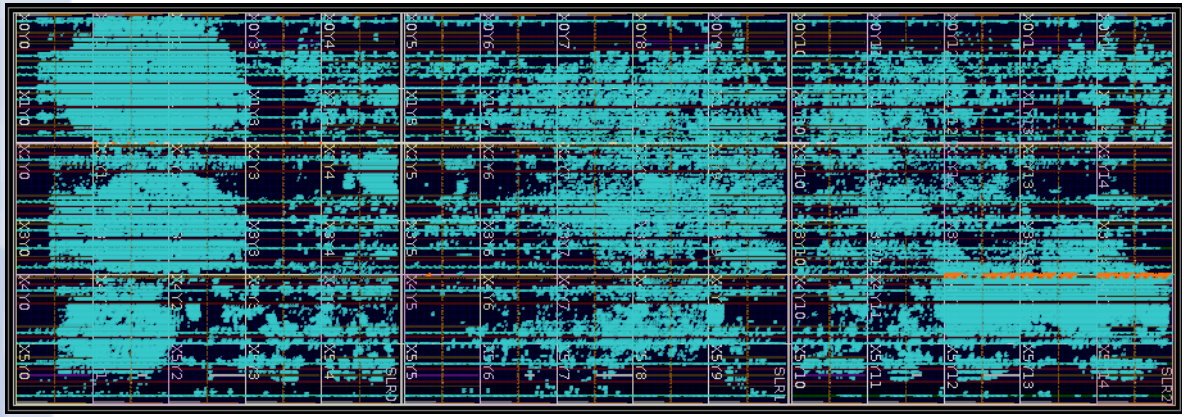
1. *The precision for used values can be adjusted manually looking on the distribution.*
2. *The result can be controlled by simulation and tests:*
    1. using HLS4ML
    2. using Xilinx HLS C-simulation
    3. using Xilinx C/RTL Co-simulation
    4. using Xilinx Vivado to build a test bench in FPGA.

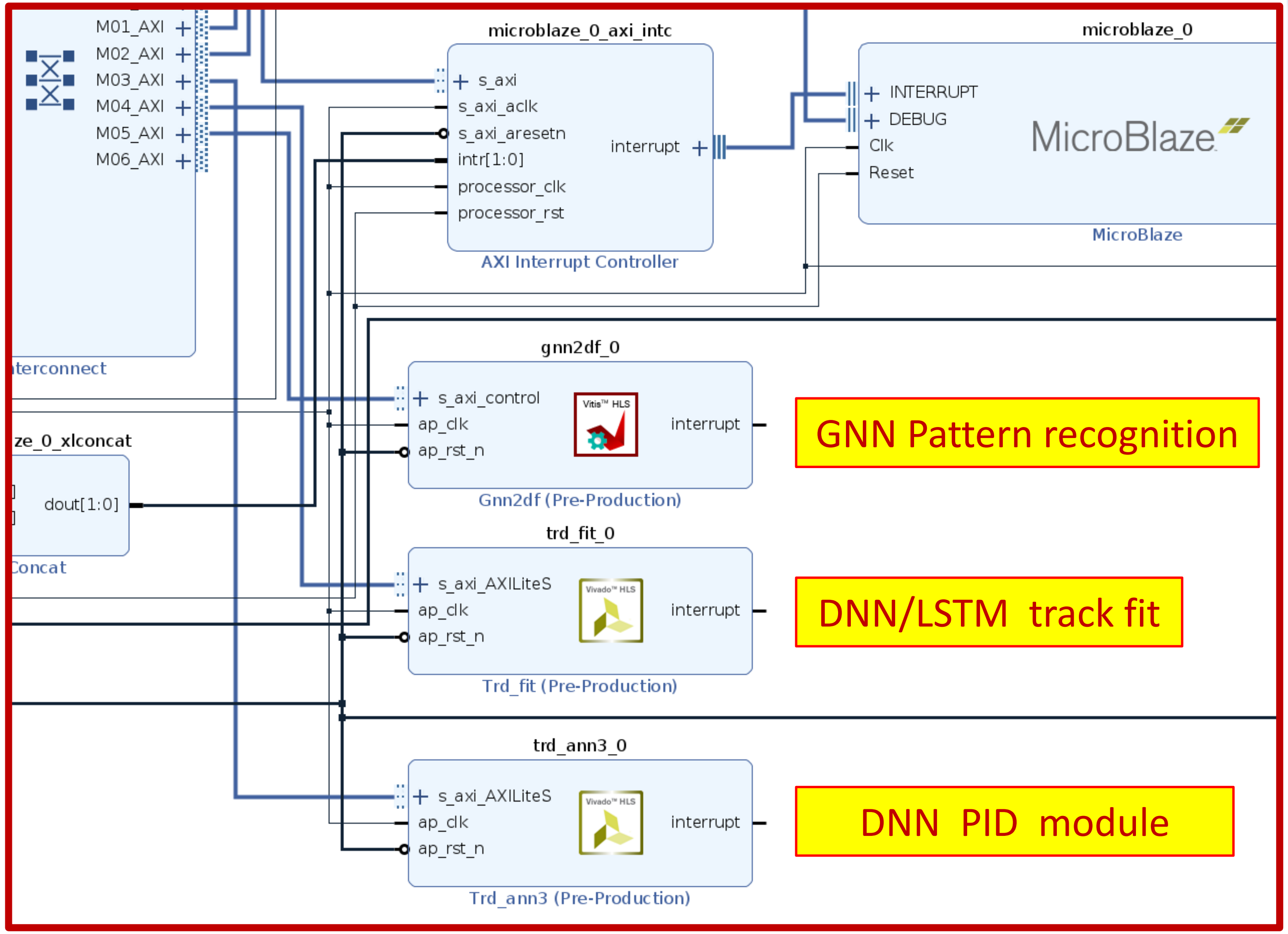
3. *HLS4ML also provides an interface for training using QKeras “quantization aware training” and study impact on FPGA metrics.*
    - ➢ QKeras is a library to train models with quantization in the training, maintained by Google
4. *The question of the accuracy of ML is generally quite complicated. In practice, it can be estimated using realistic Monte Carlo simulations or by comparison with the results obtained using conventional algorithms when working with experimental data.*
5. *The quantization aware training is usually compared to the full version of the neural network used as a reference.*



Question 3 (cont) FPGA test bench

- ❑ *The logic test was performed with the MicroBlaze processor and the AXI Lite interface.*





GNN Pattern recognition

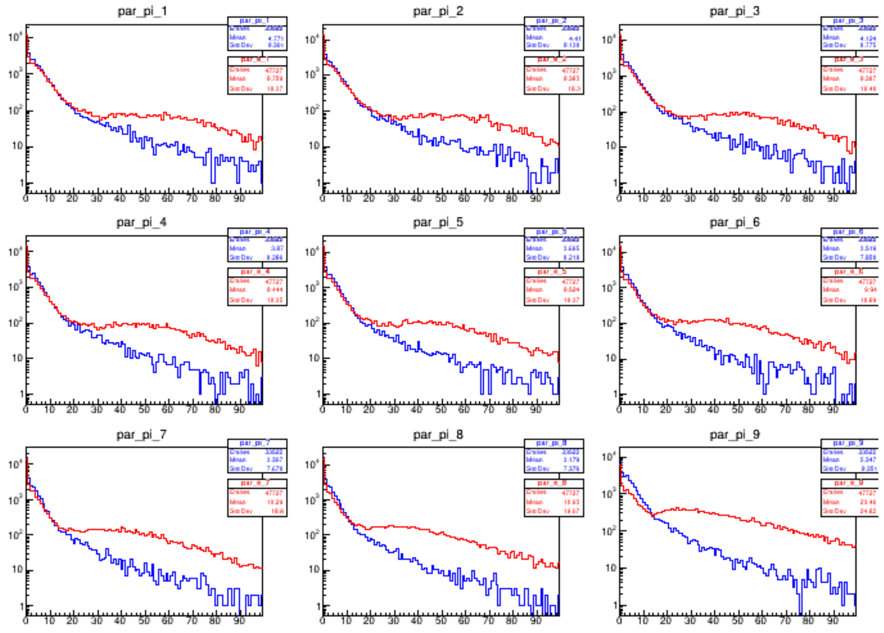
DNN/LSTM track fit

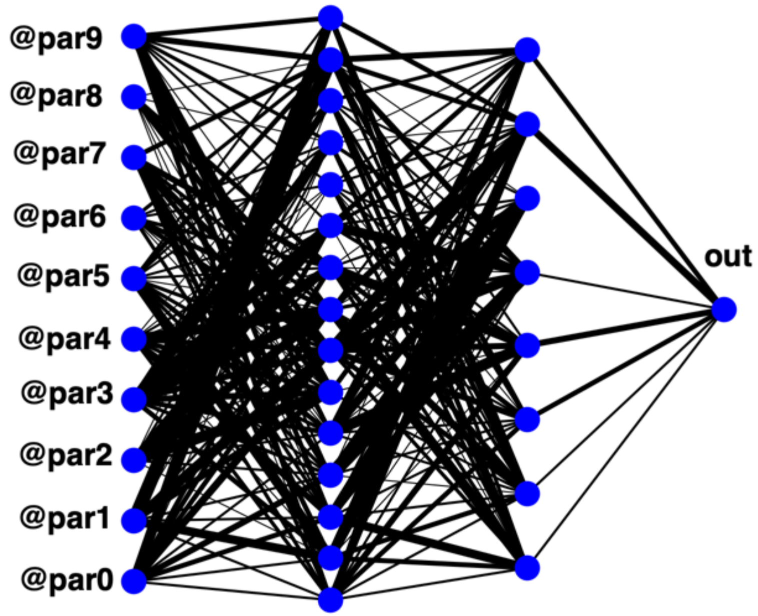
DNN PID module

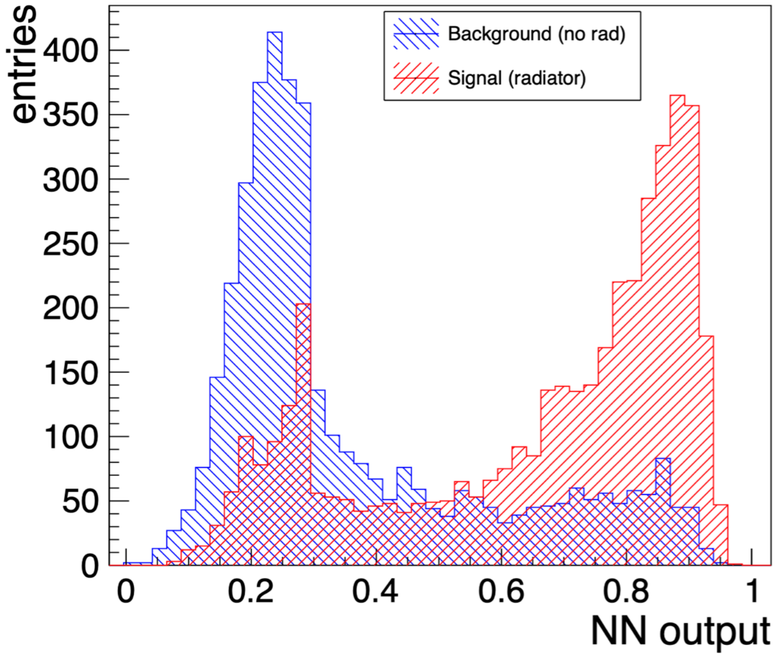
Question 4

Why implement the algorithms immediately in an FPGA board and not test it beforehand with existing data in a CPU? Of course this is much slower but will show how good the algorithms work.

- ❑ *Yes, we have been using a neural network for offline data processing from GEMTRD for a long time.*





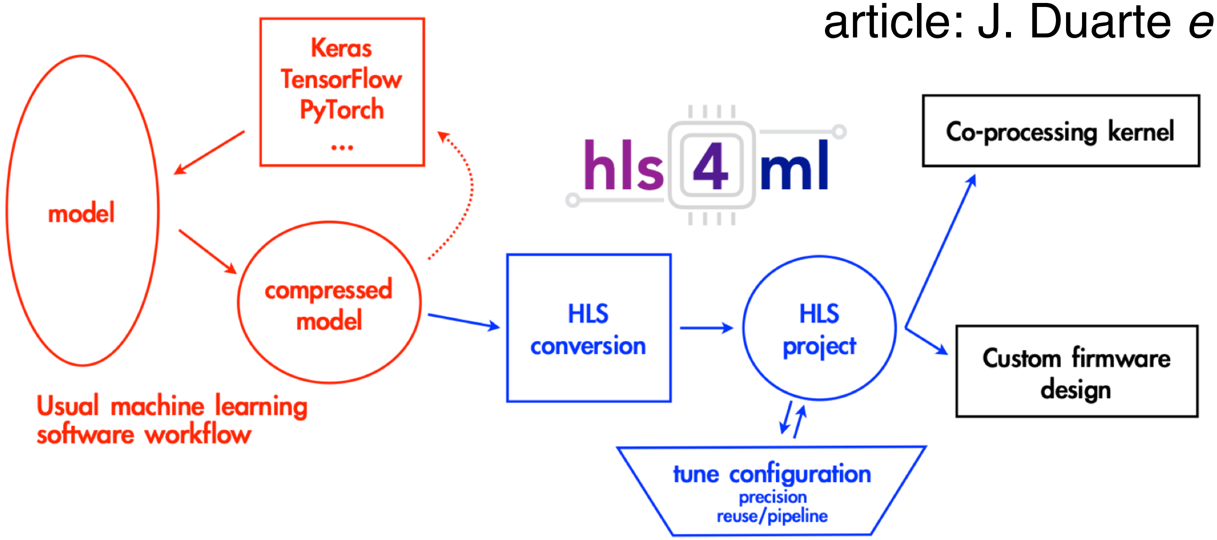


- ❑ *For data analysis we used a neural network library provided by ROOT /TMVA package:*
  - ➢ MultiLayerPerceptron (MLP)
- ❑ *Top left plot shows ionization difference for e/pi in several bins along the track*
- ❑ *Top right plot shows neural network output for single TRD module:*
  - ➢ Red - electrons with radiator
  - ➢ Blue – electrons without radiator.

Question 4 (cont) hls4ml package

- A package hls4ml is developed based on High-Level Synthesis (HLS) to build machine learning models in FPGAs.

article: J. Duarte *et al* 2018 *JINST* **13** P07027



model

Keras
TensorFlow
PyTorch
...

compressed model

Usual machine learning software workflow

hls 4 ml

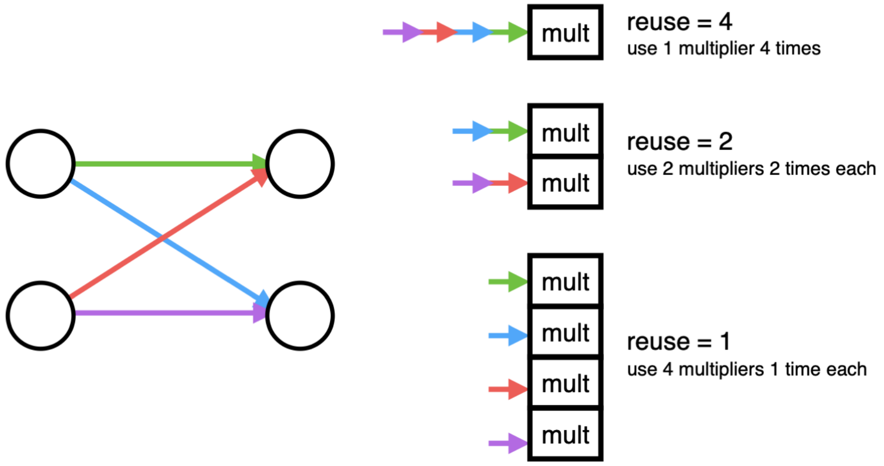
HLS conversion

HLS project

tune configuration
precision
reuse/pipeline

Co-processing kernel

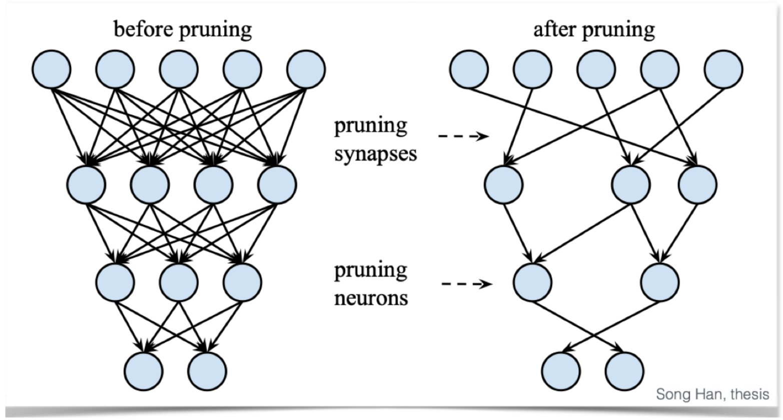
Custom firmware design



reuse = 4
use 1 multiplier 4 times

reuse = 2
use 2 multipliers 2 times each

reuse = 1
use 4 multipliers 1 time each



Song Han, thesis

Question 5

Please detail the planned work and deliverables from the electrical engineer, and how they fit into the current budget of 0.05+0.1 FTEs.

- ❑ *Provide firmware support for surrounding infrastructure*
  - ➢ Ethernet TCP/IP interface
  - ➢ Fiber optic serial interface
  - ➢ Event building, data storage
  - ➢ MicroBlaze interface
- ❑ *Assistance for design implementation and testing*
  - ➢ Device utilization
  - ➢ Preprocessing
  - ➢ Troubleshooting
  - ➢ Monitoring

Question 6

Regarding labor: How can a PhD student salary be cut by -20% or -40%? Is it planned to hire the student later? But then the problem is just shifted.

- *For this proposal, the graduate student could be cut back in one of two ways.*
  - The student's start could be delayed
  - Also, the student could be hired at a lower level of support from JLab and supplemented by a 25% LOE teaching assistantship.
  - The amount of time the student could devote to this work would be reduced which would also happen if there was delay in bringing on the student.

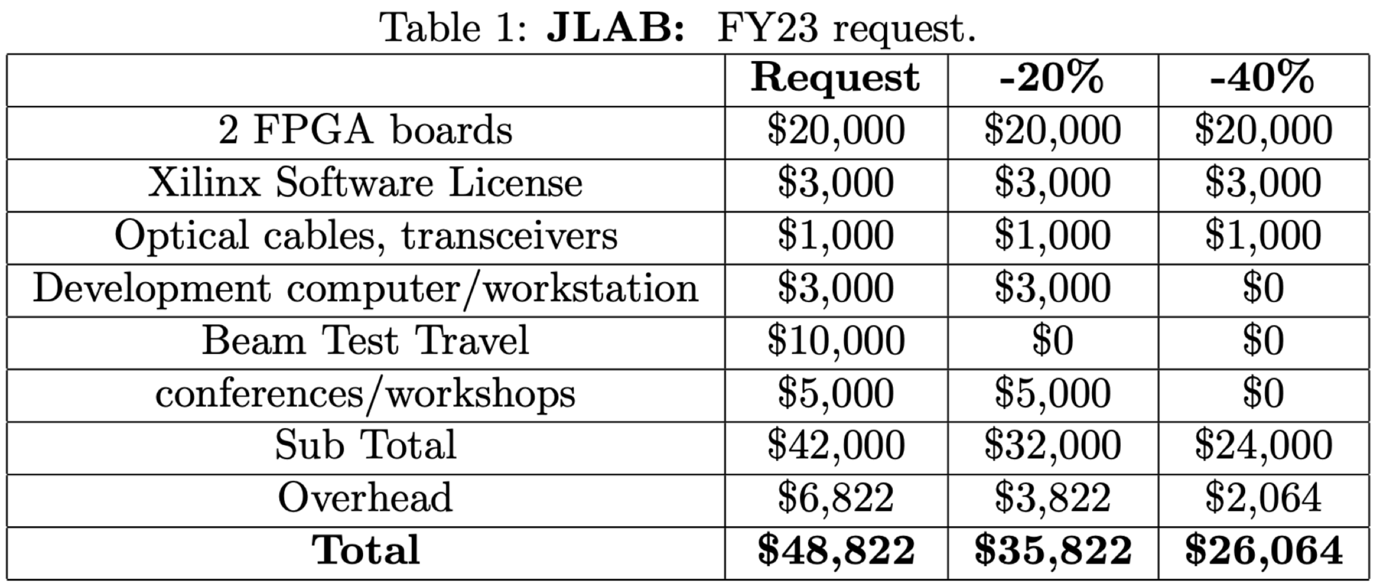


Table 1: **JLAB:** FY23 request.

	Request	**-20%**	**-40%**
2 FPGA boards	$20,000	$20,000	$20,000
Xilinx Software License	$3,000	$3,000	$3,000
Optical cables, transceivers	$1,000	$1,000	$1,000
Development computer/workstation	$3,000	$3,000	$0
Beam Test Travel	$10,000	$0	$0
conferences/workshops	$5,000	$5,000	$0
Sub Total	$42,000	$32,000	$24,000
Overhead	$6,822	$3,822	$2,064
Total	**$48,822**	**$35,822**	**$26,064**

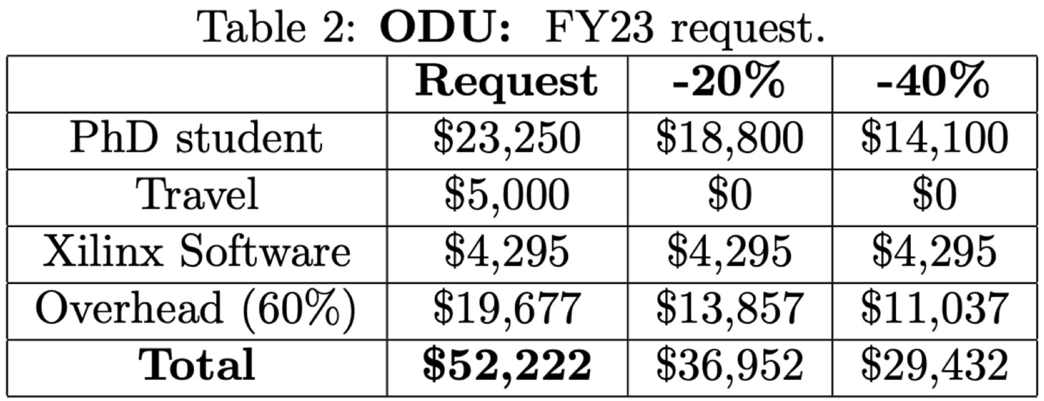


Table 2: **ODU:** FY23 request.

	Request	**-20%**	**-40%**
PhD student	$23,250	$18,800	$14,100
Travel	$5,000	$0	$0
Xilinx Software	$4,295	$4,295	$4,295
Overhead (60%)	$19,677	$13,857	$11,037
Total	**$52,222**	$36,952	$29,432

Question 7

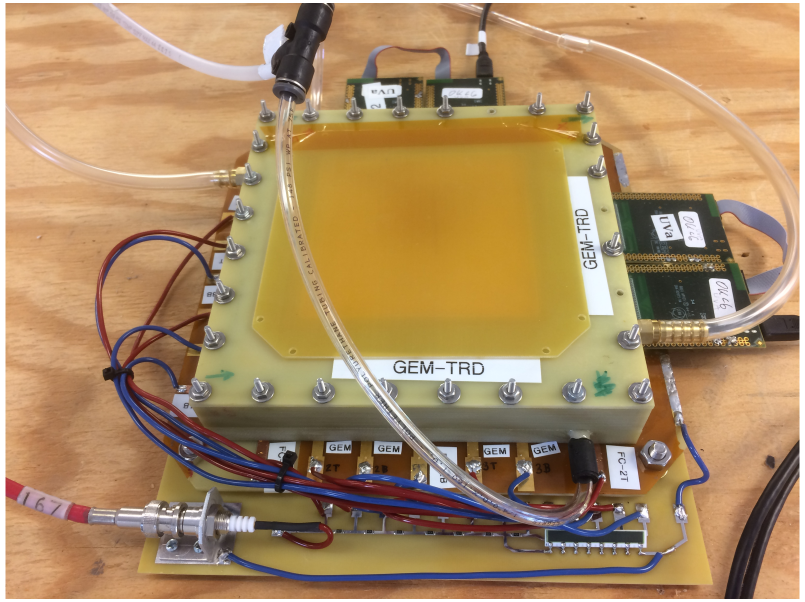
Please clarify to what extent the collaboration plans to be users of ML software vs developers. It sounds like the generic software/firmware work is essentially done by hls4ml (and others, but this one is mentioned specifically). But in the proposal, it seemed they would also be doing some development.

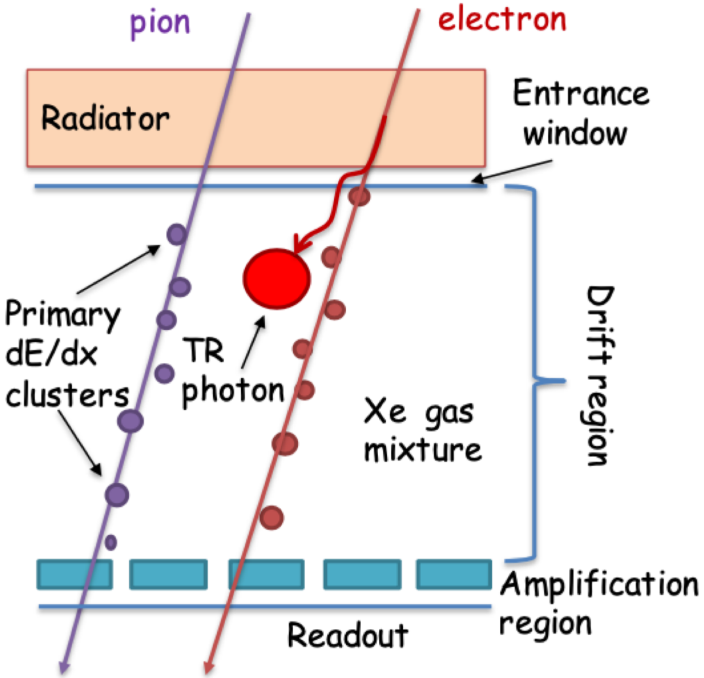
- ❑ *The key point of our proposal is to adapt and implement existing AI/ML algorithms in real FPGA hardware and performance test in a test beam with real prototype detectors for EIC. In Hall-d, we already have a beam line for test prototypes EIC detectors.*
- ❑ *However, we are also developing new machine learning methods for certain detectors (GEMTRD) and are participating in ML developments for other detectors (emCAL).*
- ❑ *While there are currently various software available to help convert algorithms from C++ to hardware description language, this is only part of the work that needs to be done. For a complete hardware implementation, special knowledge in FPGAs and electronics will be required.*
- ❑ *That is why this project is a multi-disciplinary endeavor between Physics, Electrical Engineering, and Computer scientist.*

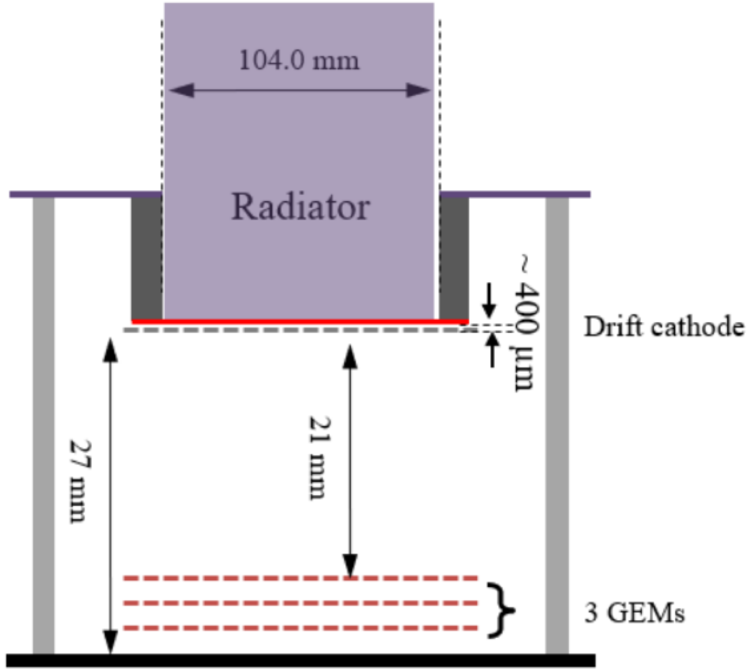
Backup

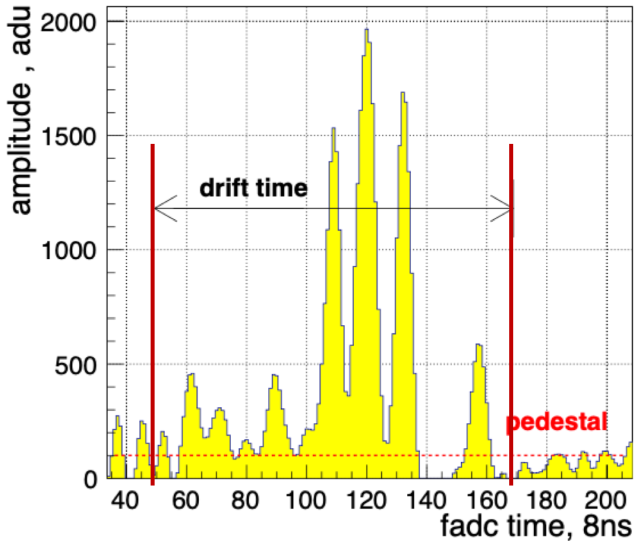
GEM-TRD prototype (eRD22) for EIC R&D

- To demonstrate the operating principle of the ML FPGA, we use the existing setup
- from the EIC detector R&D project (eRD22)
- A test module was built at the University of Virginia
- The prototype of GEMTRD/T module has a size of 10 cm × 10 cm with a corresponding to a total of 512 channels for X/Y coordinates.
- The readout is based on flash ADC system developed at JLAB (fADC125) @125 MHz sampling.
- GEM-TRD provides e/hadron separation and tracking





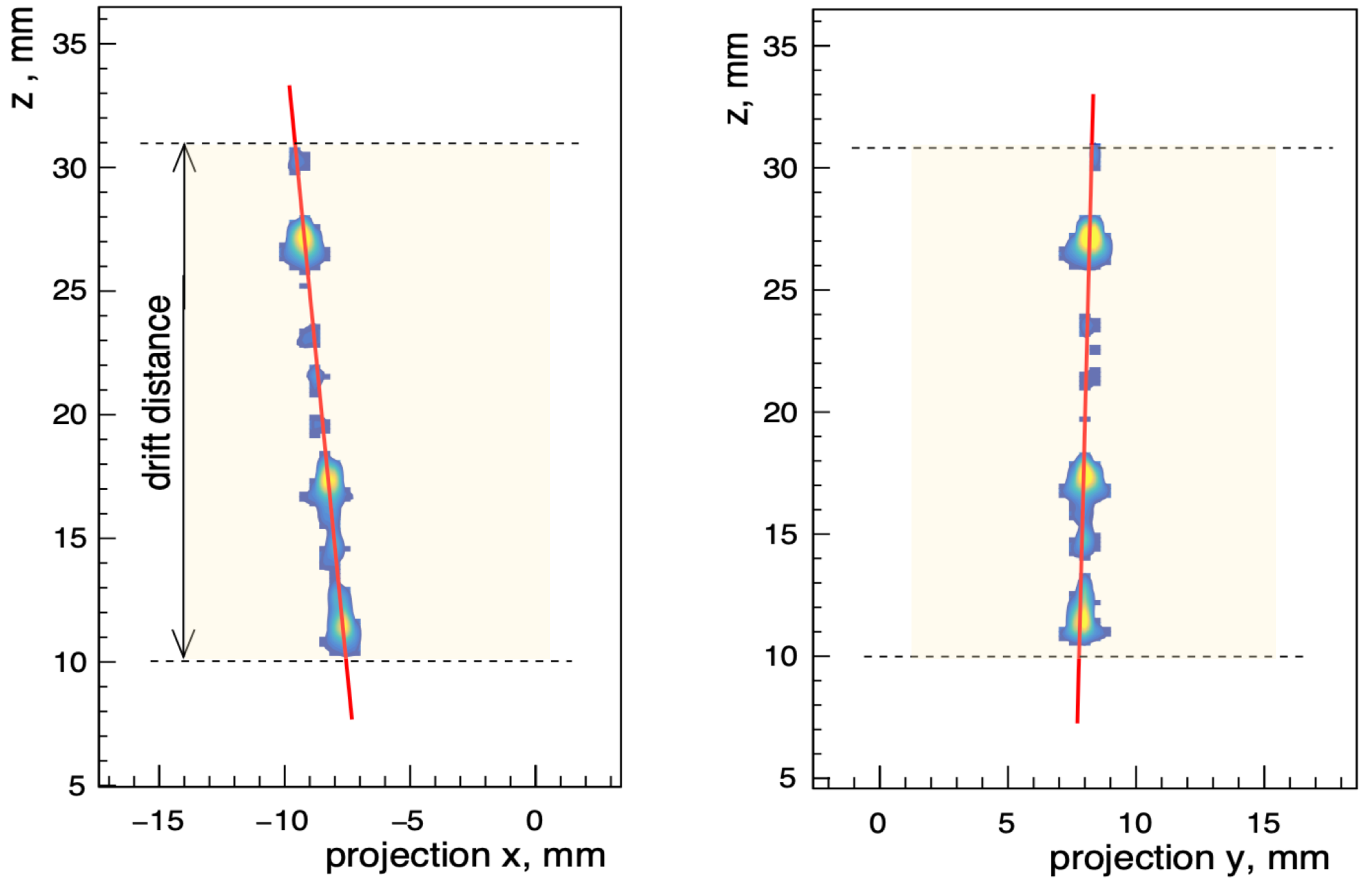




GEM-TRD principle

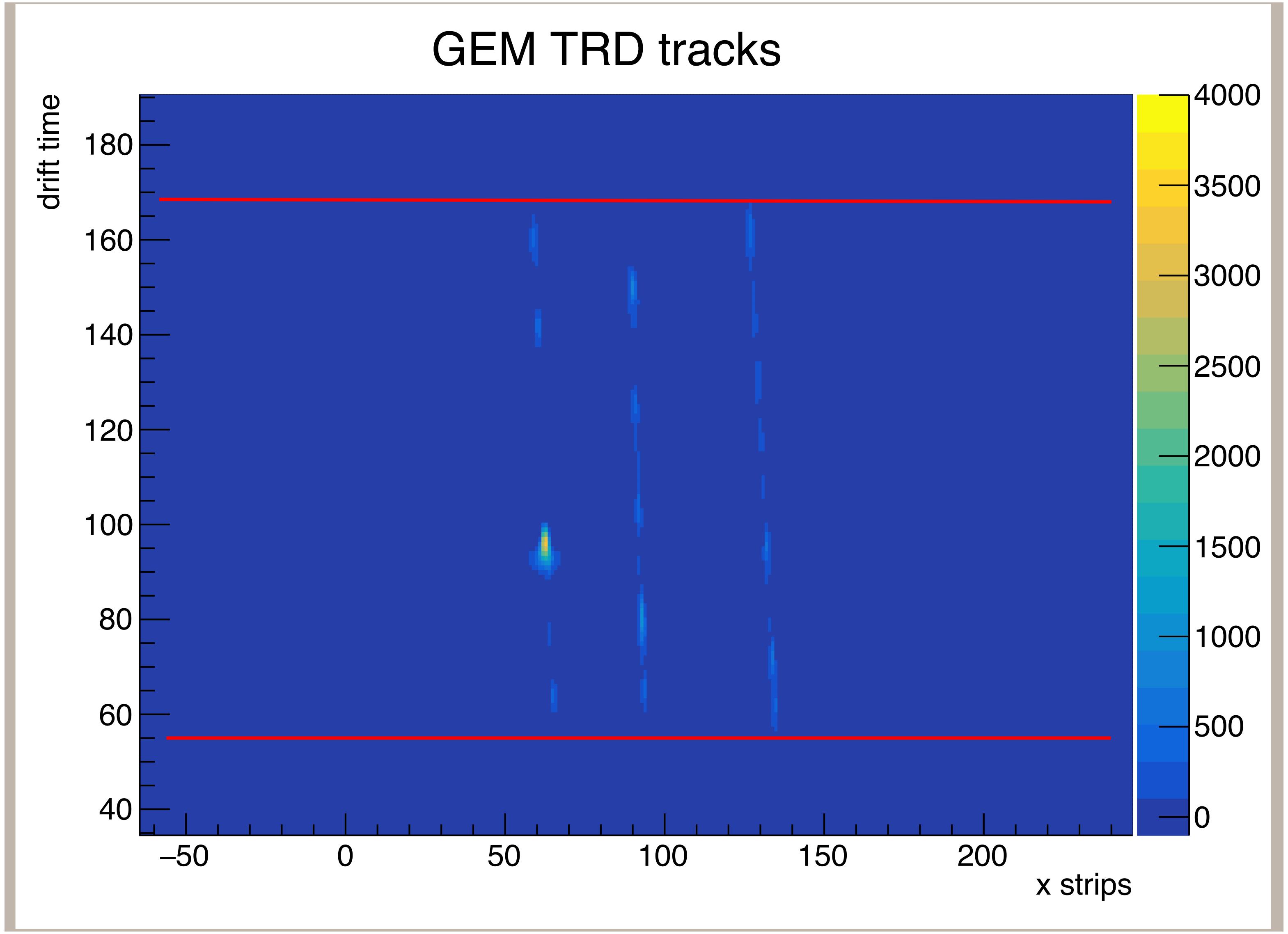
- ❑ *The e/pion separation in the GEM-TRD detector is based on counting the ionization along the particle track.*
- ❑ *For electrons, the ionization is higher due to the absorption of transition radiation photons*
- ❑ *So, particle identification with TRD consists of several steps:*
  - ➢ The first step is to cluster the incoming signals and create "hits".
  - ➢ The next is "pattern recognition" - sorting hits by track.
  - ➢ Finding a track
  - ➢ Ionization measurement along a track
  - ➢ As a bonus, TRD will provide a track segment for the global tracking system.

GEM-TRD can work as micro TPC, providing 3D track segments



GEM-TRD tracks

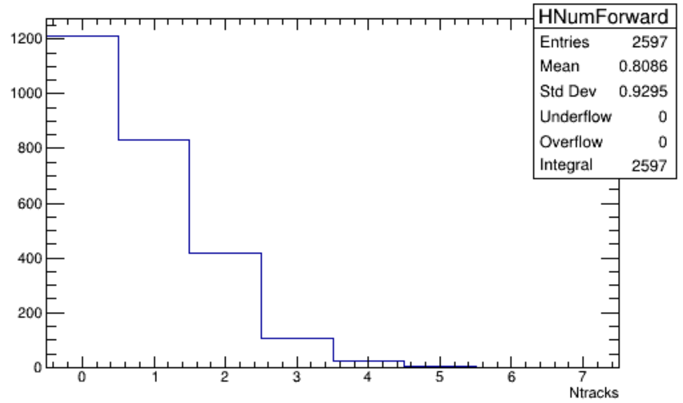
- ❑ *In a real experiment, GEMTRD will have multiple tracks.*
- ❑ *So we also need a fast algorithm for pattern recognition*
- ❑ *As well as for track fitting.*



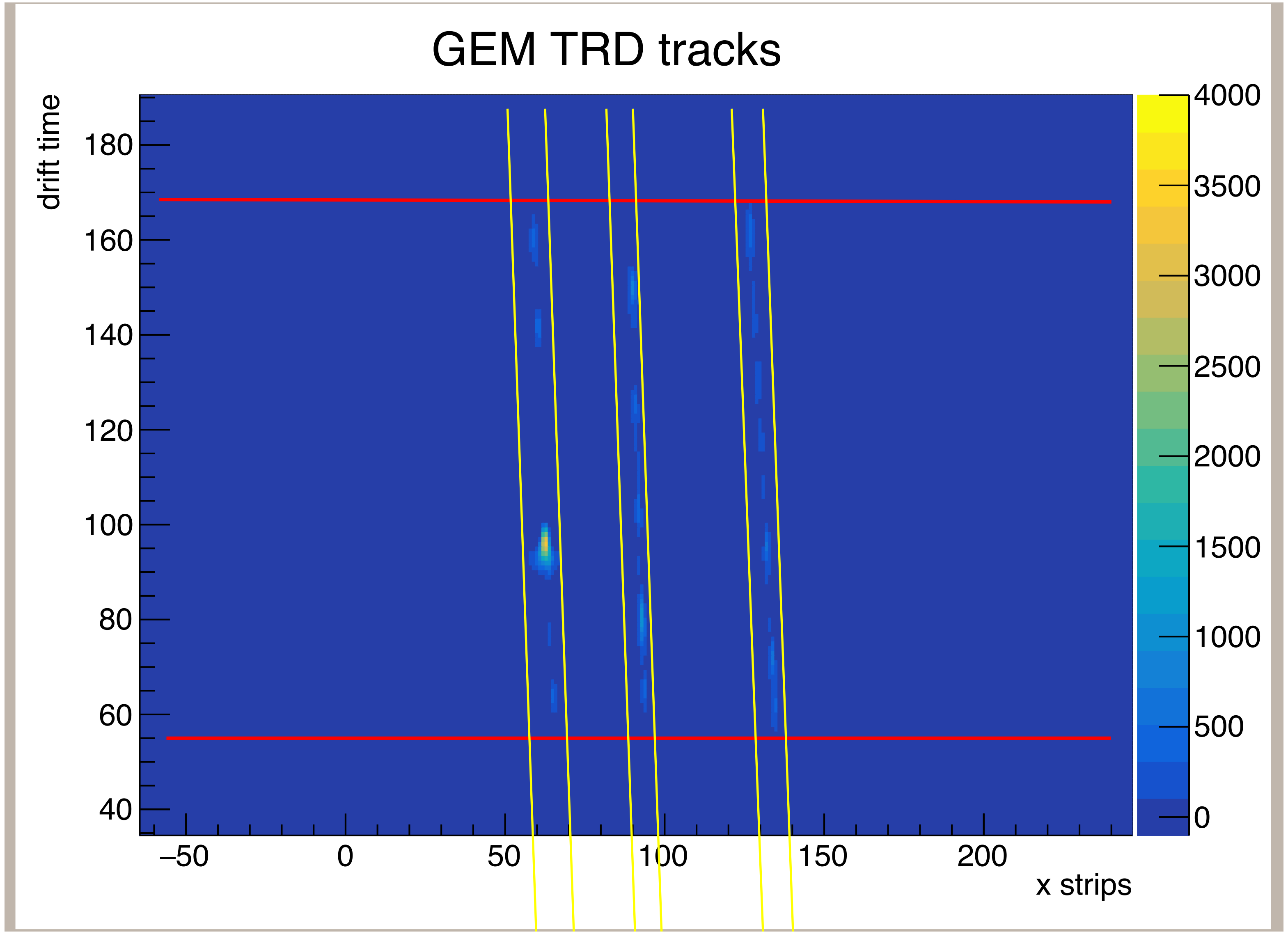
GEMTRD tracks

- ❑ *In a real experiment, GEMTRD will have multiple tracks.*
- ❑ *So we also need a fast algorithm for pattern recognition*
- ❑ *As well as for track fitting.*
- ❑ *The decision was made to try the Graph Neural Network (GNN) for pattern recognition.*
- ❑ *And a recurrent neural network – LSTM, for track fitting.*

Number of tracks in forward region in GlueX experiment



HNumForward	
Entries	2597
Mean	0.8086
Std Dev	0.9295
Underflow	0
Overflow	0
Integral	2597



Existing GNN tracking projects

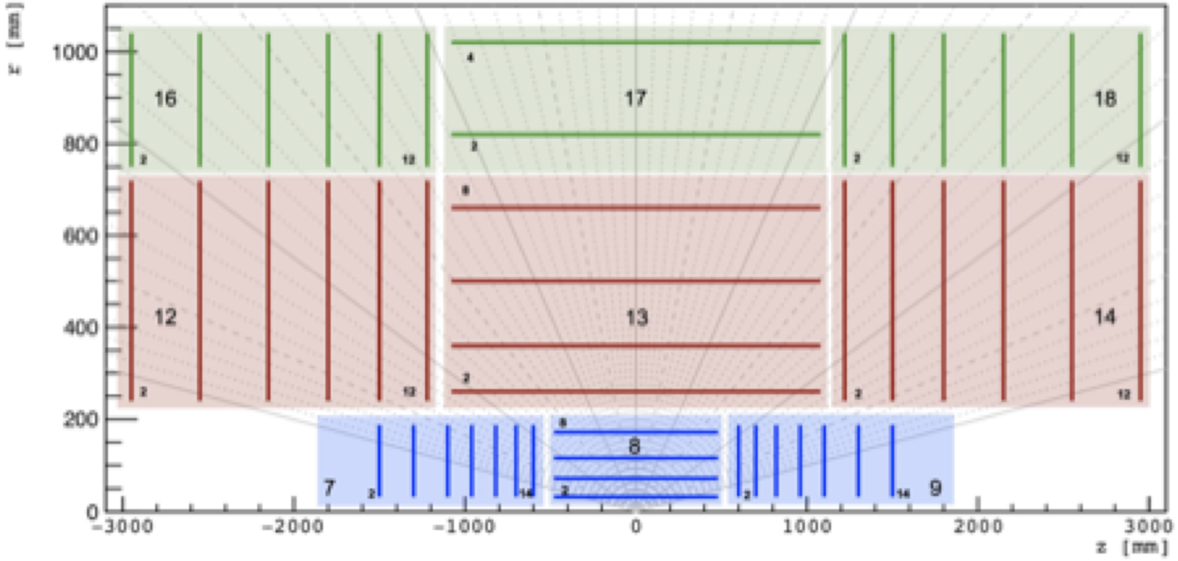
❑ **TrackML Dataset**

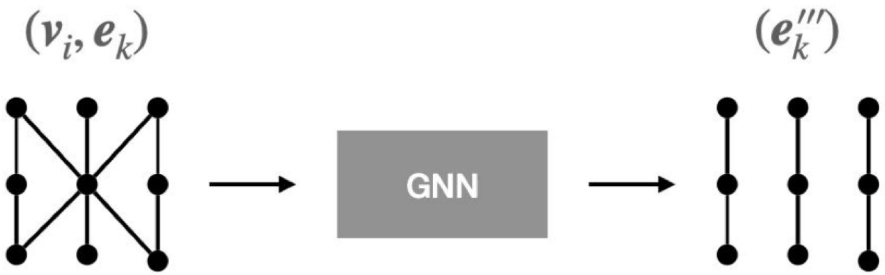
Public dataset hosted on Kaggle for particle tracking: https://www.kaggle.com/c/trackml-particle-identification

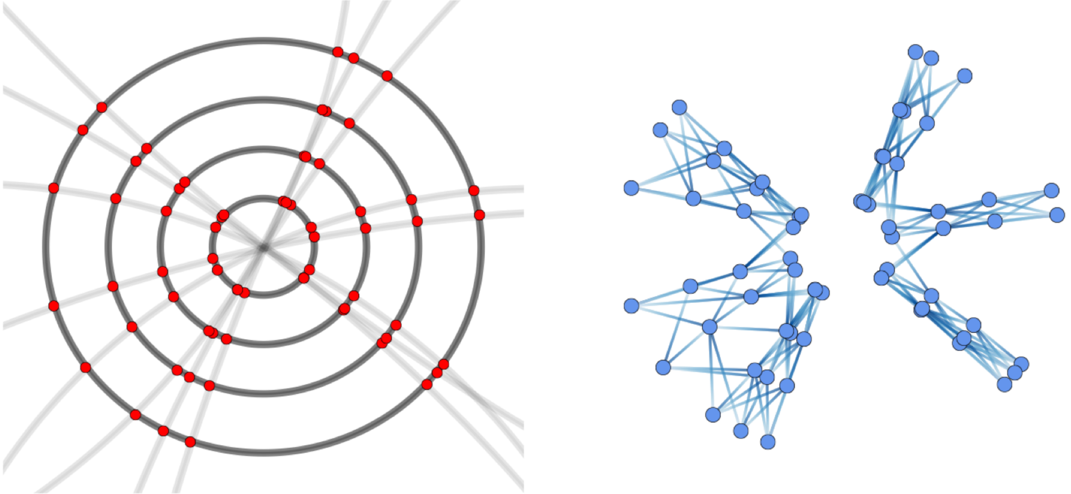
❑ HEP advanced tracking algorithms at the exascale **(Project Exa.TrkX)**

❑ https://exatrkx.github.io/

❑ https://github.com/jmduarte/exatrkx-neurips19/tree/master/gnn-tracking







Javier Duarte arXiv:2012.01249v2 [hep-ph] 7 Dec 2020

So we decided to start by evaluating an Exa.TrkX solution

Moving forward : ML on FPGA

- *Offline analysis using ML looks promising.*
- *Can it be done in real time ?*
- *Here are some of the possible solutions :*
  - ➢ Computer farm.
  - ➢ CPU + GPU
  - ➢ CPU + FPGA
  - ➢ FPGA only

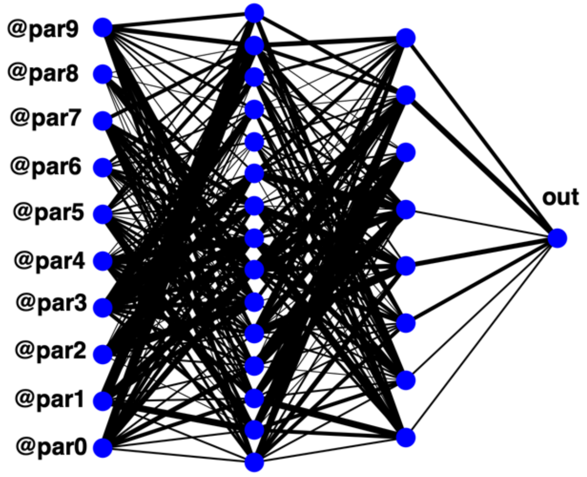
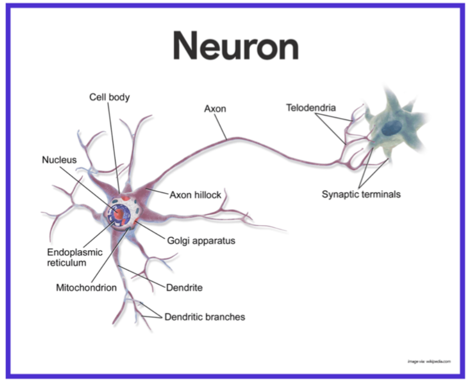
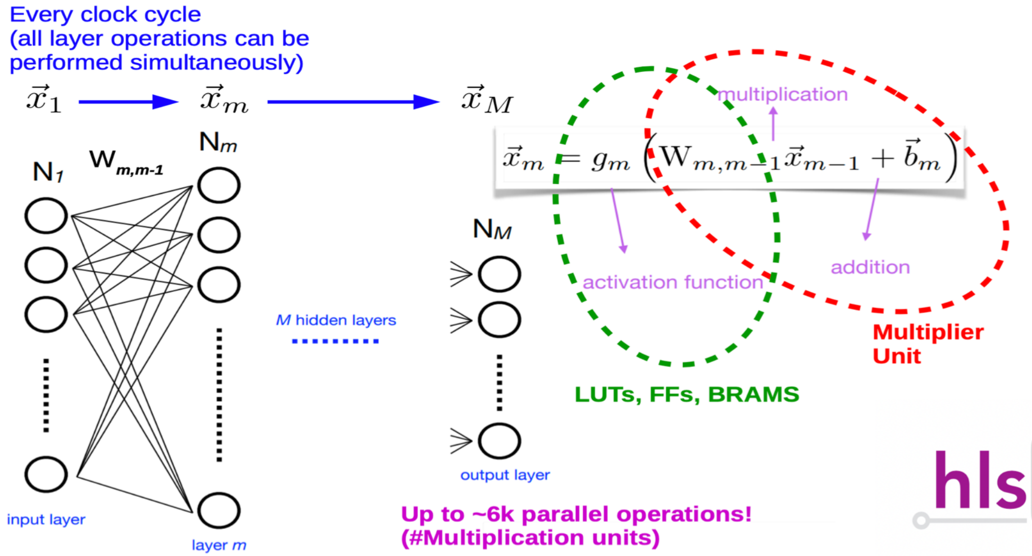


Image: https://nurseslabs.com/nervous-system/



Inference on an FPGA



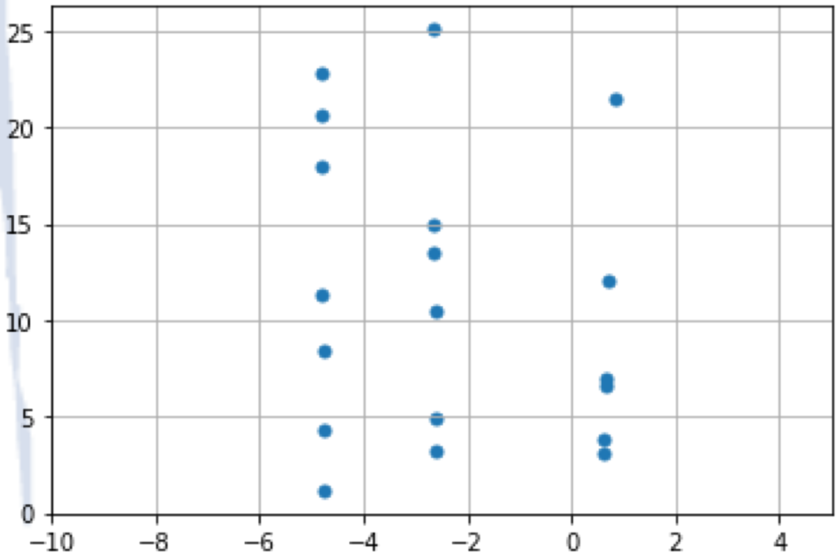
$$\vec{x}_m = g_m\left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m\right)$$

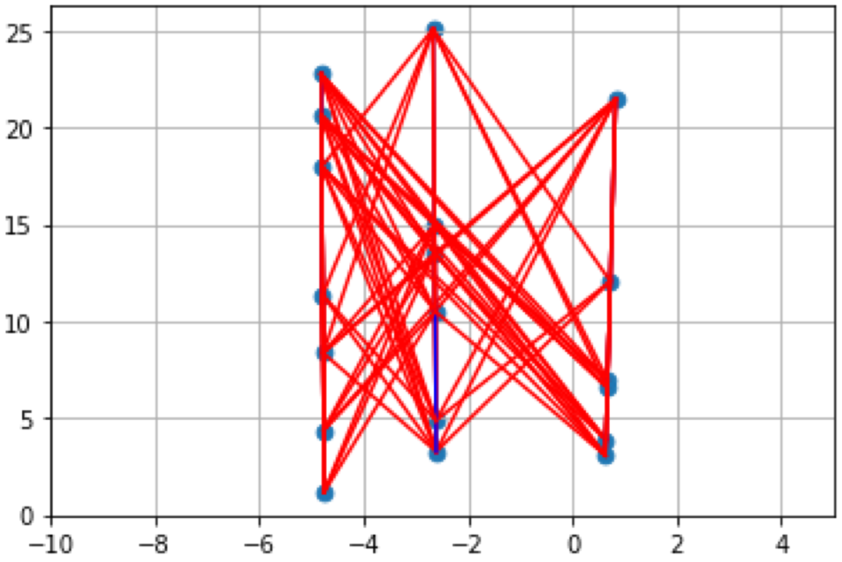
IRIS-HEP th Febraury 13 , 2019 Dylan Rankin [MIT]

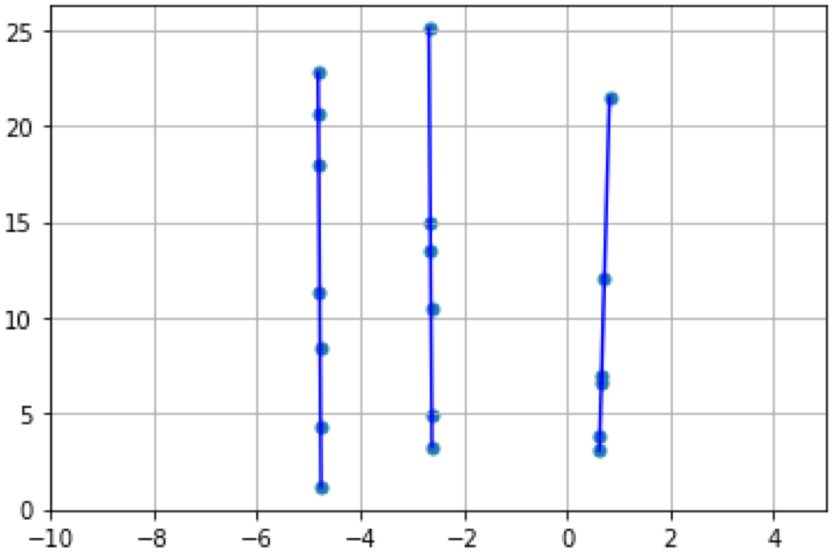
- Modern FPGAs have DSP slices - specialized hardware blocks placed between gateways and routers that perform mathematical calculations.
- The number of DSP slices can be up to 6000-12000 per chip.

GNN for pattern recognition

- ❑ *Graph Neural Networks (GNNs) designed for the tasks of hit classification and segment classification.*
  - ➢ These models read a graph of connected hits and compute features on the nodes and edges.
- ❑ *The input and output of GNN is a graph with a number of features for nodes and edges.*
  - ➢ In our case we use the edge classification
- ❑ *A complete graph on N vertices contains N(N - 1)/2 edges.*
  - ➢ This will require a lot of resources which are limited in FPGA.
- ❑ *To keep resources under control, we can construct the graph for a specific geometry and limit the minimum particle momentum.*
- ❑ *In our case we have a straight track segments, with a quite narrow angular distribution ~15 degree.*
- ❑ *Thus, for the input hits (left), we connect only those edges that satisfy our geometry and the momentum of most tracks (middle)*
- ❑ *The trained GNN processes the input graph and sets the probability for each edge as output.*
- ❑ *The right plot shows edges with a probability greater than 0.7*

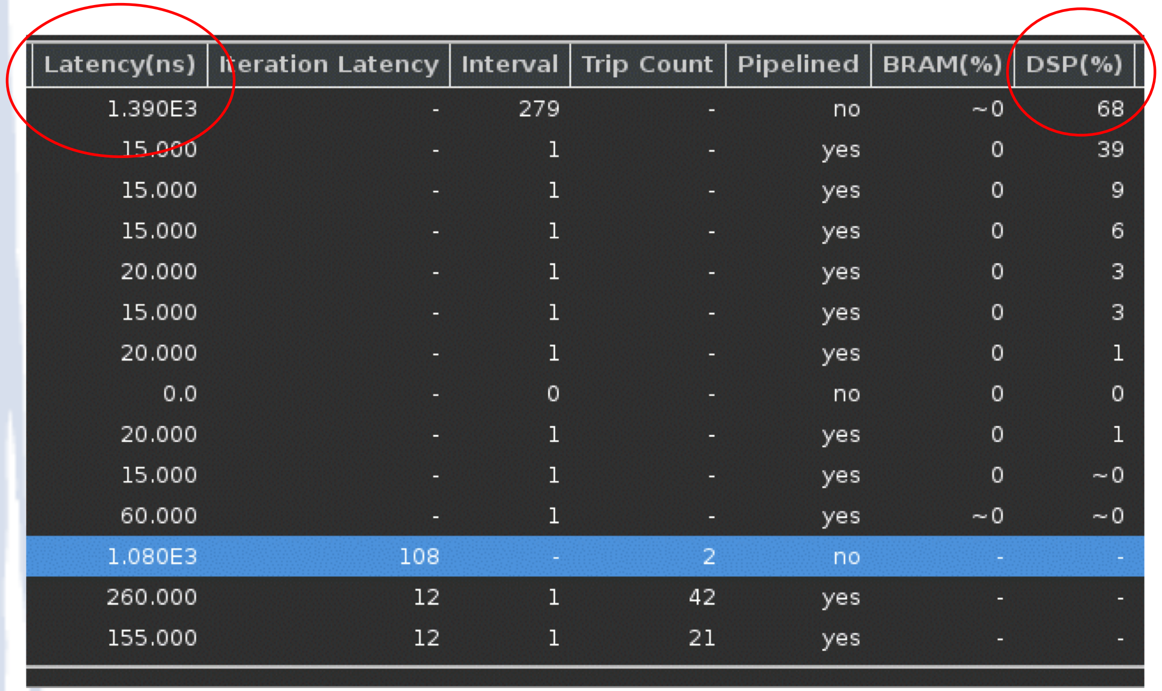




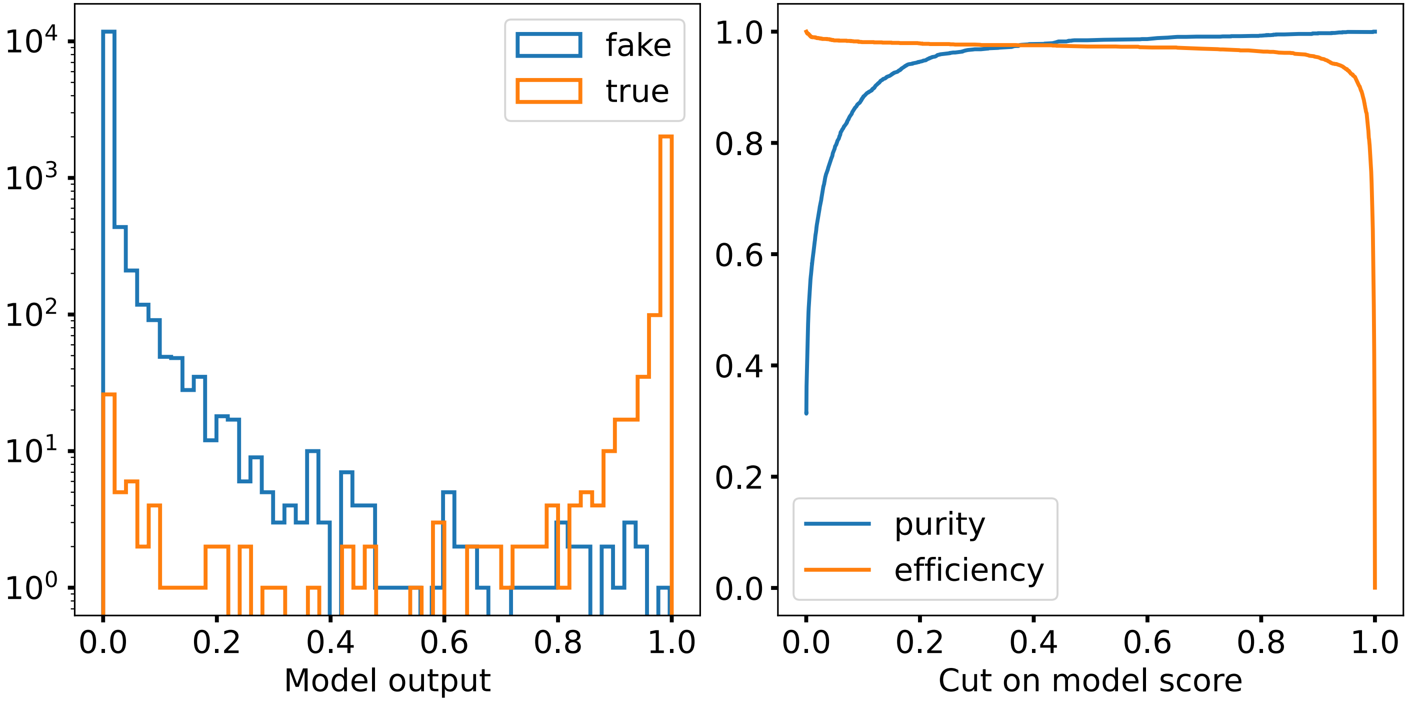


GNN performance

- ❑ *This type of graph neural network is not yet supported in HLS4ML.*
- ❑ *So we did a manual conversion first to C++ and then to Verilog using Vitis_HLS.*
- ❑ *This neural network has not been optimized, so it consumes a lot of resources - 70% of DSPs, (4651 of 6840).*
  - ➢ At the moment it can serve up to 21 hits and 42 edges, or , in our case (GEM-TRD), it will be 3-4 tracks.
- ❑ *However, it performs all calculations in 1.4 µs (left plot) (thanks to Ben Raydo), providing good purity and efficiency (right plot).*



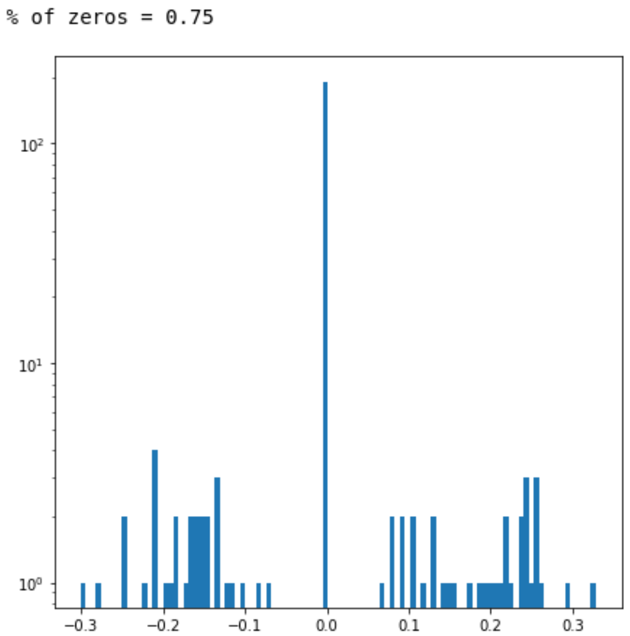
Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM(%)	DSP(%)
1.390E3	-	279	-	no	~0	68
15.000	-	1	-	yes	0	39
15.000	-	1	-	yes	0	9
15.000	-	1	-	yes	0	6
20.000	-	1	-	yes	0	3
15.000	-	1	-	yes	0	3
20.000	-	1	-	yes	0	1
0.0	-	0	-	no	0	0
20.000	-	1	-	yes	0	1
15.000	-	1	-	yes	0	~0
60.000	-	1	-	yes	~0	~0
1.080E3	108	-	2	no	-	-
260.000	12	1	42	yes	-	-
155.000	12	1	21	yes	-	-

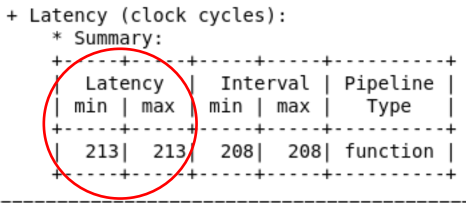


RNN/LSTM for track fit

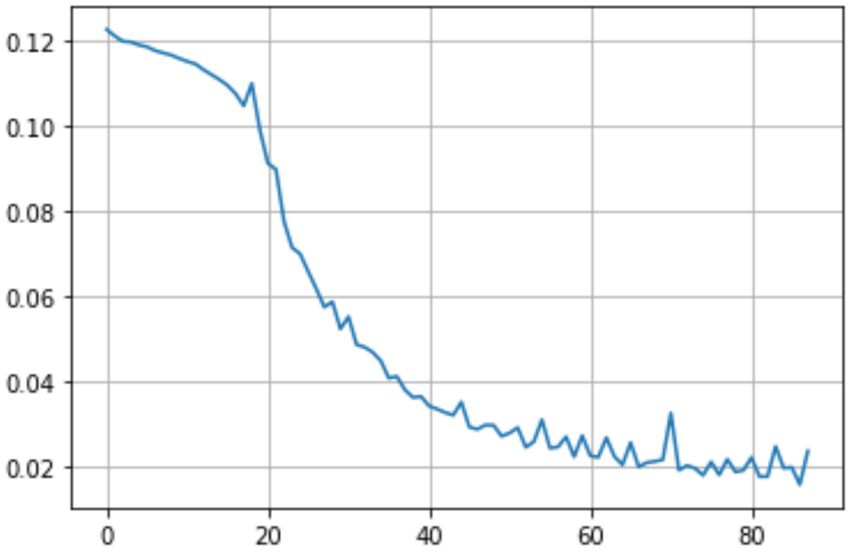
Jefferson Lab
Thomas Jefferson National Accelerator Facility

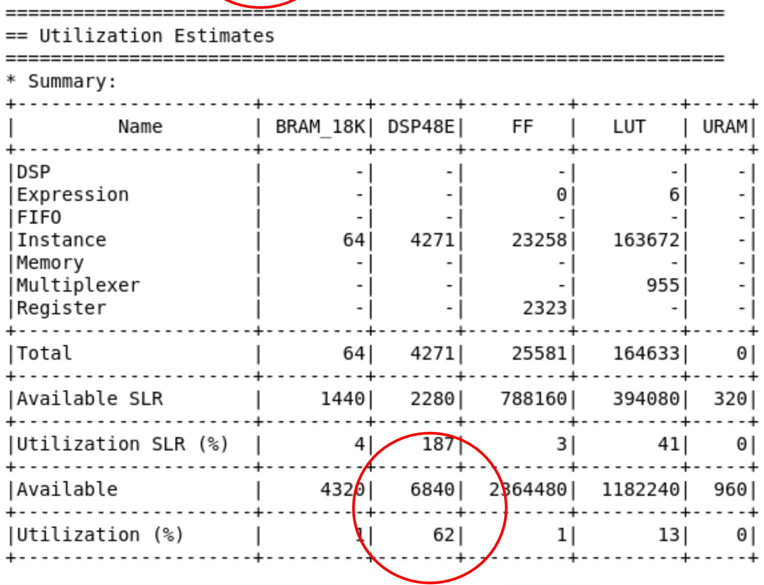
- ❑ *The hits sorted by tracks from the pattern recognition GNN are fed into another neural network trained to fit the tracks.*
- ❑ *We tested DNN and RNN/LSTM neural networks. (thanks to Dylan Rankin for help)*
- ❑ *DNN is faster, but LSTM seems to be more reliable in the case of a stochastic distribution of hits on the track.*
  - ➢ The work on optimization of NN is ongoing.
- ❑ *The LSTM network after pruning consumes 19% of the DSP resources and has a latency of 1 μs.*



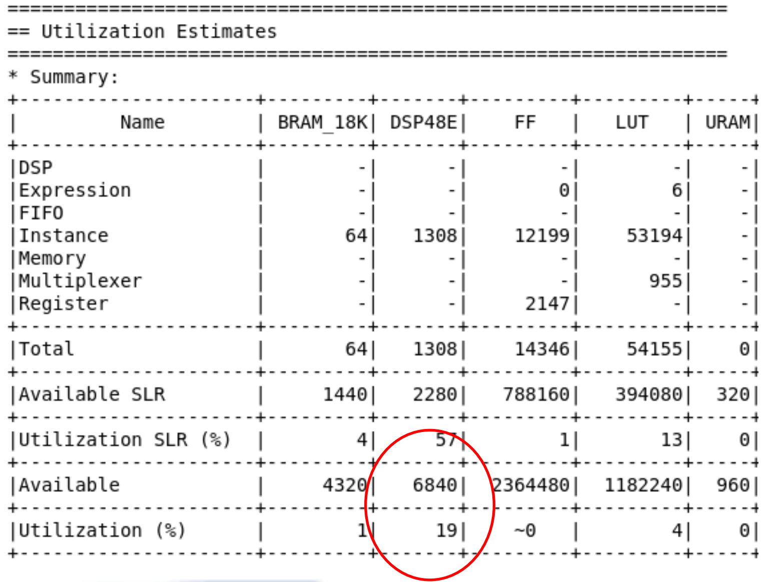


```
+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+----------+
    |  Latency  |  Interval | Pipeline |
    | min | max | min | max |   Type   |
    +-----+-----+-----+-----+----------+
    |  213|  213|  208|  208| function |
    +-----+-----+-----+-----+----------+
```





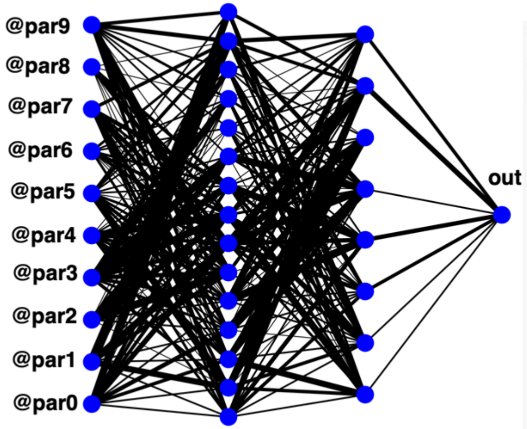
```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|        6|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |       64|   4271|    23258|   163672|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|      955|    -|
|Register             |        -|      -|     2323|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |       64|   4271|    25581|   164633|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        4|    187|        3|       41|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        1|     62|        1|       13|    0|
+---------------------+---------+-------+---------+---------+-----+
```

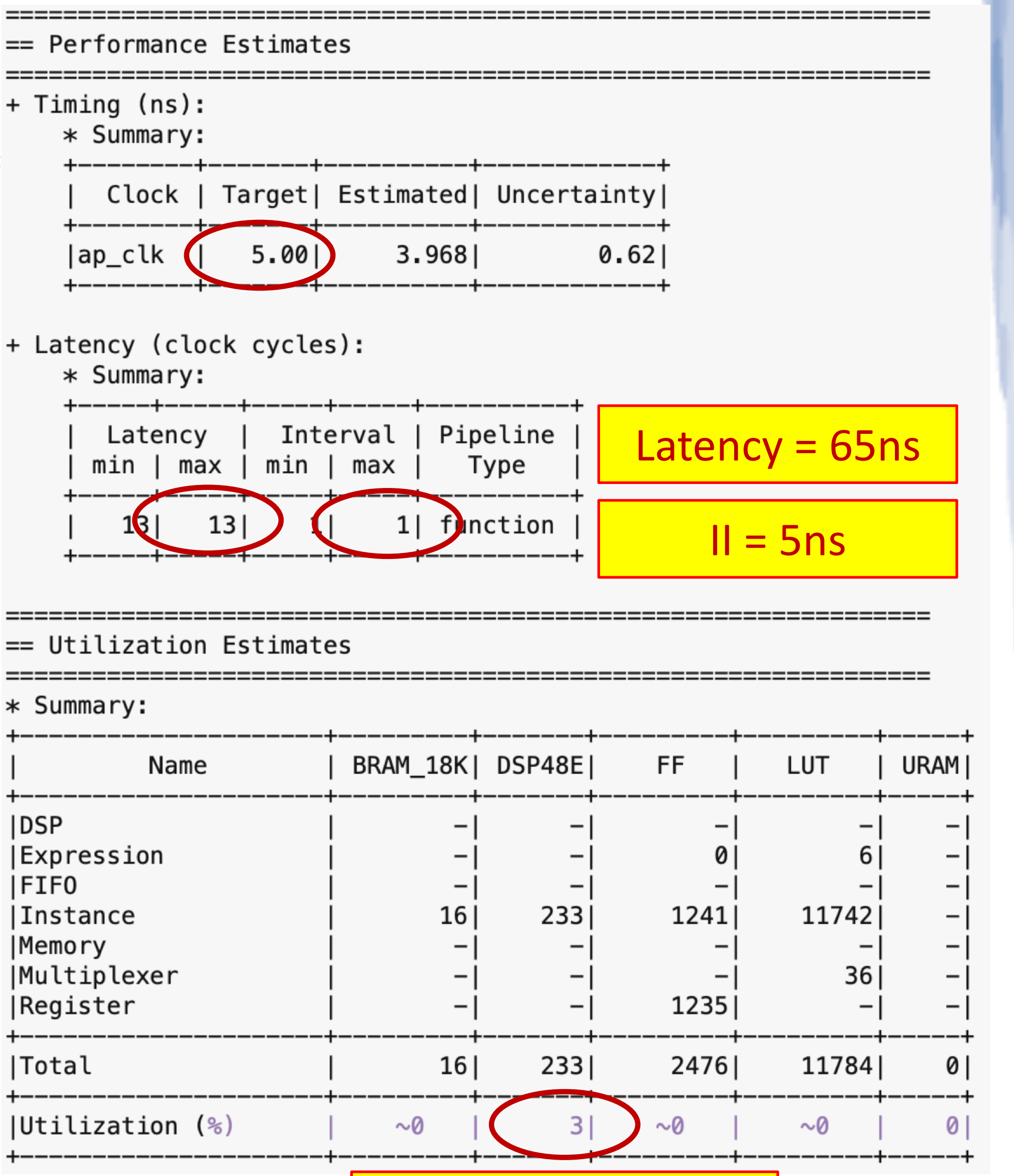


```
================================================================
== Utilization Estimates
================================================================
* Summary:
+---------------------+---------+-------+---------+---------+-----+
|         Name        | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+---------------------+---------+-------+---------+---------+-----+
|DSP                  |        -|      -|        -|        -|    -|
|Expression           |        -|      -|        0|        6|    -|
|FIFO                 |        -|      -|        -|        -|    -|
|Instance             |       64|   1308|    12199|    53194|    -|
|Memory               |        -|      -|        -|        -|    -|
|Multiplexer          |        -|      -|        -|      955|    -|
|Register             |        -|      -|     2147|        -|    -|
+---------------------+---------+-------+---------+---------+-----+
|Total                |       64|   1308|    14346|    54155|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available SLR        |     1440|   2280|   788160|   394080|  320|
+---------------------+---------+-------+---------+---------+-----+
|Utilization SLR (%)  |        4|     57|        1|       13|    0|
+---------------------+---------+-------+---------+---------+-----+
|Available            |     4320|   6840|  2364480|  1182240|  960|
+---------------------+---------+-------+---------+---------+-----+
|Utilization (%)      |        1|     19|   ~0    |        4|    0|
+---------------------+---------+-------+---------+---------+-----+
```

MLP neural network for PID

- *After the track is fit, the ionization along the track can be counted.*
- *The distance along the track is divided into 10-20 bins, and the ionization energy in these bins is fed to the input of the MLP neural network.*
- *Typically neural network weights often have many zeros, thus, it is possible to reduce the size of the network by removing weights close to zero (~50%)*
- *The network performance near the working value of 90% efficiency.*

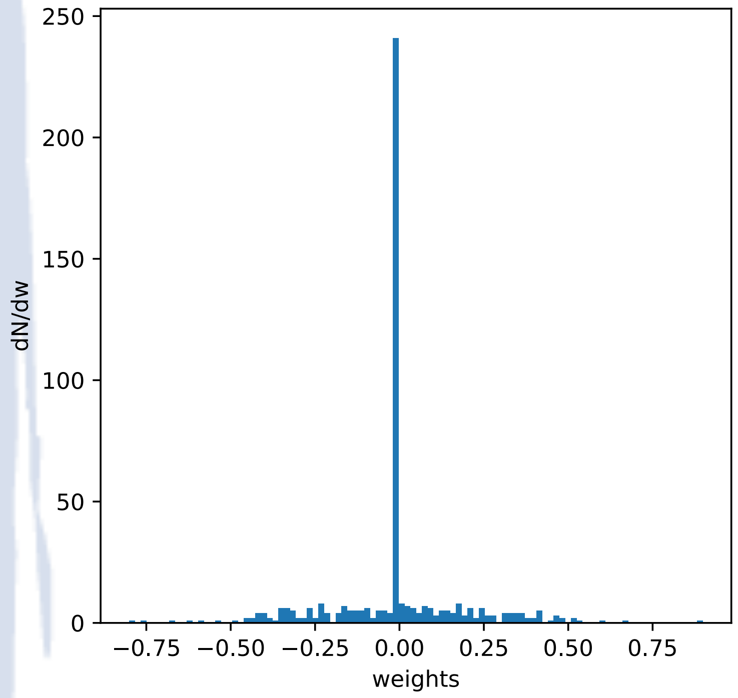


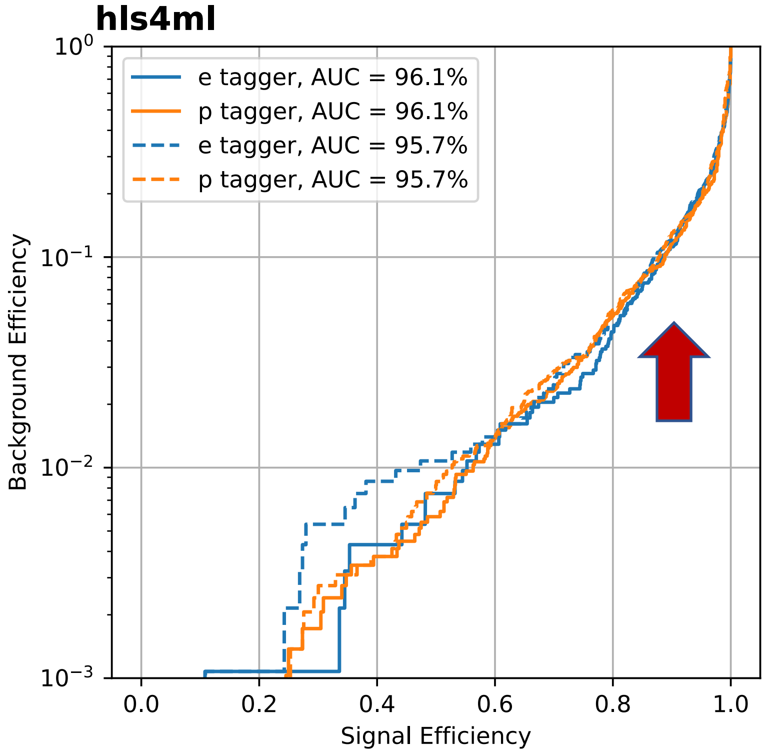


```
================================================================
== Performance Estimates
================================================================
+ Timing (ns):
    * Summary:
    +--------+--------+---------+-------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+--------+---------+-------------+
    |ap_clk  |    5.00|     3.968|        0.62|
    +--------+--------+---------+-------------+

+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+-----------+
    |  Latency  |  Interval | Pipeline |
    | min | max | min | max |   Type   |
    +-----+-----+-----+-----+-----------+
    |   13|   13|    1|    1| function |
    +-----+-----+-----+-----+-----------+

================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+--------+--------+-----+
|       Name      | BRAM_18K| DSP48E|   FF   |   LUT  | URAM|
+-----------------+---------+-------+--------+--------+-----+
|DSP              |        -|      -|       -|       -|    -|
|Expression       |        -|      -|       0|       6|    -|
|FIFO             |        -|      -|       -|       -|    -|
|Instance         |       16|    233|    1241|   11742|    -|
|Memory           |        -|      -|       -|       -|    -|
|Multiplexer      |        -|      -|       -|      36|    -|
|Register         |        -|      -|    1235|       -|    -|
+-----------------+---------+-------+--------+--------+-----+
|Total            |       16|    233|    2476|   11784|    0|
+-----------------+---------+-------+--------+--------+-----+
|Utilization (%)  |       ~0|      3|      ~0|      ~0|    0|
+-----------------+---------+-------+--------+--------+-----+
```

Latency = 65ns

II = 5ns





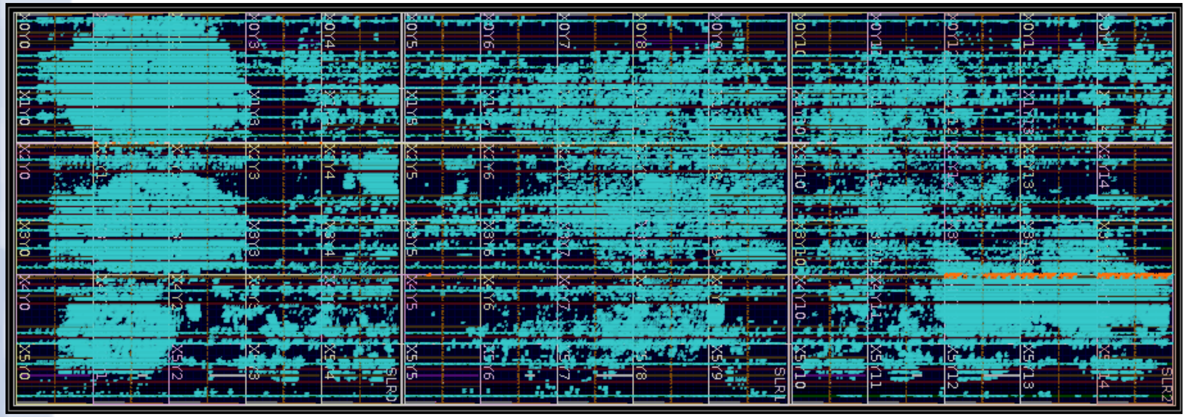
DSP utilization 3%

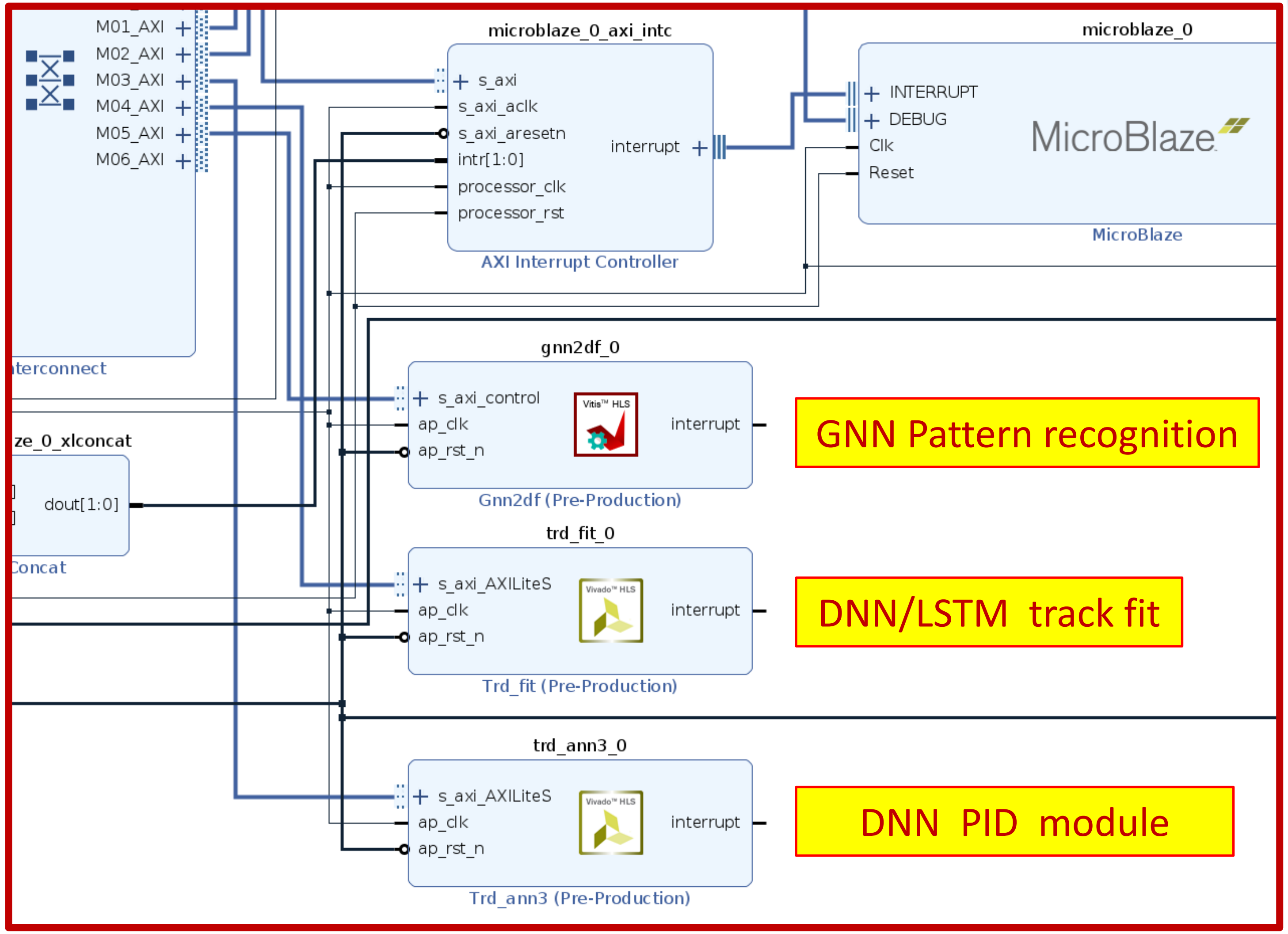
FPGA test bench

- *Several version of IPs were synthesized and tested on FPGAs.*
- *The logic test was performed with the MicroBlaze processor and the AXI Lite interface.*
- *We are currently working on a fast I/O interface to get data directly from the detector..*

FPGA IP SYNTHESIS SUMMARY.

	GNN	LSTM	DNN	CNN	GarNet
Clock, ns	5	5	5	5	5
Latency, clocks	278	239	13	260	5643
Interval, clocks	279	234	1	245	5643
Latency, ns	1390	1195	65	1300	23215
Utilization DSP (%)	68	27	3	71	3





GNN Pattern recognition

DNN/LSTM track fit

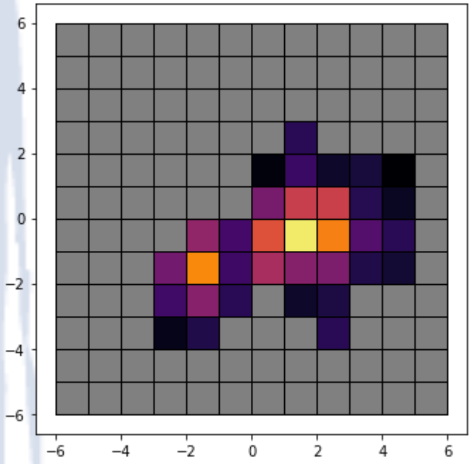
DNN PID module

CNN for calorimeter reconstruction

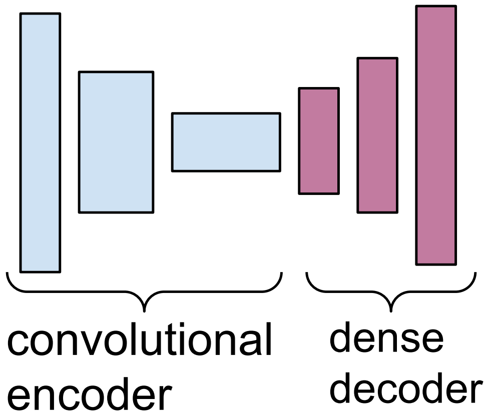
Jefferson Lab
Thomas Jefferson National Accelerator Facility

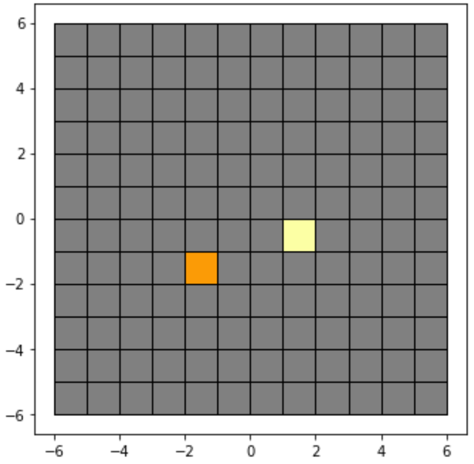
- *In this work we used a convolutional encoder with a decoder consisting of dense layers, which provide e-π separation scores as the output.*
- *This was done to minimize a network size in FPGA and due to current limitation of HSL4ML of supported network layer types.*
- *FPGA synthesis with reuse factor of 2 has a latency of 1.3μs and an interval of 245 clocks. It uses 71% of DPS resources*

Actual values	Predicted results	
	e	π
e	98.8 %	1.2 %
π	2.9 %	97.1 %

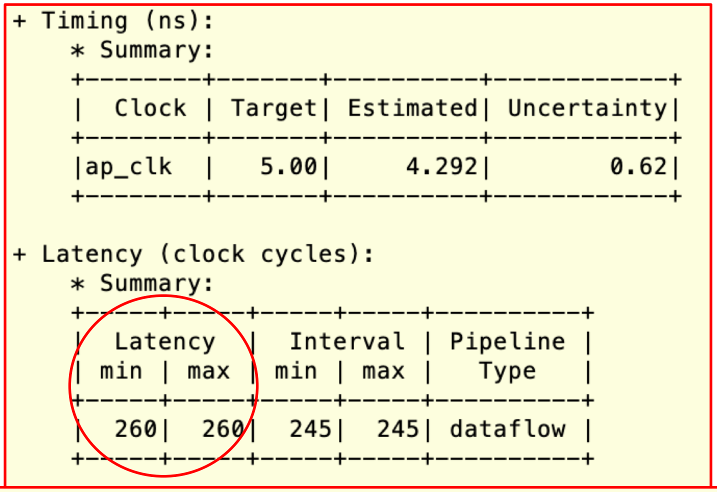


Input data

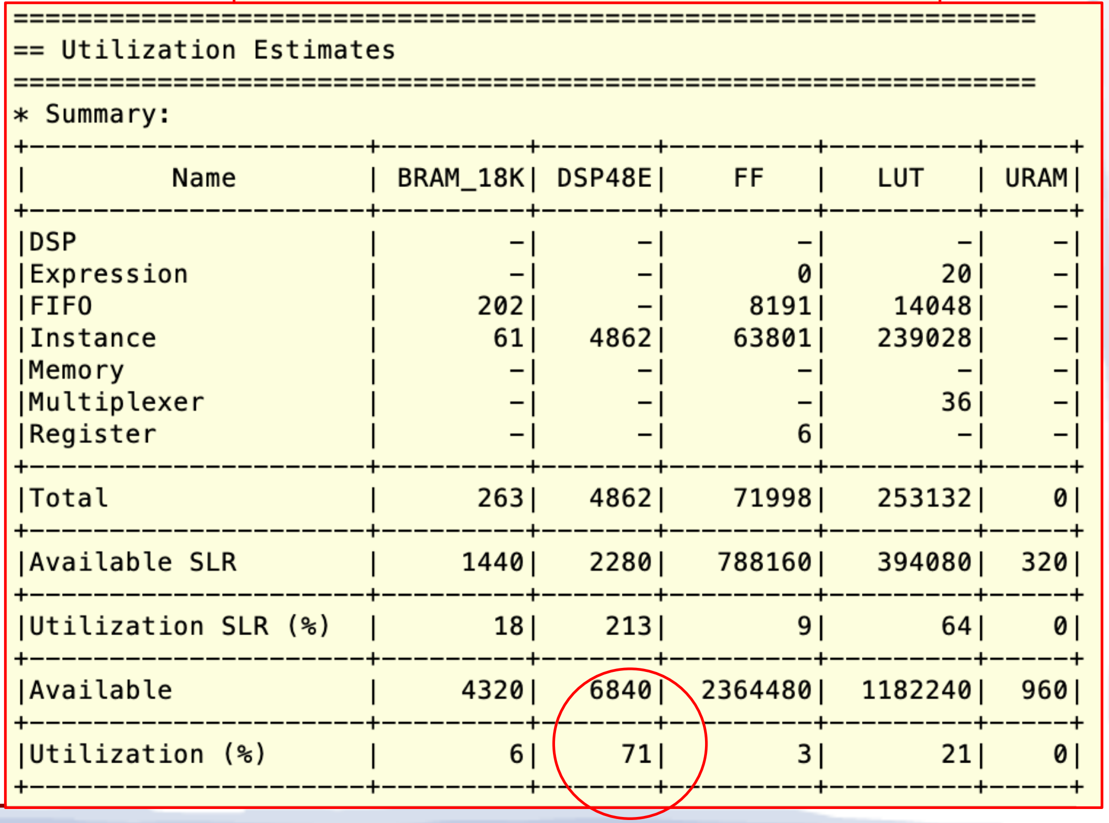




output



```
+ Timing (ns):
    * Summary:
    +--------+-------+----------+------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+----------+------------+
    |ap_clk  |   5.00|     4.292|        0.62|
    +--------+-------+----------+------------+

+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+----------+
    |  Latency  |  Interval | Pipeline |
    | min | max | min | max |   Type   |
    +-----+-----+-----+-----+----------+
    |  260|  260|  245|  245| dataflow |
    +-----+-----+-----+-----+----------+
```



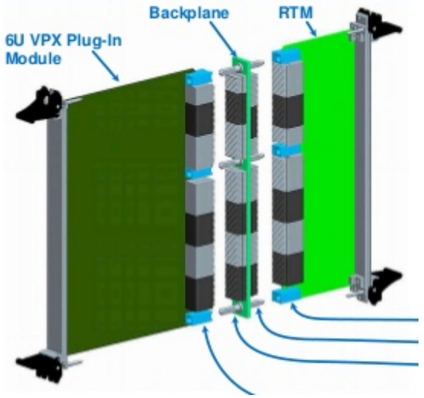
```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------------+---------+-------+---------+--------+-----+
|          Name         | BRAM_18K| DSP48E|    FF   |   LUT  | URAM|
+-----------------------+---------+-------+---------+--------+-----+
|DSP                    |        -|      -|        -|       -|    -|
|Expression             |        -|      -|        0|      20|    -|
|FIFO                   |      202|      -|     8191|   14048|    -|
|Instance               |       61|   4862|    63801|  239028|    -|
|Memory                 |        -|      -|        -|       -|    -|
|Multiplexer            |        -|      -|        -|      36|    -|
|Register               |        -|      -|        6|       -|    -|
+-----------------------+---------+-------+---------+--------+-----+
|Total                  |      263|   4862|    71998|  253132|    0|
+-----------------------+---------+-------+---------+--------+-----+
|Available SLR          |     1440|   2280|   788160|  394080|  320|
+-----------------------+---------+-------+---------+--------+-----+
|Utilization SLR (%)    |       18|    213|        9|      64|    0|
+-----------------------+---------+-------+---------+--------+-----+
|Available              |     4320|   6840|  2364480| 1182240|  960|
+-----------------------+---------+-------+---------+--------+-----+
|Utilization (%)        |        6|     71|        3|      21|    0|
+-----------------------+---------+-------+---------+--------+-----+
```

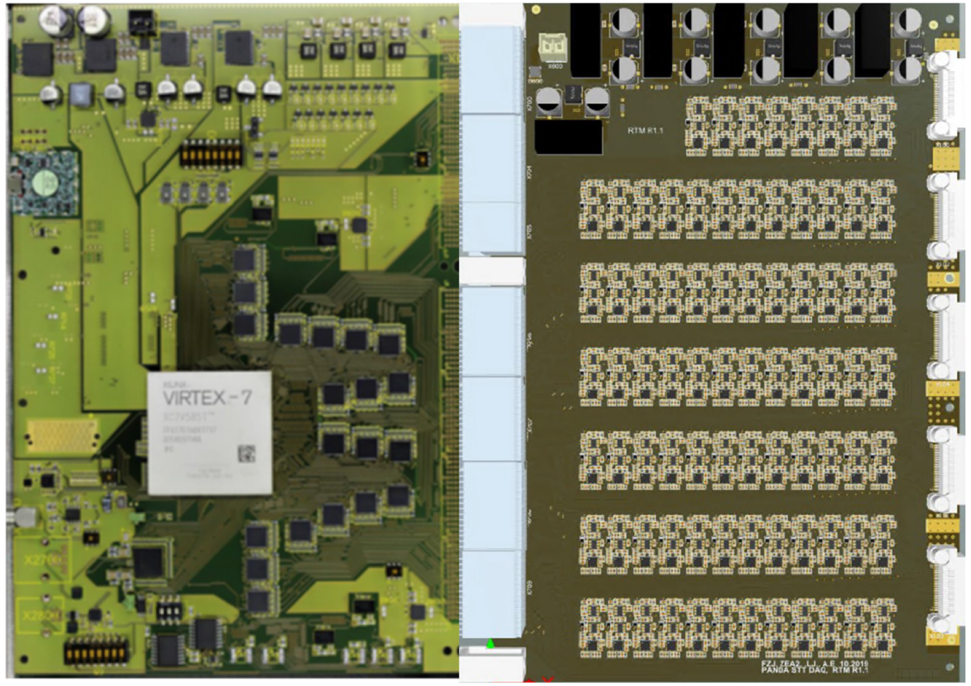
ADC based DAQ for PANDA STT

Level 0 Open VPX Crate

ADC based DAQ for PANDA STT (one of approaches):

- 160 channels (shaping, sampling and processing) per payload slot, 14 payload slots+2 controllers;
- **totally 2200 channels per crate**;
- time sorted output data stream (arrival time, energy,...)
- noise rejection, pile up resolution, base line correction, ..

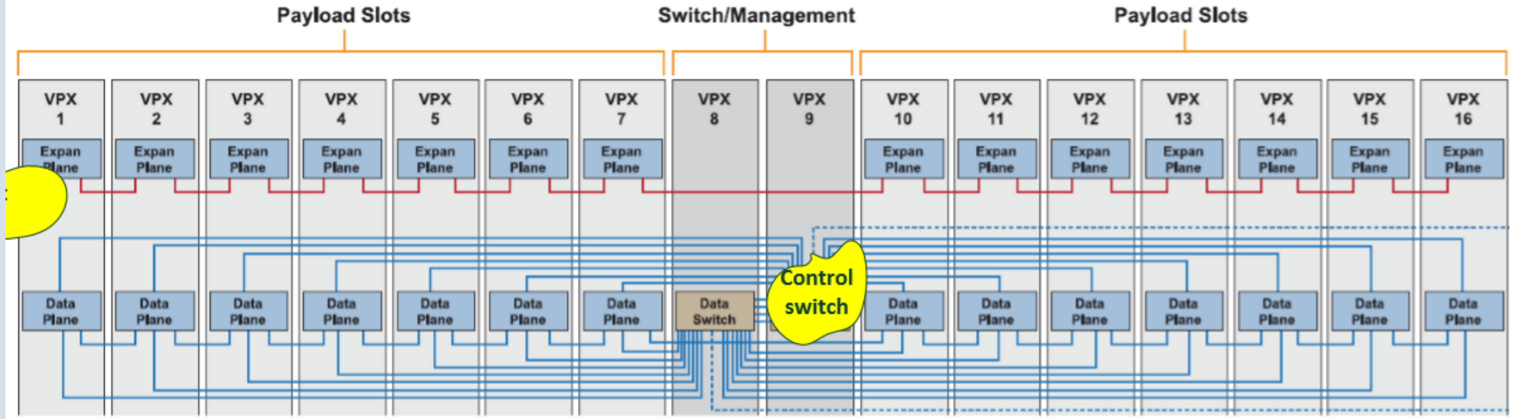




- 40 4-channel ADCs (configurable up to 1 GSPS);
- Single Virtex7 FPGA

- 160 Amplifiers;
- 5 connectors for 32-pins samtec cables

- *All information from the straw tube tracker is processed in one unit.*
- *Allows to build a complete STT event.*
- *This unit can also be used for calorimeters readout and processing.*



https://doi.org/10.1088/1748-0221/17/04/C04022
2022_JINST_17_C04022

Powerful Backplane up to 670 GBs

L. Jokhovets, P Kulessa ..

JÜLICH
Forschungszentrum

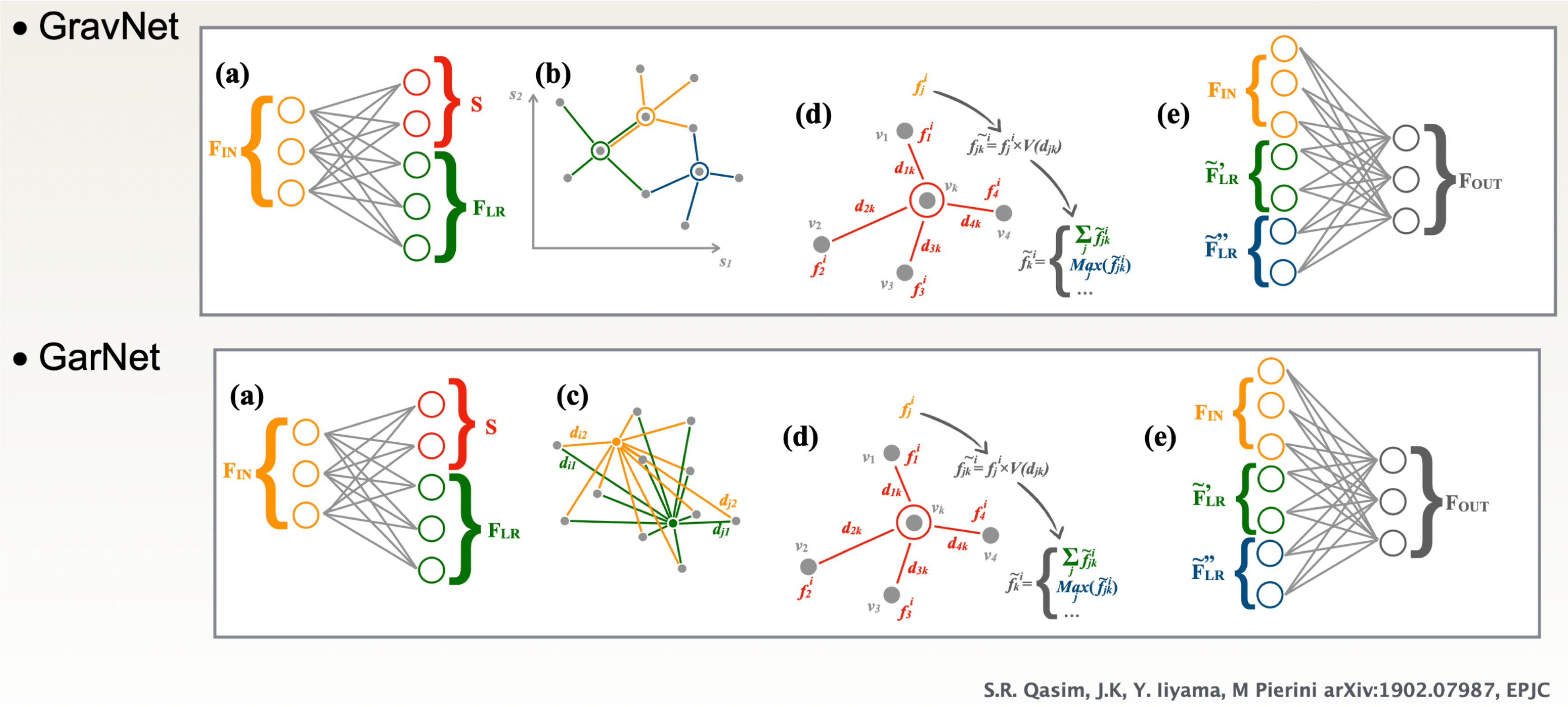
GPU vs FPGA

- Machine learning methods are widely used and have proven to be very powerful in particle physics.
- Although the methods of machine learning and artificial intelligence are developed by many groups and have a lot in common, nevertheless, the hardware used and performance is different.
- While the large numerical processing capability of GPUs is attractive, these technologies are optimized for high throughput, not low latency.
- FPGA-based trigger and data acquisition systems have extremely low, sub-microsecond latency requirements that are unique to particle physics.
- Definitely FPGA can work on a computer farm as an ML accelerator, but the internal FPGA performance will be degraded due to slow I/O through the computer and the PCIe bus. Not to mention the latency, which will increase by 2-3 orders of magnitude.
- Therefore, the most effective would be the use of ML-FPGA directly between the front-end stream and a computer farm, on which it is already more efficient to use the CPU and GPU for ML/AI.

GarNet for GEM-TRD and calorimeter

- ❑ *Another type of neural network, GarNet, shows good offline performance for particle identification using GEM-TRD.*
- ❑ *It is supported in HLS4ML and we are currently working on its implementation for FPGA.*
- ❑ *The IP core is synthesized, but the latency is too large for an online application, so more optimization work is required.*

"Learning representations of irregular particle-detector geometry with distance-weighted graph networks"
arXiv:1902.07987v2 [physics.data-an] 24 Jul 2019



• GravNet

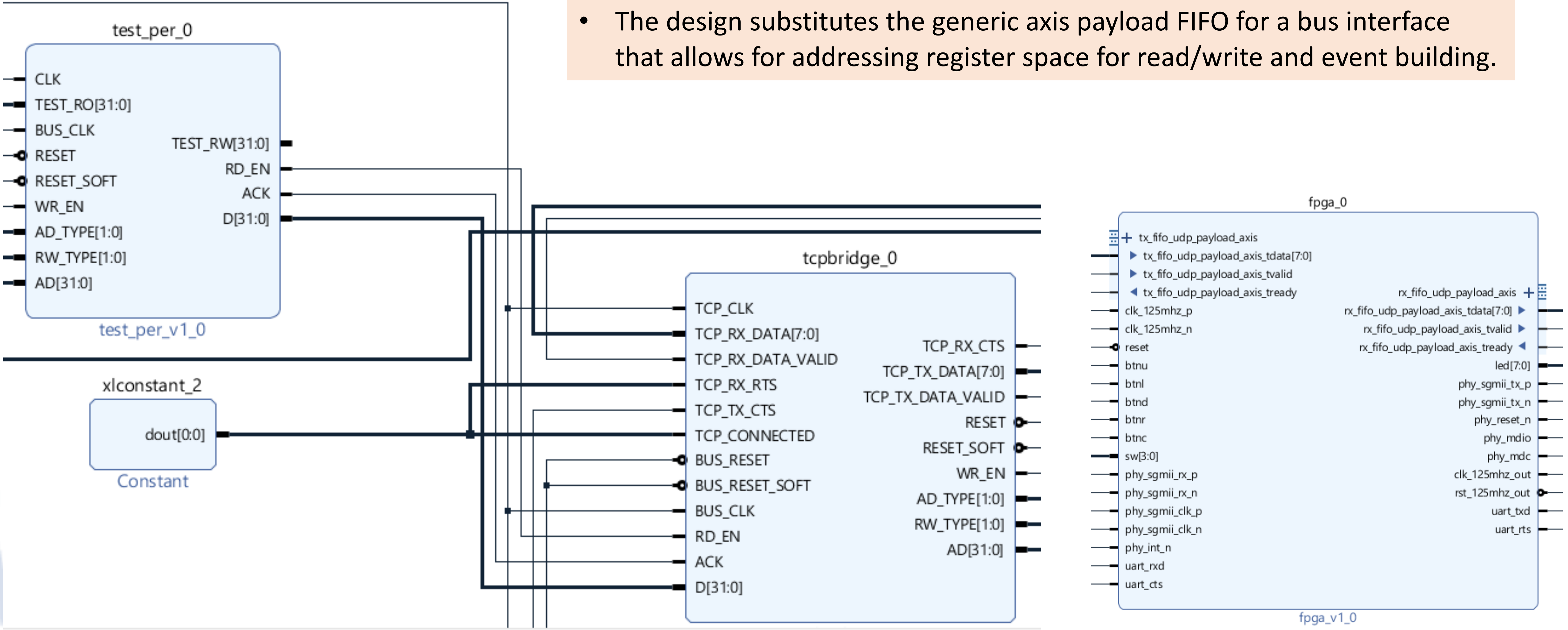
• GarNet

S.R. Qasim, J.K, Y. Iiyama, M Pierini arXiv:1902.07987, EPJC

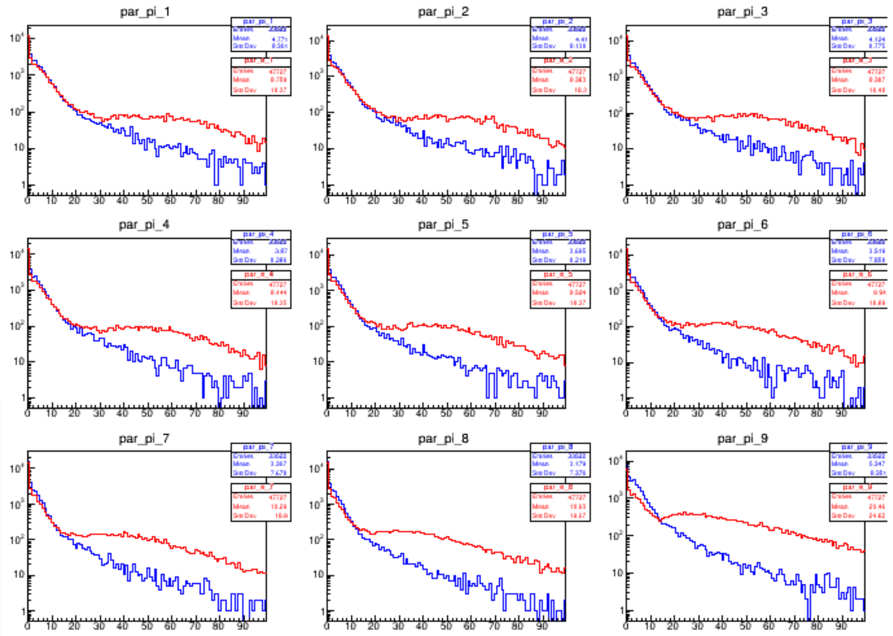
Developing ethernet interface

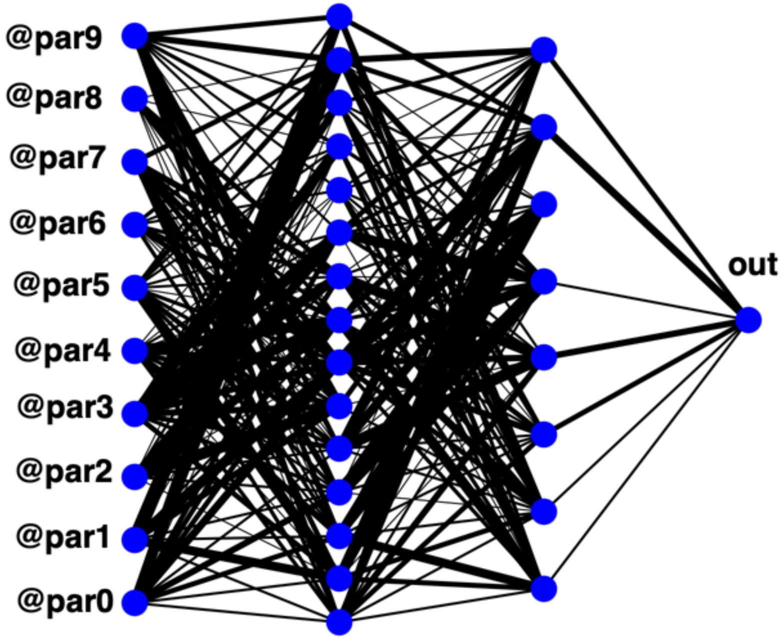
By Cody Dickover

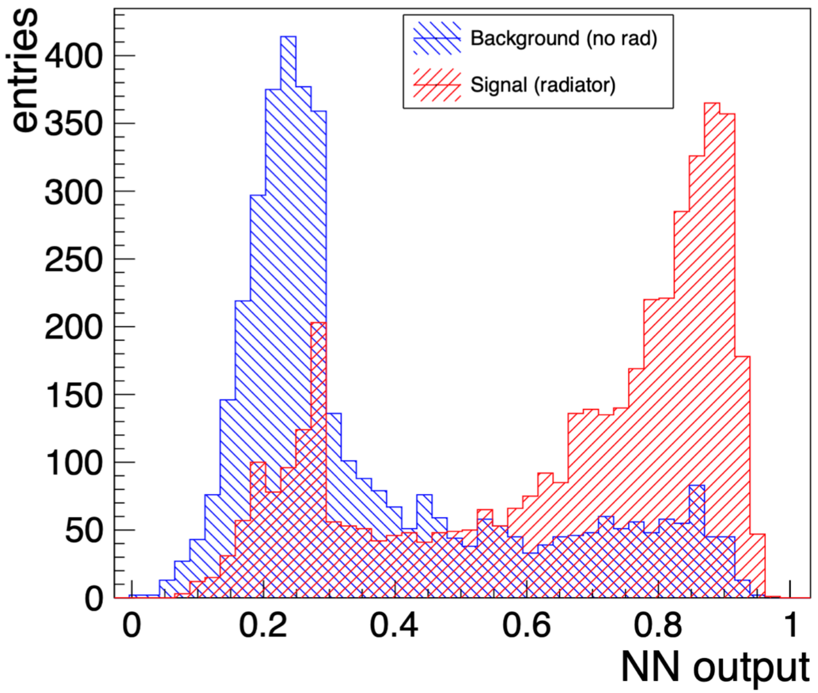
- Currently we using Microblaze setup for tests.
- For the beam test we need high speed interface to Detector/FADC.
- The design substitutes the generic axis payload FIFO for a bus interface that allows for addressing register space for read/write and event building.



GEM-TRD offline analysis

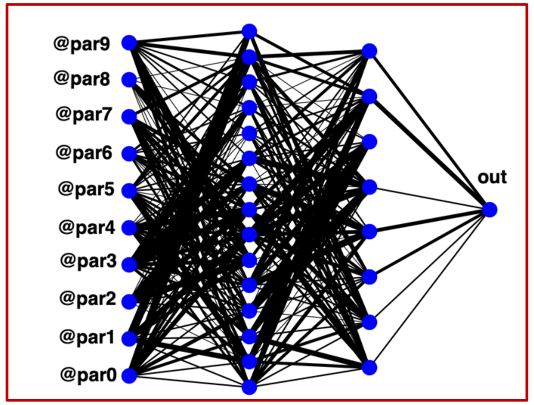






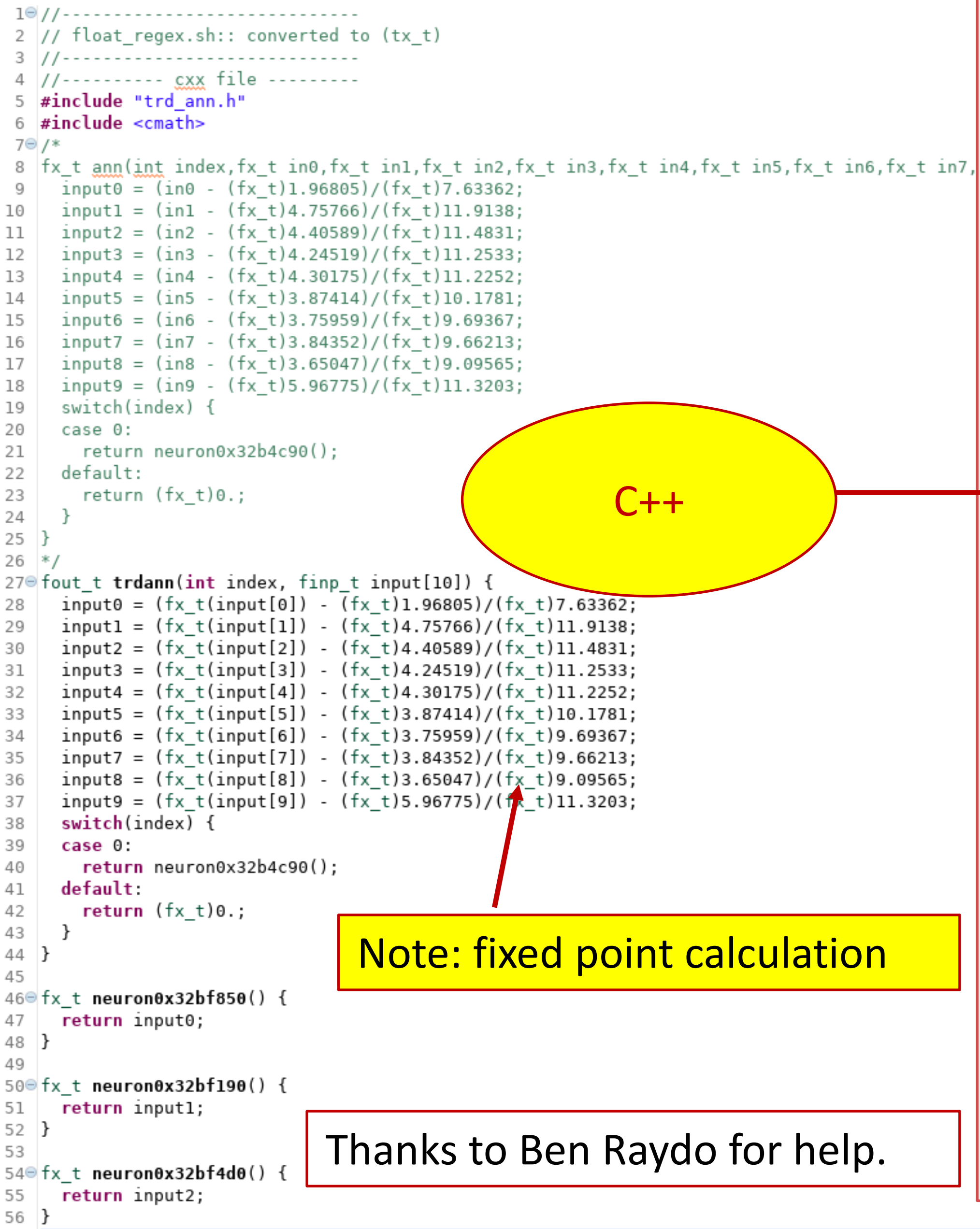
- ❑ *For data analysis we used a neural network library provided by root /TMVA package :*
  - ➢ MultiLayerPerceptron (MLP)
- ❑ *Top left plot shows ionization difference for e/pi in several bins along the track*
- ❑ *Top right plot shows neural network output for single TRD module:*
  - ➢ Red - electrons with radiator
  - ➢ Blue – electrons without radiator.

Xilinx HLS: C++ to Verilog



The C/C++ code of the trained network is used as input for Vivado_HLS.

The Xilinx Vivado HLS (High-Level Synthesis) tool provides a higher level of abstraction for the user by synthesizing functions written in C,C++ into IP blocks, by generating the appropriate ,low-level, VHDL and Verilog code. Then those blocks can be integrated into a real hardware system.

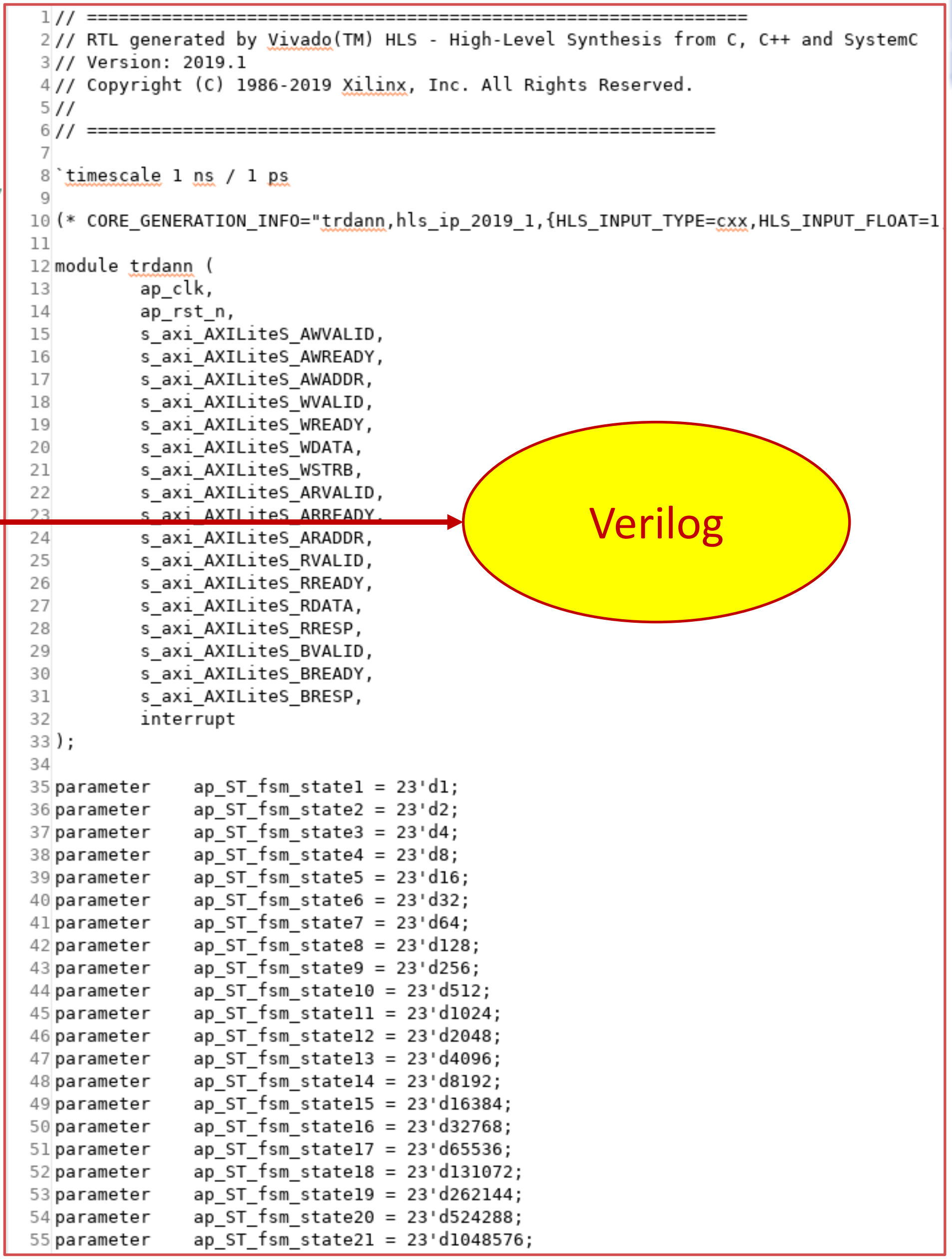


```cpp
//----------------------------
// float_regex.sh:: converted to (tx_t)
//----------------------------
//---------- cxx file ---------
#include "trd_ann.h"
#include <cmath>
/*
fx_t ann(int index,fx_t in0,fx_t in1,fx_t in2,fx_t in3,fx_t in4,fx_t in5,fx_t in6,fx_t in7,
  input0 = (in0 - (fx_t)1.96805)/(fx_t)7.63362;
  input1 = (in1 - (fx_t)4.75766)/(fx_t)11.9138;
  input2 = (in2 - (fx_t)4.40589)/(fx_t)11.4831;
  input3 = (in3 - (fx_t)4.24519)/(fx_t)11.2533;
  input4 = (in4 - (fx_t)4.30175)/(fx_t)11.2252;
  input5 = (in5 - (fx_t)3.87414)/(fx_t)10.1781;
  input6 = (in6 - (fx_t)3.75959)/(fx_t)9.69367;
  input7 = (in7 - (fx_t)3.84352)/(fx_t)9.66213;
  input8 = (in8 - (fx_t)3.65047)/(fx_t)9.09565;
  input9 = (in9 - (fx_t)5.96775)/(fx_t)11.3203;
  switch(index) {
  case 0:
    return neuron0x32b4c90();
  default:
    return (fx_t)0.;
  }
}
*/
fout_t trdann(int index, finp_t input[10]) {
  input0 = (fx_t(input[0]) - (fx_t)1.96805)/(fx_t)7.63362;
  input1 = (fx_t(input[1]) - (fx_t)4.75766)/(fx_t)11.9138;
  input2 = (fx_t(input[2]) - (fx_t)4.40589)/(fx_t)11.4831;
  input3 = (fx_t(input[3]) - (fx_t)4.24519)/(fx_t)11.2533;
  input4 = (fx_t(input[4]) - (fx_t)4.30175)/(fx_t)11.2252;
  input5 = (fx_t(input[5]) - (fx_t)3.87414)/(fx_t)10.1781;
  input6 = (fx_t(input[6]) - (fx_t)3.75959)/(fx_t)9.69367;
  input7 = (fx_t(input[7]) - (fx_t)3.84352)/(fx_t)9.66213;
  input8 = (fx_t(input[8]) - (fx_t)3.65047)/(fx_t)9.09565;
  input9 = (fx_t(input[9]) - (fx_t)5.96775)/(fx_t)11.3203;
  switch(index) {
  case 0:
    return neuron0x32b4c90();
  default:
    return (fx_t)0.;
  }
}

fx_t neuron0x32bf850() {
  return input0;
}

fx_t neuron0x32bf190() {
  return input1;
}

fx_t neuron0x32bf4d0() {
  return input2;
}
```

C++

Note: fixed point calculation

Thanks to Ben Raydo for help.



```verilog
// ==============================================================
// RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
// Version: 2019.1
// Copyright (C) 1986-2019 Xilinx, Inc. All Rights Reserved.
//
// ===========================================================

`timescale 1 ns / 1 ps

(* CORE_GENERATION_INFO="trdann,hls_ip_2019_1,{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=1

module trdann (
        ap_clk,
        ap_rst_n,
        s_axi_AXILiteS_AWVALID,
        s_axi_AXILiteS_AWREADY,
        s_axi_AXILiteS_AWADDR,
        s_axi_AXILiteS_WVALID,
        s_axi_AXILiteS_WREADY,
        s_axi_AXILiteS_WDATA,
        s_axi_AXILiteS_WSTRB,
        s_axi_AXILiteS_ARVALID,
        s_axi_AXILiteS_ARREADY,
        s_axi_AXILiteS_ARADDR,
        s_axi_AXILiteS_RVALID,
        s_axi_AXILiteS_RREADY,
        s_axi_AXILiteS_RDATA,
        s_axi_AXILiteS_RRESP,
        s_axi_AXILiteS_BVALID,
        s_axi_AXILiteS_BREADY,
        s_axi_AXILiteS_BRESP,
        interrupt
);

parameter    ap_ST_fsm_state1 = 23'd1;
parameter    ap_ST_fsm_state2 = 23'd2;
parameter    ap_ST_fsm_state3 = 23'd4;
parameter    ap_ST_fsm_state4 = 23'd8;
parameter    ap_ST_fsm_state5 = 23'd16;
parameter    ap_ST_fsm_state6 = 23'd32;
parameter    ap_ST_fsm_state7 = 23'd64;
parameter    ap_ST_fsm_state8 = 23'd128;
parameter    ap_ST_fsm_state9 = 23'd256;
parameter    ap_ST_fsm_state10 = 23'd512;
parameter    ap_ST_fsm_state11 = 23'd1024;
parameter    ap_ST_fsm_state12 = 23'd2048;
parameter    ap_ST_fsm_state13 = 23'd4096;
parameter    ap_ST_fsm_state14 = 23'd8192;
parameter    ap_ST_fsm_state15 = 23'd16384;
parameter    ap_ST_fsm_state16 = 23'd32768;
parameter    ap_ST_fsm_state17 = 23'd65536;
parameter    ap_ST_fsm_state18 = 23'd131072;
parameter    ap_ST_fsm_state19 = 23'd262144;
parameter    ap_ST_fsm_state20 = 23'd524288;
parameter    ap_ST_fsm_state21 = 23'd1048576;
```

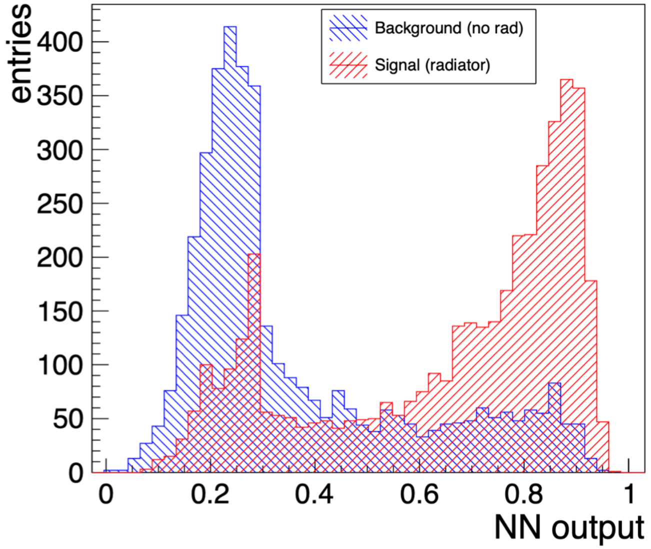
Verilog

Test NN IP in FPGA

Test tools:

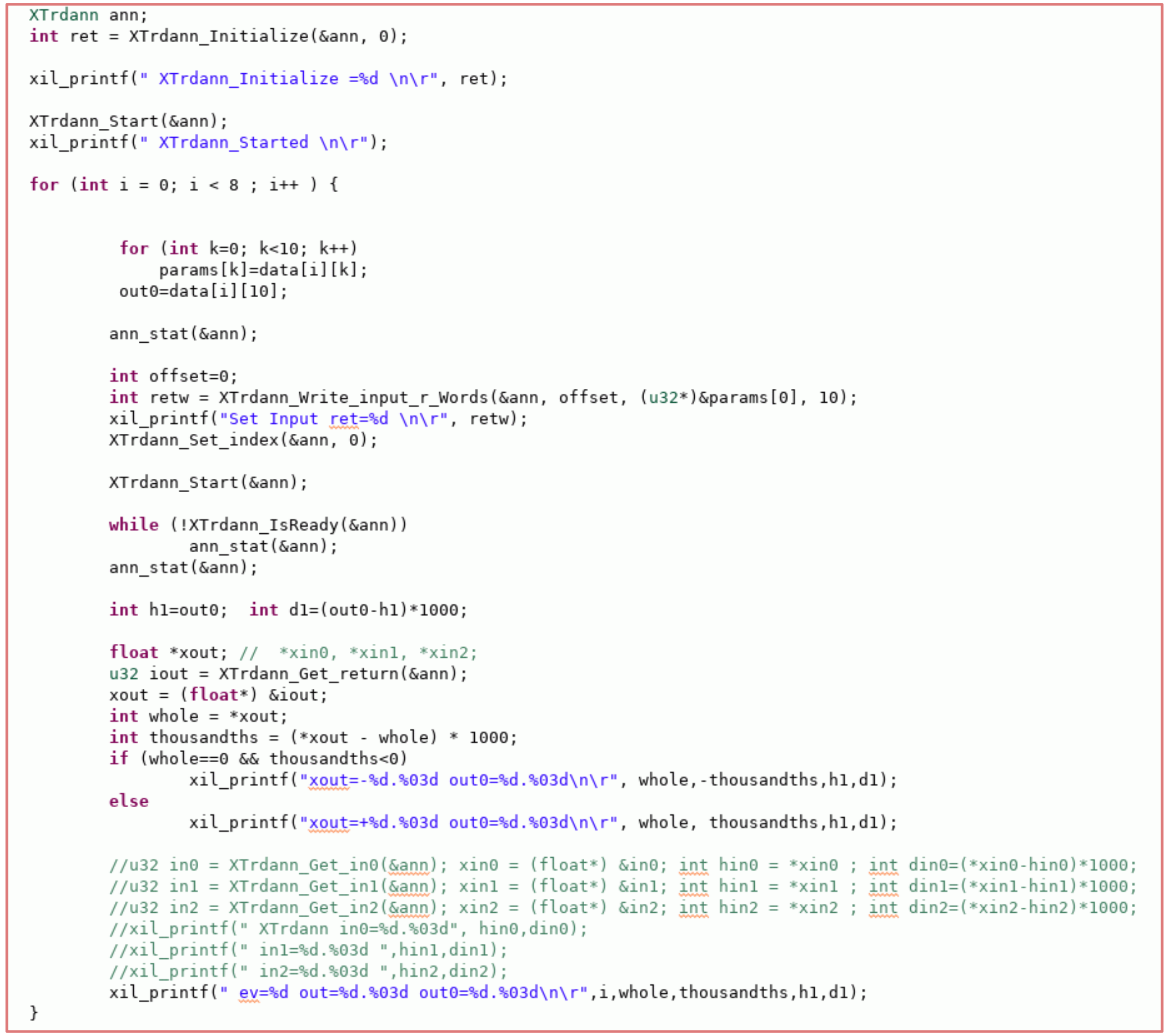
1. Vivado SDK
2. Petalinux

```
ev=0 out=0.192 out0=0.197
ev=1 out=0.192 out0=0.197
ev=2 out=0.233 out0=0.236
ev=3 out=0.192 out0=0.197
ev=4 out=0.165 out0=0.169
ev=5 out=0.192 out0=0.196
ev=6 out=0.462 out0=0.470
ev=7 out=0.187 out0=0.191
```



C++ code for test :

XTrdann ann; // create an instance of ML core.



```
XTrdann ann;
int ret = XTrdann_Initialize(&ann, 0);

xil_printf(" XTrdann_Initialize =%d \n\r", ret);

XTrdann_Start(&ann);
xil_printf(" XTrdann_Started \n\r");

for (int i = 0; i < 8 ; i++ ) {


         for (int k=0; k<10; k++)
             params[k]=data[i][k];
         out0=data[i][10];

        ann_stat(&ann);

        int offset=0;
        int retw = XTrdann_Write_input_r_Words(&ann, offset, (u32*)&params[0], 10);
        xil_printf("Set Input ret=%d \n\r", retw);
        XTrdann_Set_index(&ann, 0);

        XTrdann_Start(&ann);

        while (!XTrdann_IsReady(&ann))
                ann_stat(&ann);
        ann_stat(&ann);

        int h1=out0;  int d1=(out0-h1)*1000;

        float *xout; //  *xin0, *xin1, *xin2;
        u32 iout = XTrdann_Get_return(&ann);
        xout = (float*) &iout;
        int whole = *xout;
        int thousandths = (*xout - whole) * 1000;
        if (whole==0 && thousandths<0)
                xil_printf("xout=-%d.%03d out0=%d.%03d\n\r", whole,-thousandths,h1,d1);
        else
                xil_printf("xout=+%d.%03d out0=%d.%03d\n\r", whole, thousandths,h1,d1);

        //u32 in0 = XTrdann_Get_in0(&ann); xin0 = (float*) &in0; int hin0 = *xin0 ; int din0=(*xin0-hin0)*1000;
        //u32 in1 = XTrdann_Get_in1(&ann); xin1 = (float*) &in1; int hin1 = *xin1 ; int din1=(*xin1-hin1)*1000;
        //u32 in2 = XTrdann_Get_in2(&ann); xin2 = (float*) &in2; int hin2 = *xin2 ; int din2=(*xin2-hin2)*1000;
        //xil_printf(" XTrdann in0=%d.%03d", hin0,din0);
        //xil_printf(" in1=%d.%03d ",hin1,din1);
        //xil_printf(" in2=%d.%03d ",hin2,din2);
        xil_printf(" ev=%d out=%d.%03d out0=%d.%03d\n\r",i,whole,thousandths,h1,d1);
}
```