Development of ML FPGA filter for particle identification and tracking in real time

Sergey Furletov
Jefferson Lab

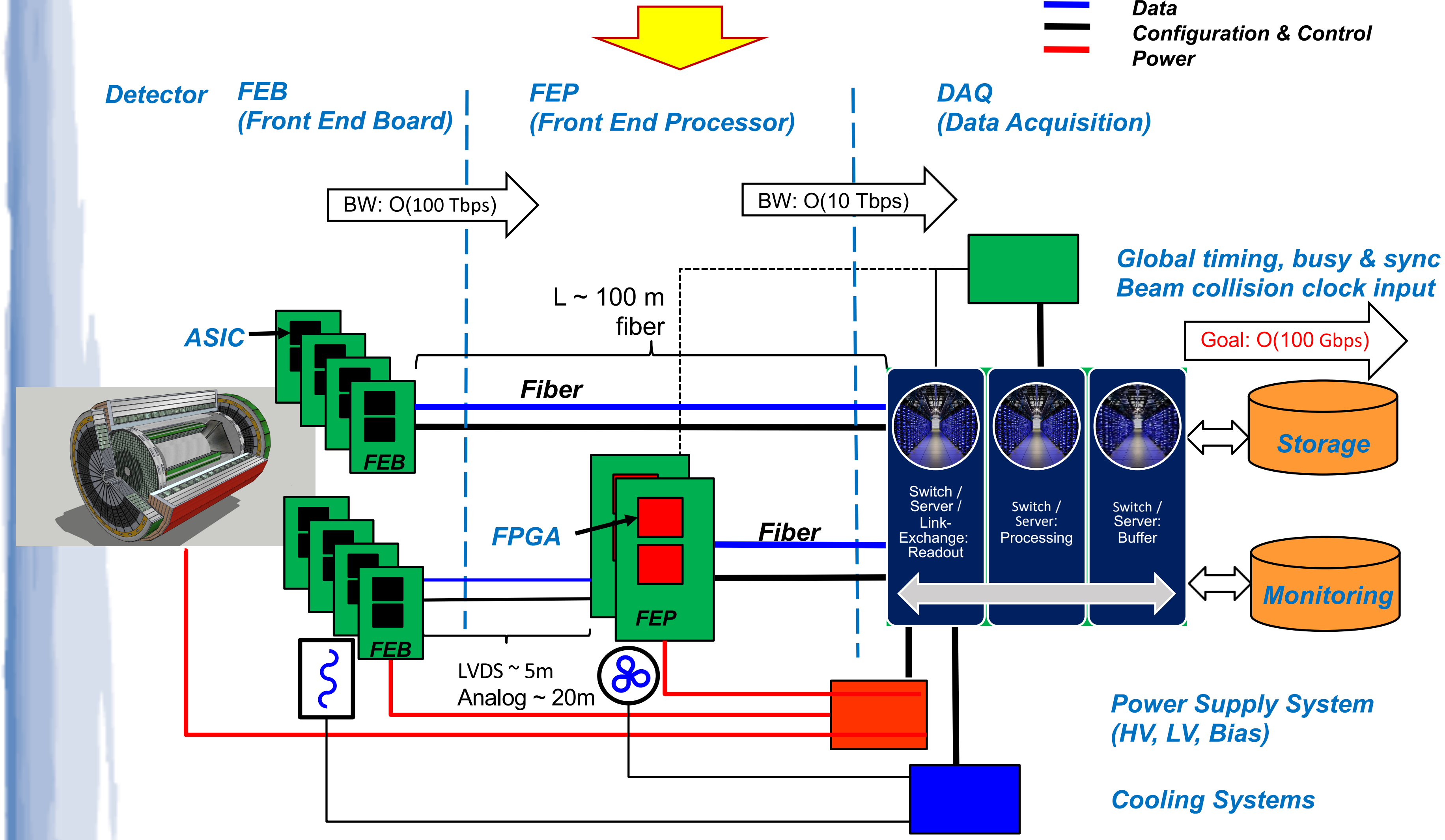
Team :
F. Barbosa, L. Belfore, N. Branson, C. Dickover, C. Fanelli, D. Furletov, S. Furletov, L. Jokhovets, D. Lawrence, D. Romanov

2nd workshop on Artificial Intelligence for the EIC

13 Oct 2022

EIC streaming readout as motivation

Jefferson Lab
Thomas Jefferson National Accelerator Facility

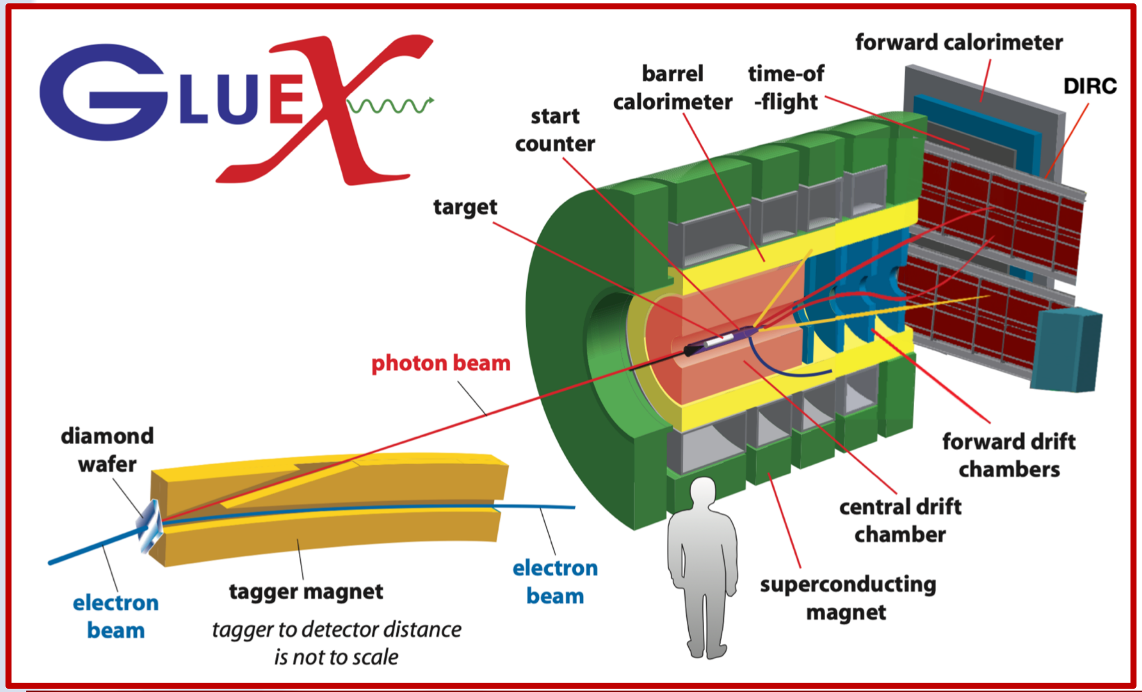


- The correct location for the ML on the FPGA filter is called "FEP" in this figure.
- This gives us a chance to reduce traffic earlier.
- Allows us to touch physics: ML brings intelligence to L1.
- However, it is now unclear how far we can go with physics at the FPGA.
- Initially, we can start in pass-through 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.

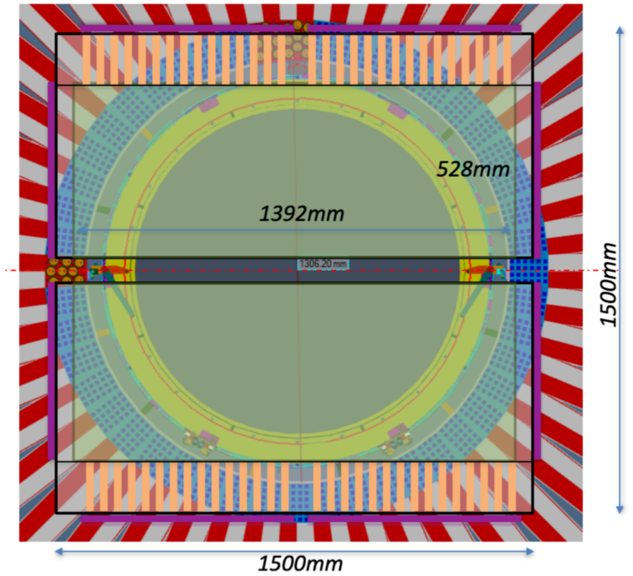
Motivation

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *Real-time data processing is a frontier field in experimental particle physics.*
- *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 work described in this report aims to test ML-FPGA algorithms in a triggered data acquisition system, as well as in streaming data acquisition, such as in the future EIC collider.*
- *The first target is the GlueX experiment, with a plan to build a Transition Radiation Detector (TRD) based on GEM technology (GEM-TRD), to improve the electron-pion separation in the GlueX experiment. It will allow to study precisely reactions with electron-positron pairs in the final states.*



GEMTRD



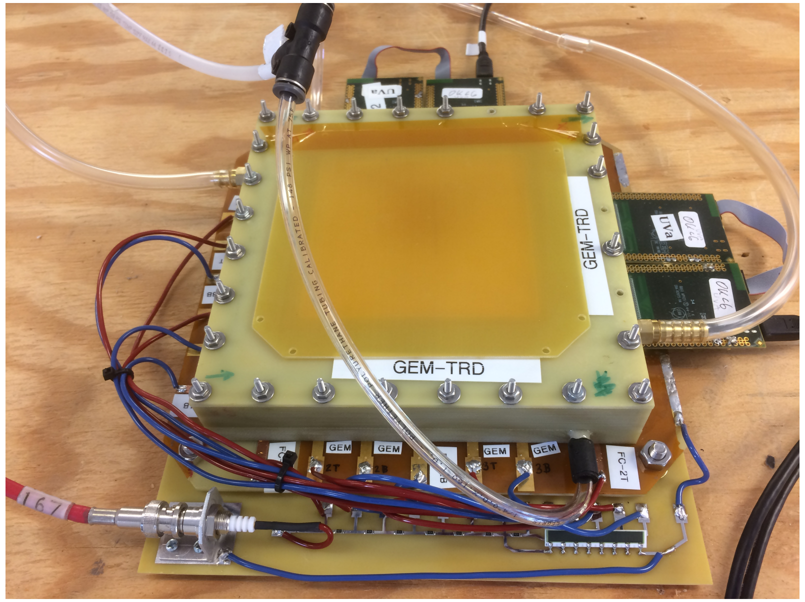
- *GEM-TRD will be installed in front of DIRC detector.*
- *Hall D is dedicated to the operation with a linearly-polarized photon beam produced by ~12 GeV electrons from CEBAF at Jefferson Lab.*
- *Typical trigger rate 40-70 kHz*
- *Data rate 0.7 – 1.2 GB/s*
- *Trigger latency 3.5 us.*

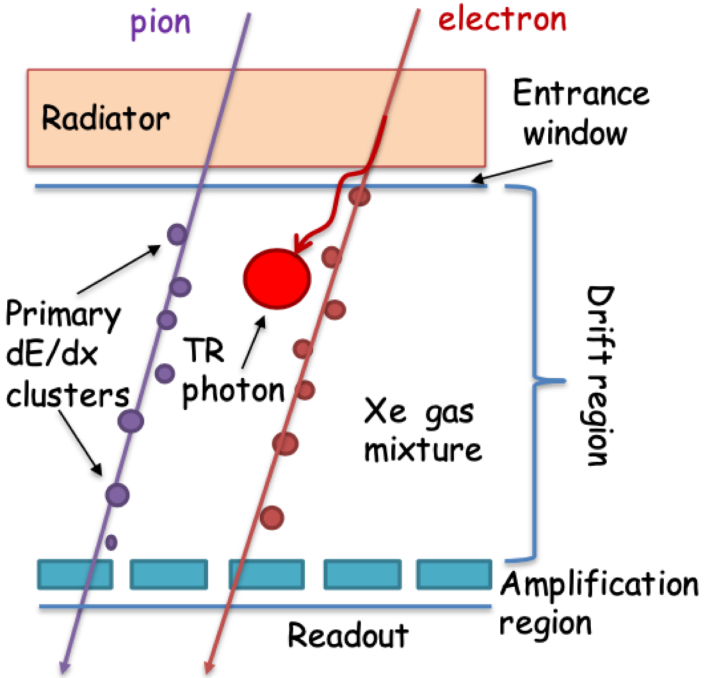
GEM-TRD prototype (eRD22) for EIC R&D

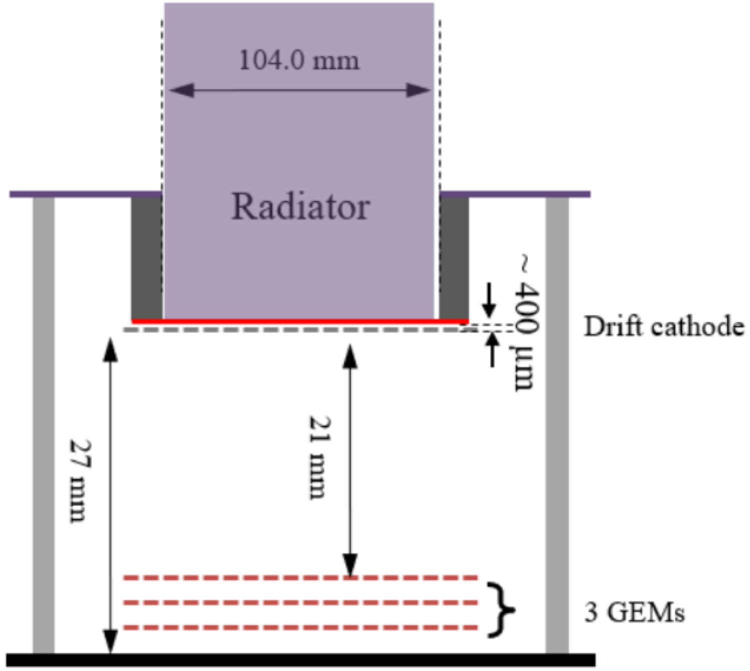
Jefferson Lab
Thomas Jefferson National Accelerator Facility

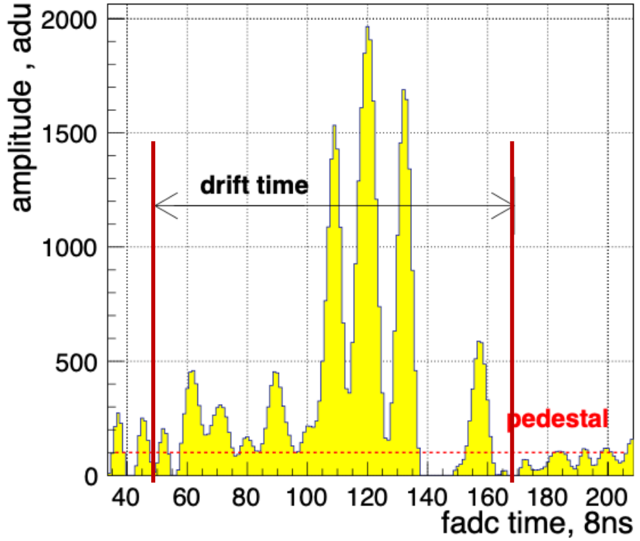
- To demonstrate the operating principle of the ML FPGA, we use the existing setup
- from the EIC detector R&D project (eRD22)
- A test module was built at the University of Virginia
- The prototype of GEMTRD/T module has a size of 10 cm × 10 cm with a corresponding to a total of 512 channels for X/Y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

- GEM-TRD provides e/hadron separation and tracking







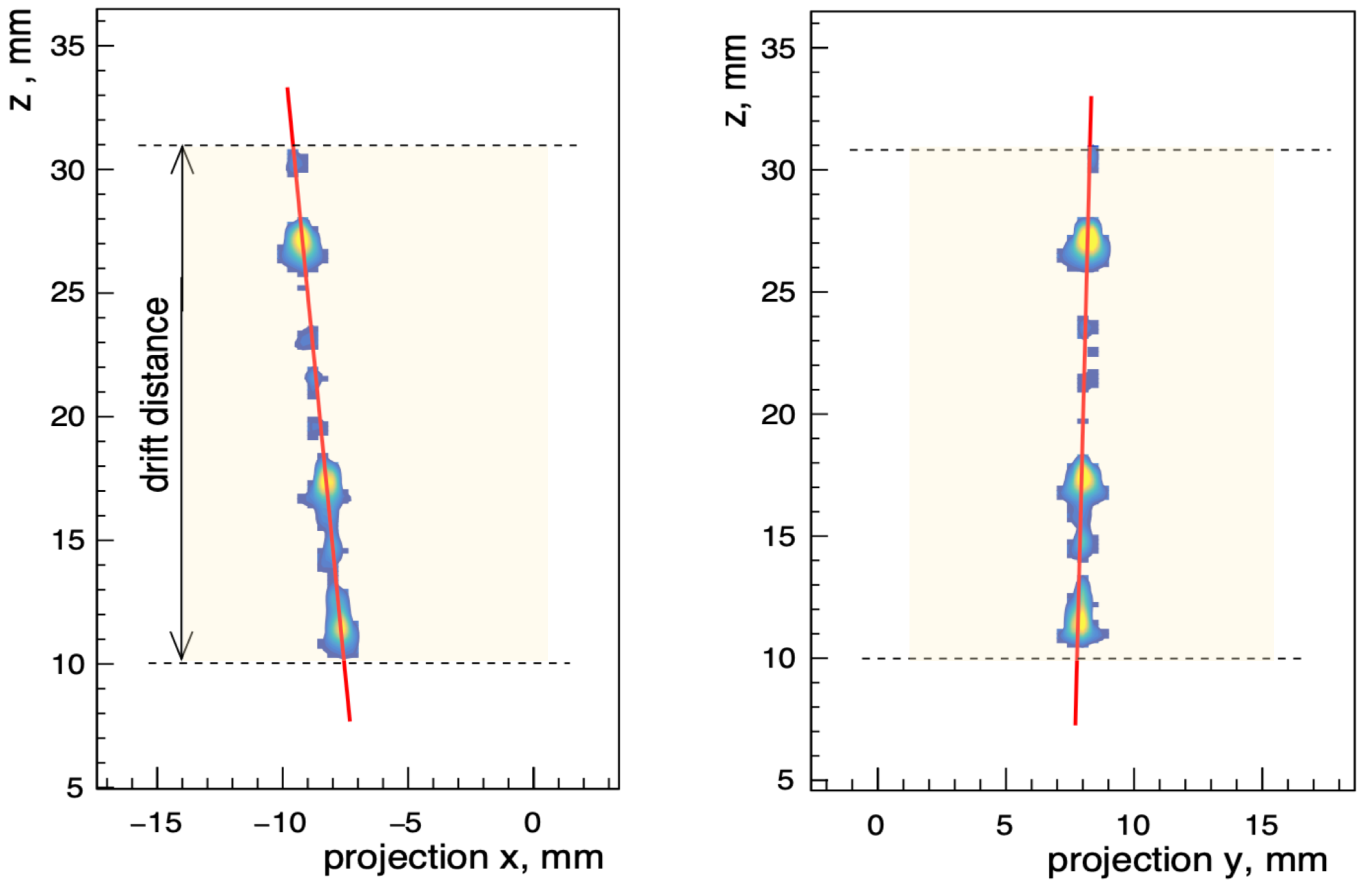


GEM-TRD principle

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *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.

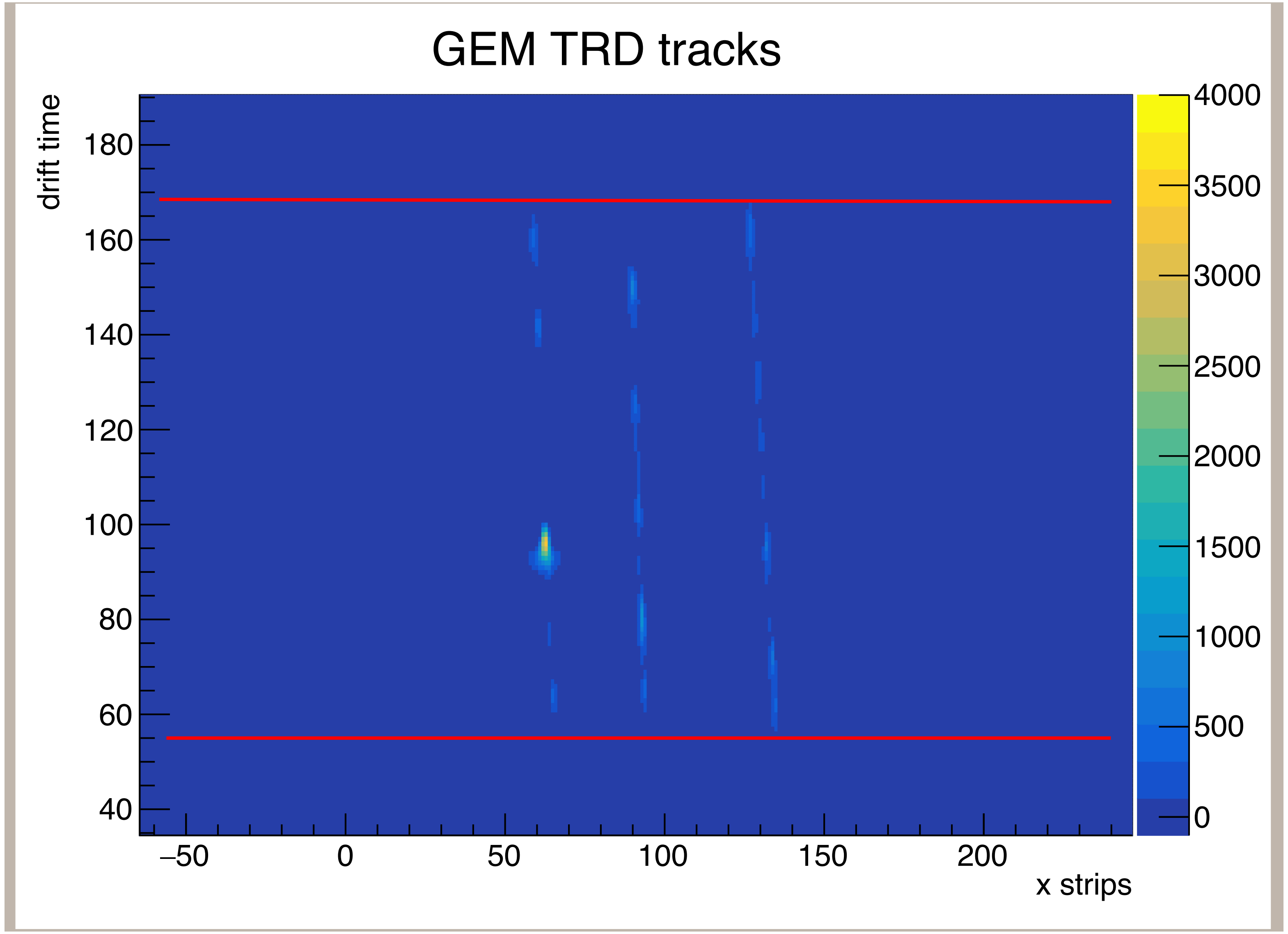
GEM-TRD can work as micro TPC, providing 3D track segments



GEM-TRD tracks

Jefferson Lab
Thomas Jefferson National Accelerator Facility

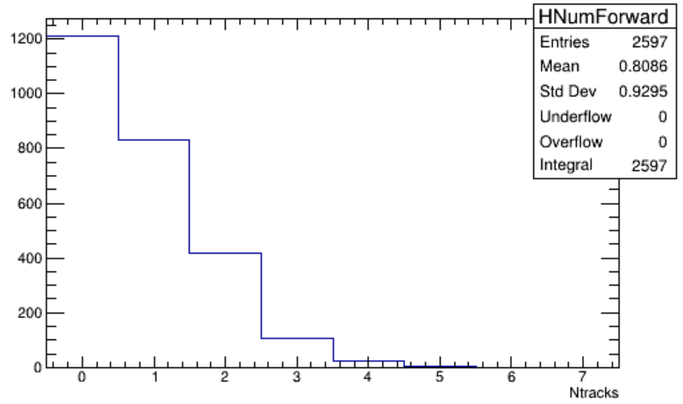
- *In a real experiment, GEMTRD will have multiple tracks.*
- *So we also need a fast algorithm for pattern recognition*
- *As well as for track fitting.*

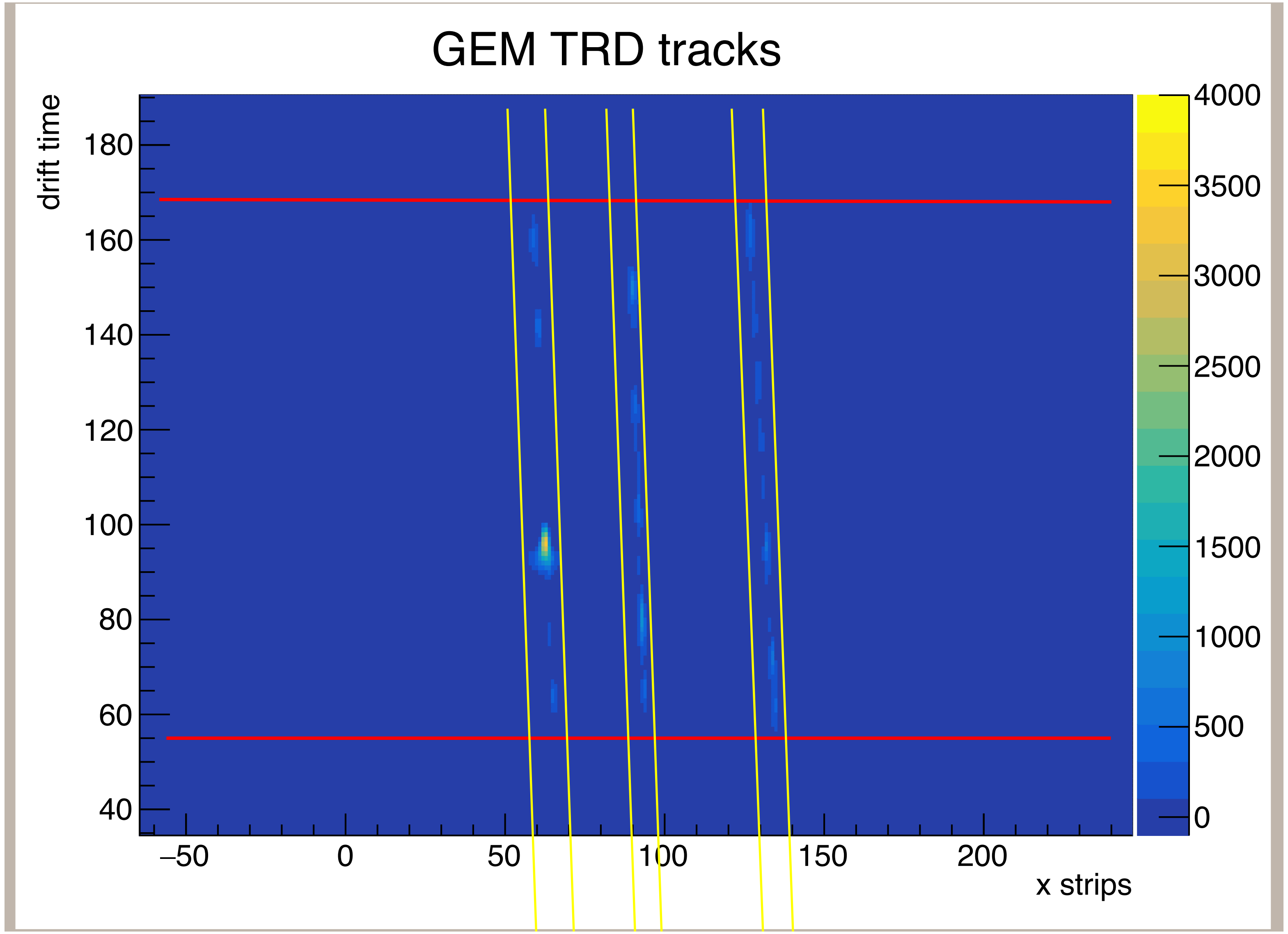


GEMTRD tracks

- 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



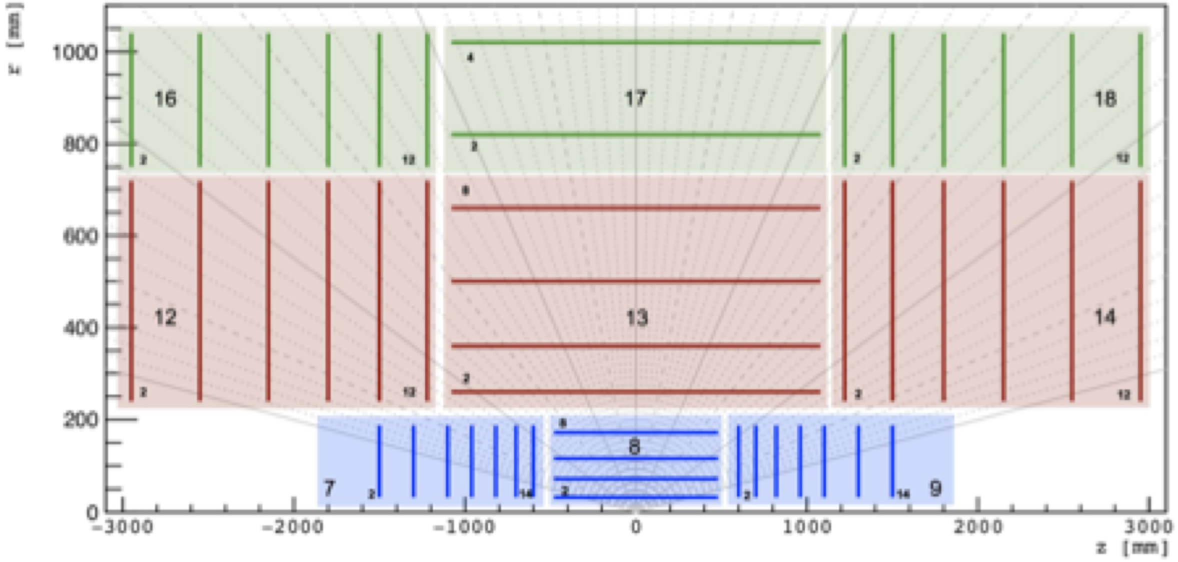


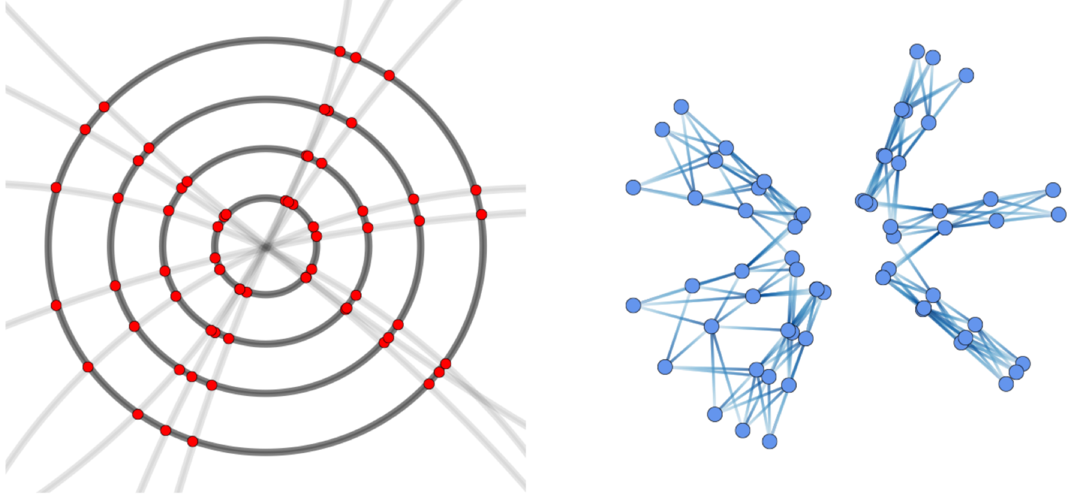
Existing GNN tracking projects

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- ❑ **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

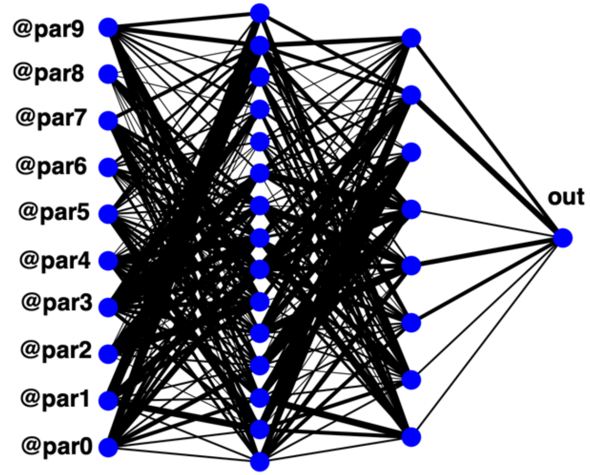
So we decided to start by evaluating an Exa.TrkX solution

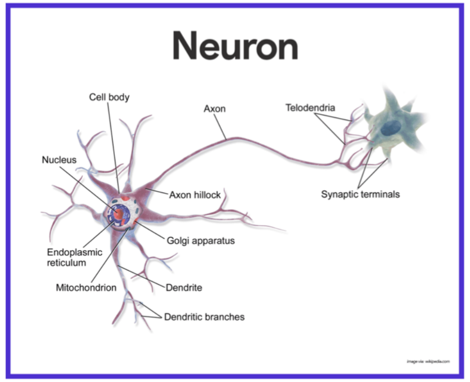
Moving forward : ML on FPGA

Jefferson Lab
Thomas Jefferson National Accelerator Facility

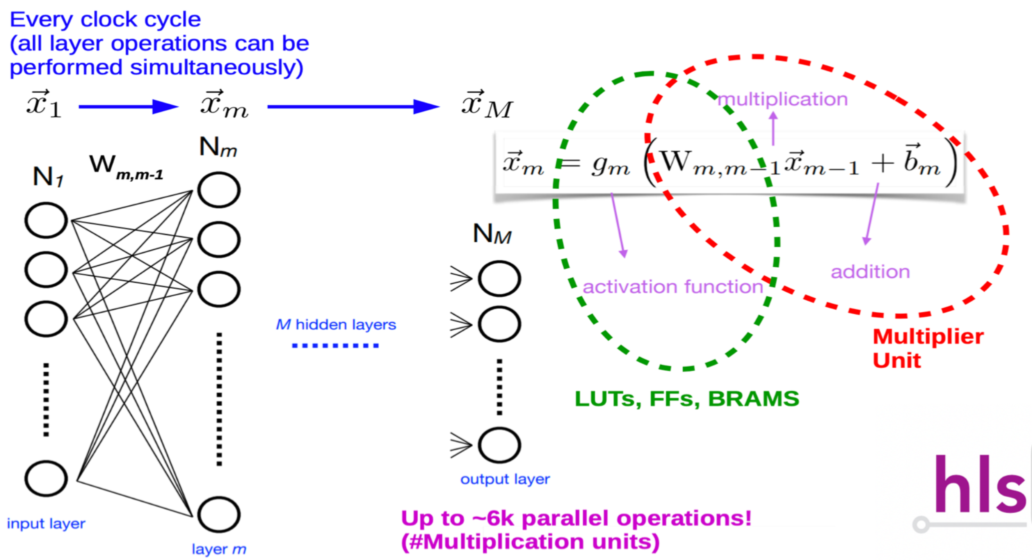
- *Offline analysis using ML looks promising.*
- *Can it be done in real time ?*
- *Here are some of the possible solutions :*
    - Computer farm.
    - CPU + GPU
    - CPU + FPGA
    - FPGA only

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





Inference on an FPGA



hls4ml

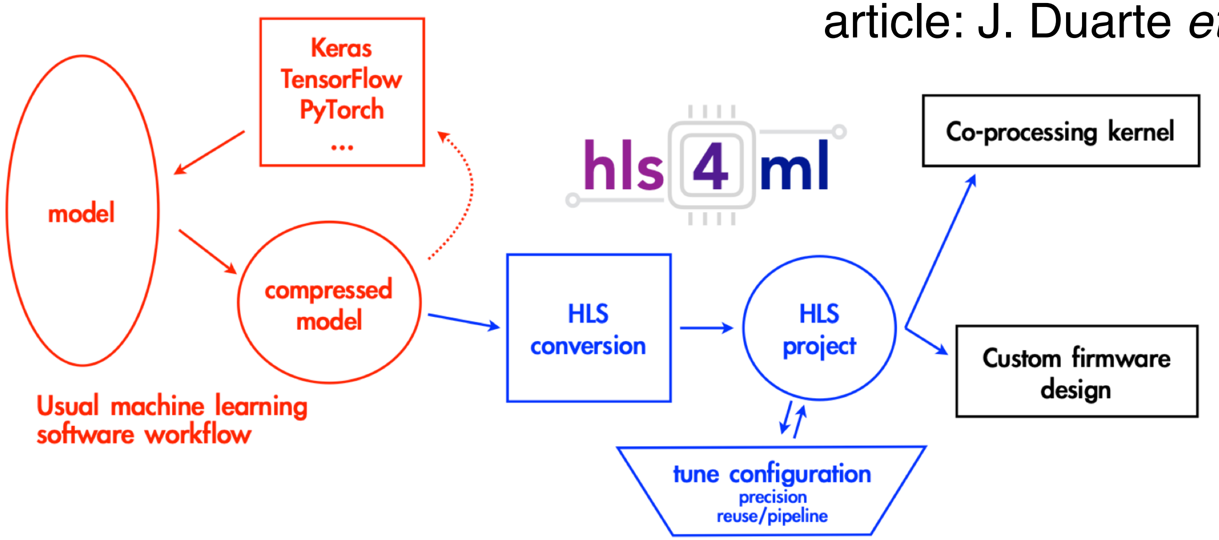
IRIS-HEP th Febraury 13 , 2019 Dylan Rankin [MIT]

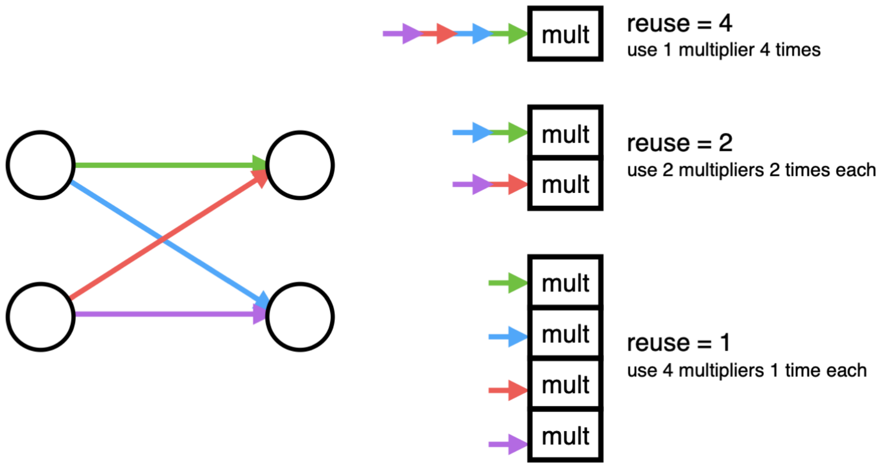
- Modern FPGAs have DSP slices - specialized hardware blocks placed between gateways and routers that perform mathematical calculations.
- The number of DSP slices can be up to 6000-12000 per chip.

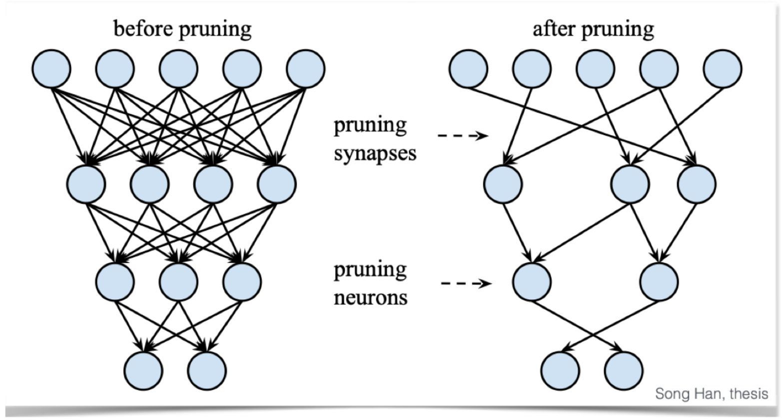
Optimization with hls4ml package

- 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



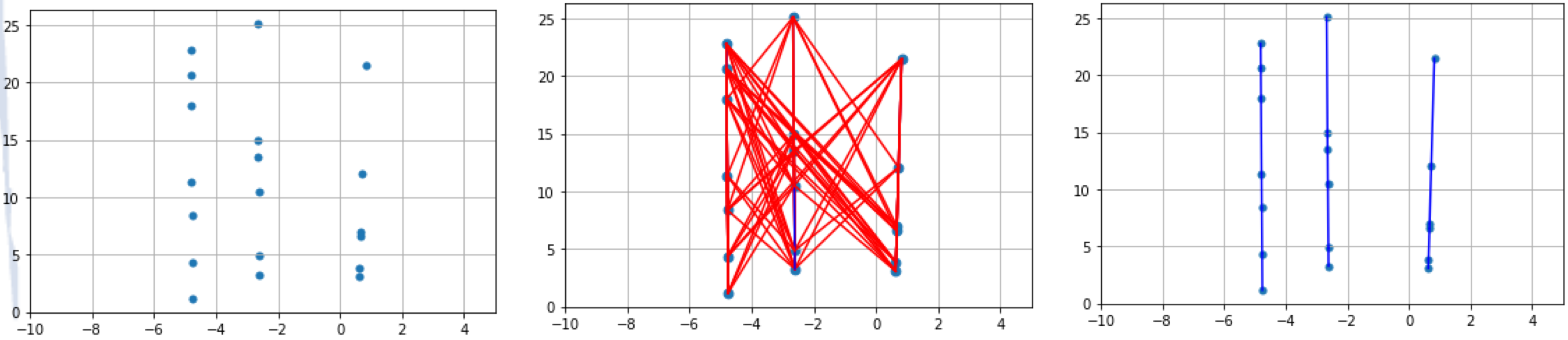




GNN for pattern recognition

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *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*

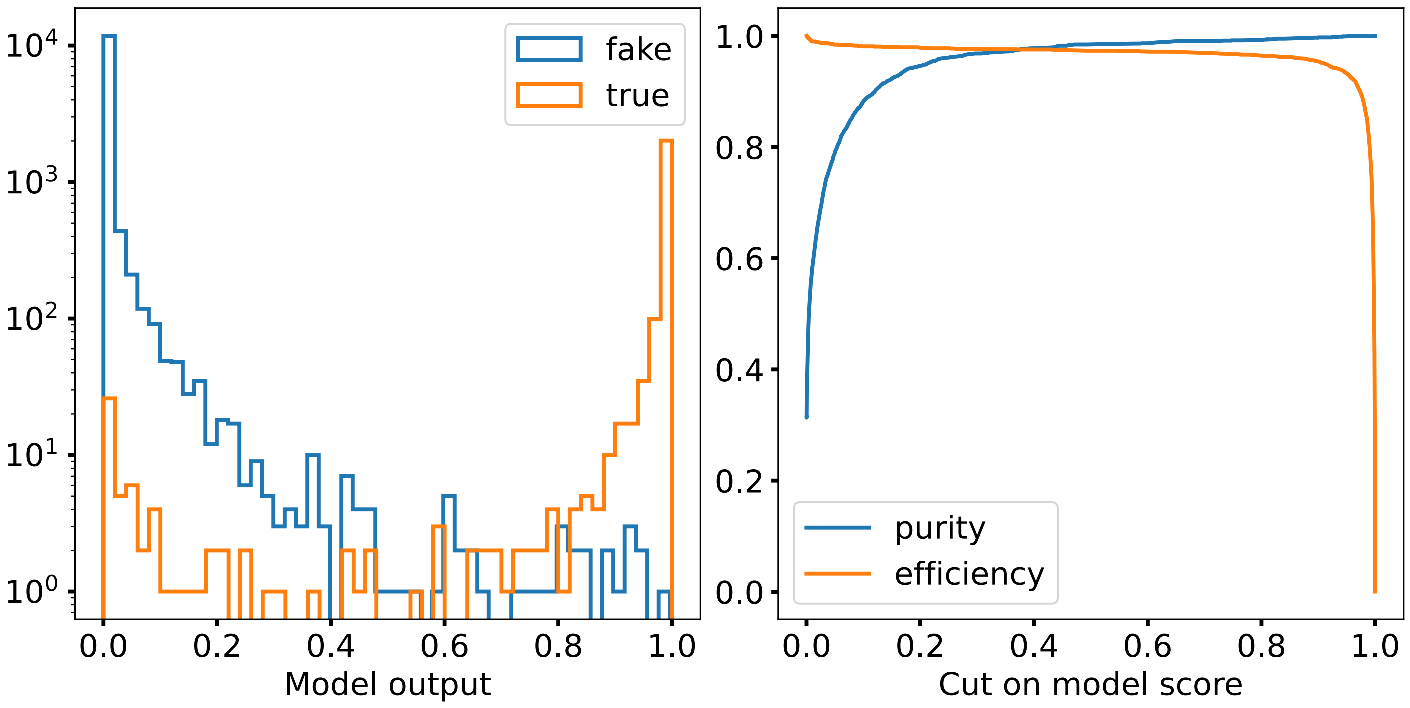


GNN performance

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *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).*

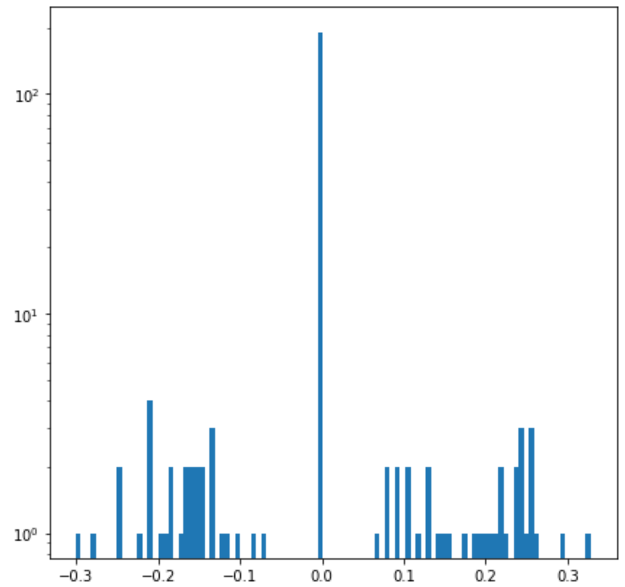
Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM(%)	DSP(%)
1.390E3	-	279	-	no	~0	68
15.000	-	1	-	yes	0	39
15.000	-	1	-	yes	0	9
15.000	-	1	-	yes	0	6
20.000	-	1	-	yes	0	3
15.000	-	1	-	yes	0	3
20.000	-	1	-	yes	0	1
0.0	-	0	-	no	0	0
20.000	-	1	-	yes	0	1
15.000	-	1	-	yes	0	~0
60.000	-	1	-	yes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

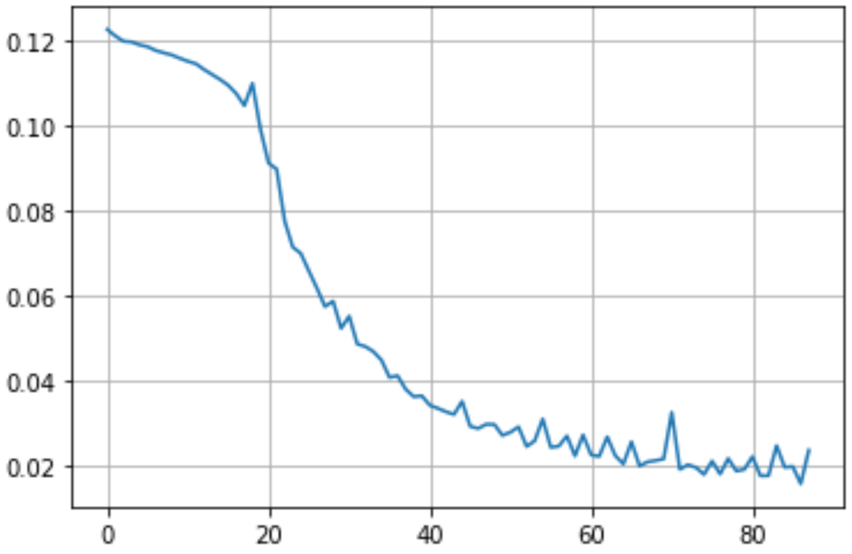


RNN/LSTM for track fit

- *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.*

% of zeros = 0.75





+ Latency (clock cycles):
  * Summary:

Latency min	Latency max	Interval min	Interval max	Pipeline 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

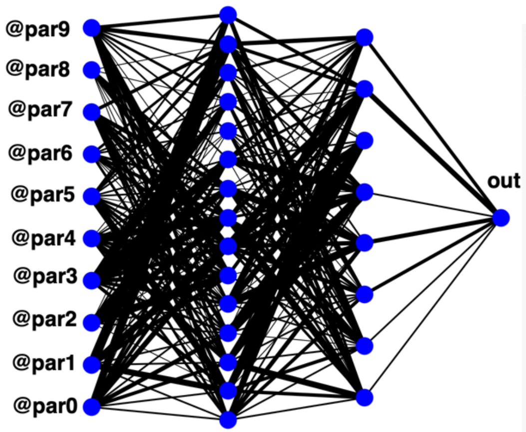
* Summary:

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

Jefferson Lab
Thomas Jefferson National Accelerator Facility

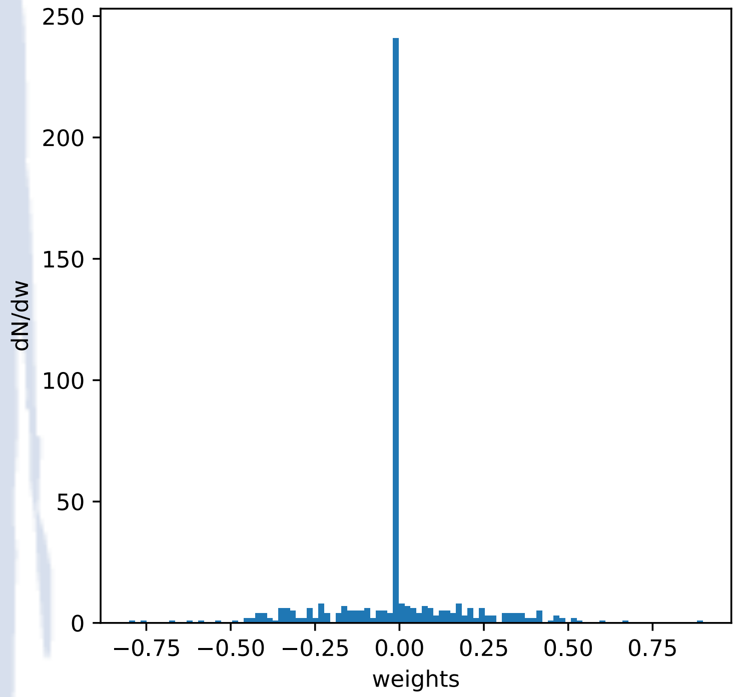
- *After the track is fit, the ionization along the track can be counted.*
- *The distance along the track is divided into 10-20 bins, and the ionization energy in these bins is fed to the input of the MLP neural network.*
- *Typically neural network weights often have many zeros, thus, it is possible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

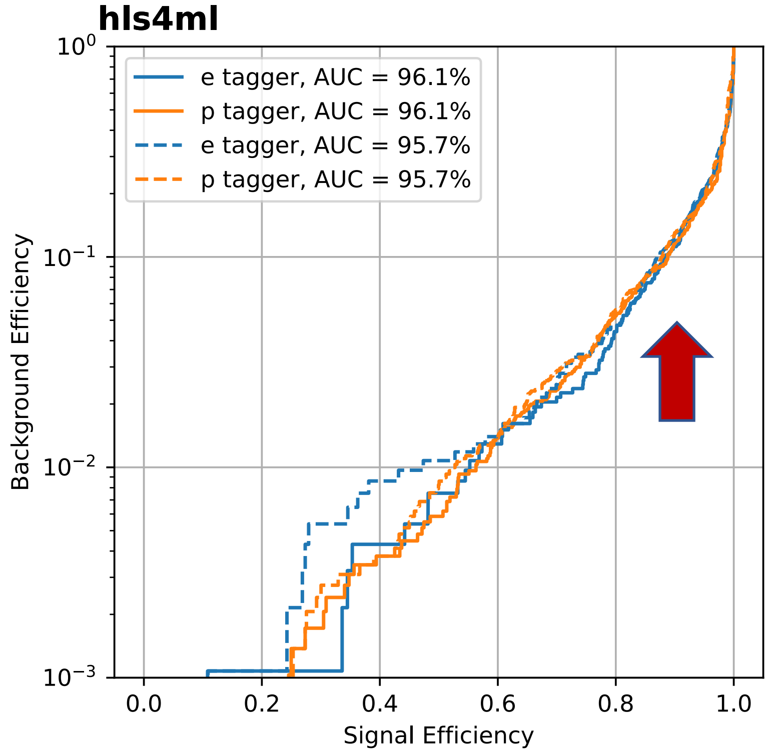



```
================================================================
== Performance Estimates
================================================================
+ Timing (ns):
    * Summary:
    +--------+--------+----------+------------+
    |  Clock | Target | Estimated| Uncertainty|
    +--------+--------+----------+------------+
    |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%

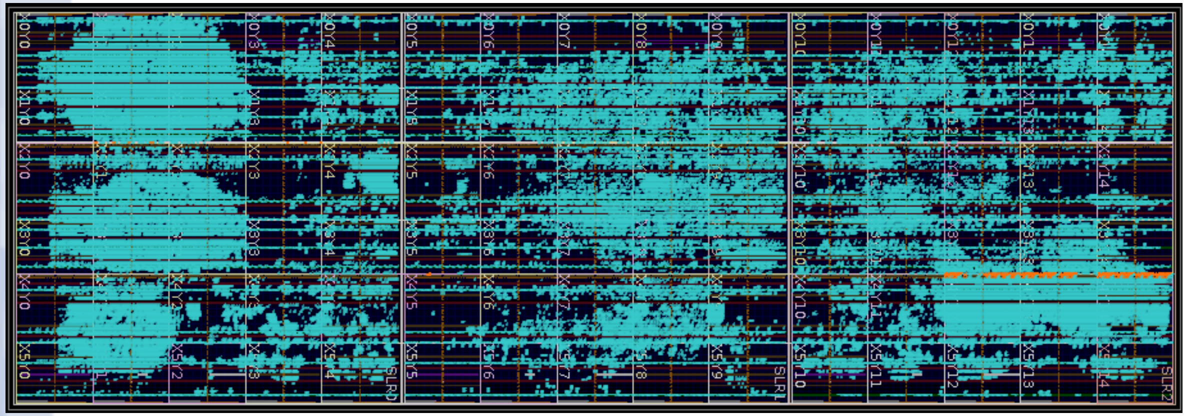
FPGA test bench

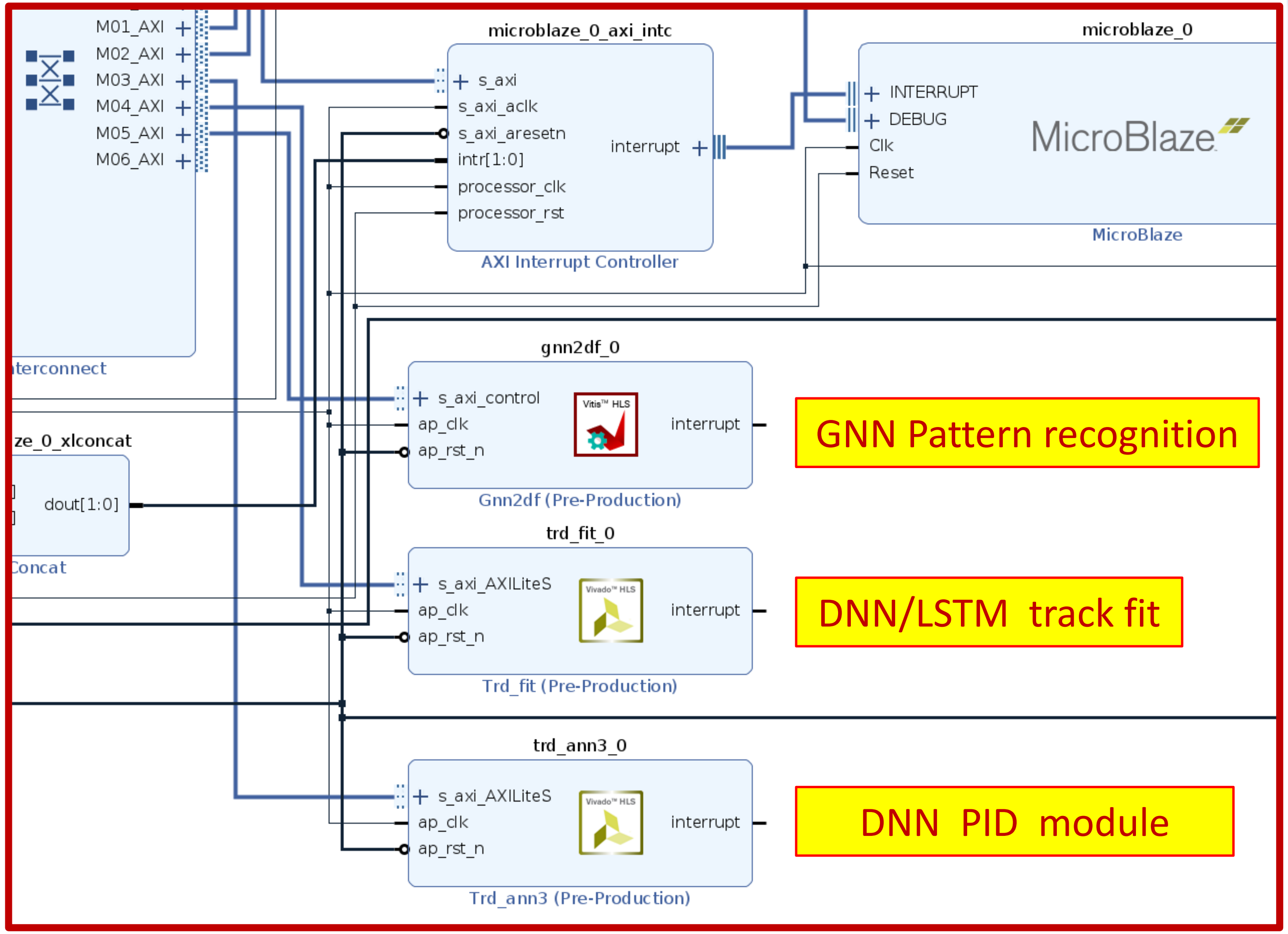
Jefferson Lab
Thomas Jefferson National Accelerator Facility

- 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.

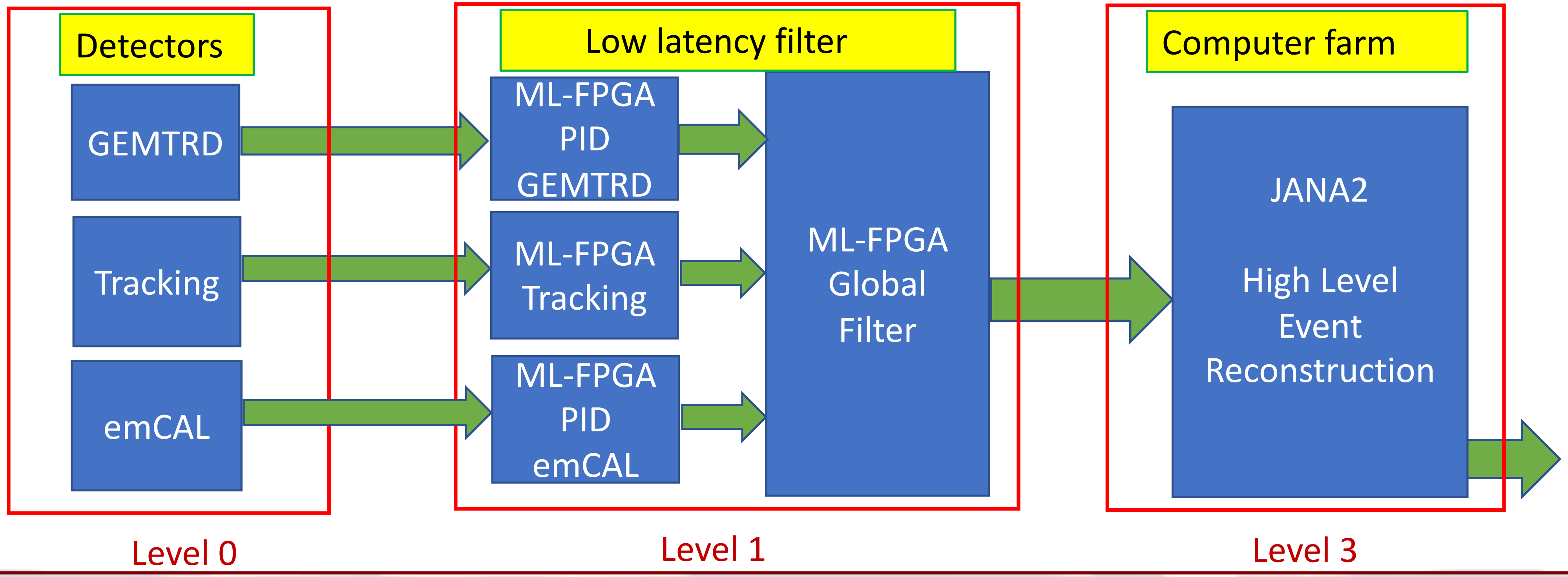
	GNN	LSTM	DNN	CNN	GarNet
Clock, ns	5	5	5	5	5
Latency, clocks	278	239	13	260	5643
Interval, clocks	279	234	1	245	5643
Latency, ns	1390	1195	65	1300	23215
Utilization DSP (%)	68	27	3	71	3





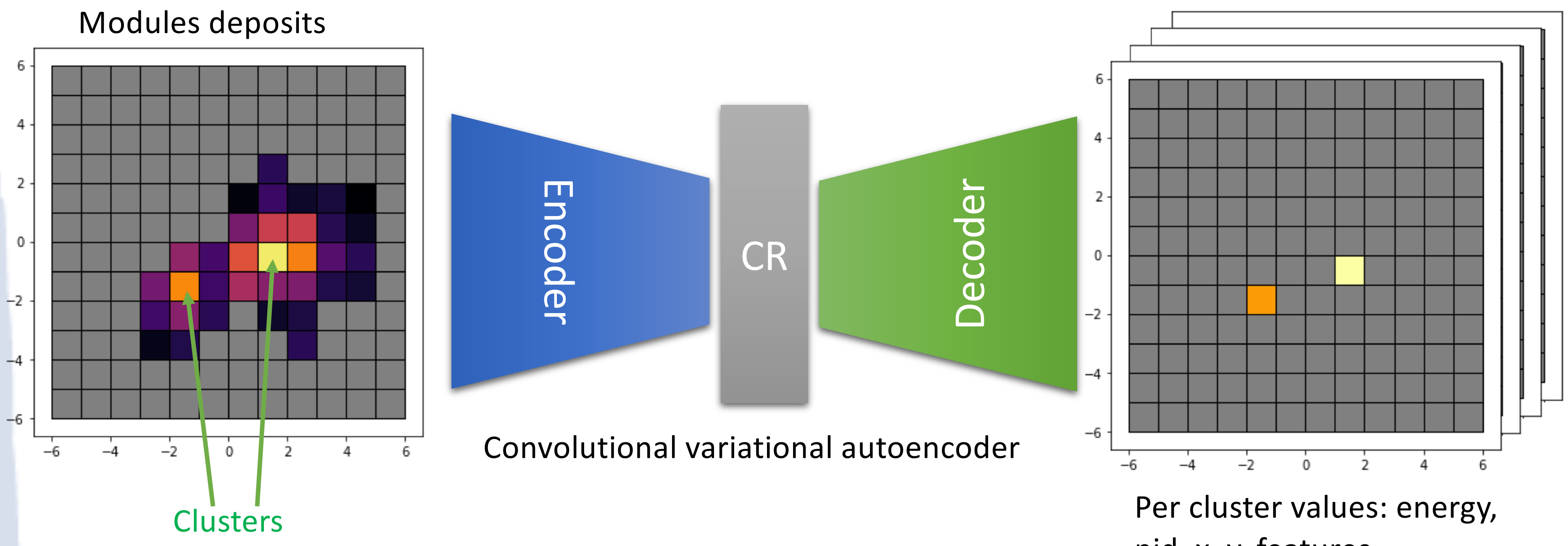
Global PID

- *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.*



Calorimeter parameters reconstruction

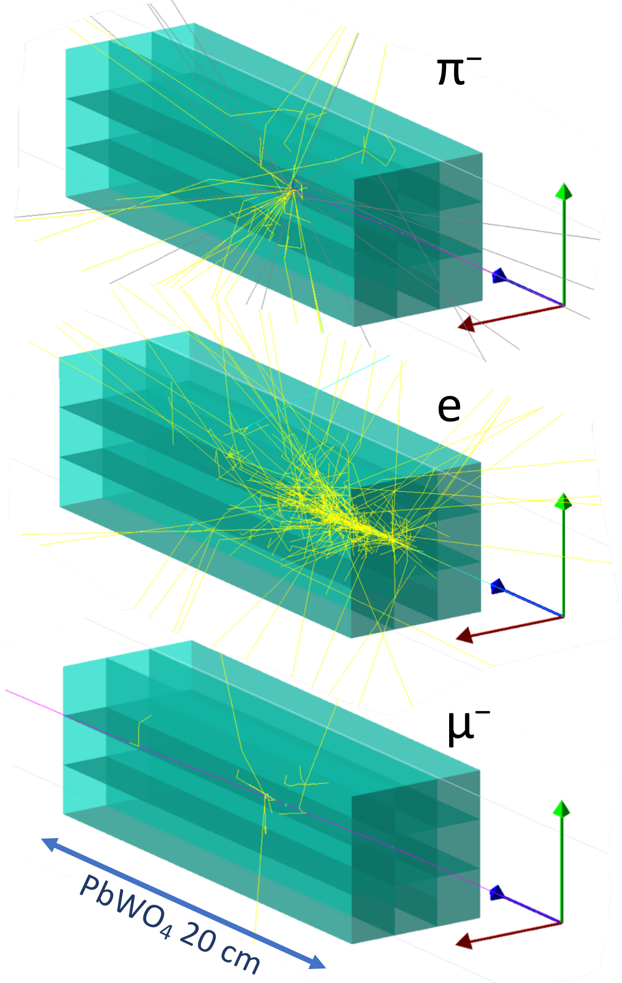
By Dmitry Romanov



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

CNN for calorimeter reconstruction

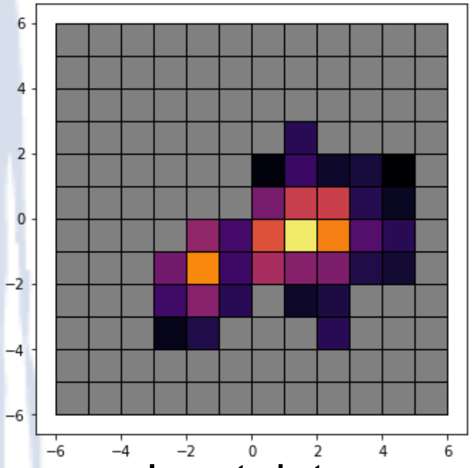
Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *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*

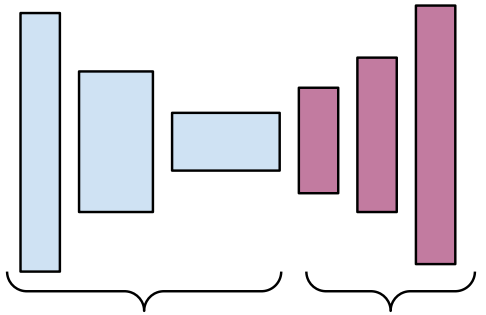
Actual values	Predicted results	
	e	π
e	98.8 %	1.2 %
π	2.9 %	97.1 %

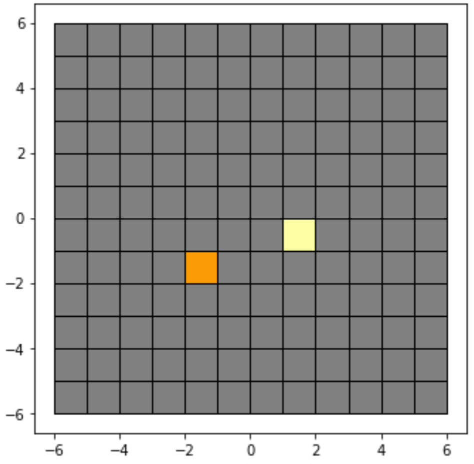
```
+ Timing (ns):
    * Summary:
    +---------+-------+----------+------------+
    |  Clock  | Target| Estimated| Uncertainty|
    +---------+-------+----------+------------+
    |ap_clk   |   5.00|     4.292|        0.62|
    +---------+-------+----------+------------+

+ 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|
+-----------------------+---------+-------+--------+--------+-----+
```



Input data





output

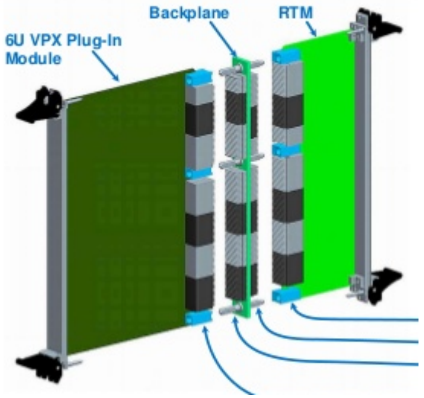
ADC based DAQ for PANDA STT

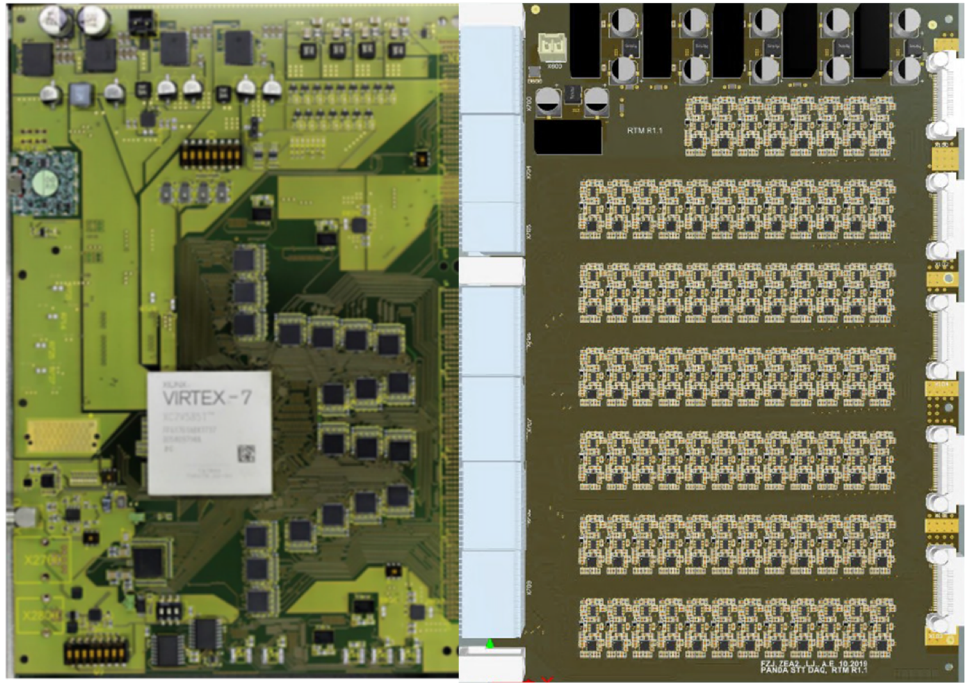
Jefferson Lab
Thomas Jefferson National Accelerator Facility

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, ..





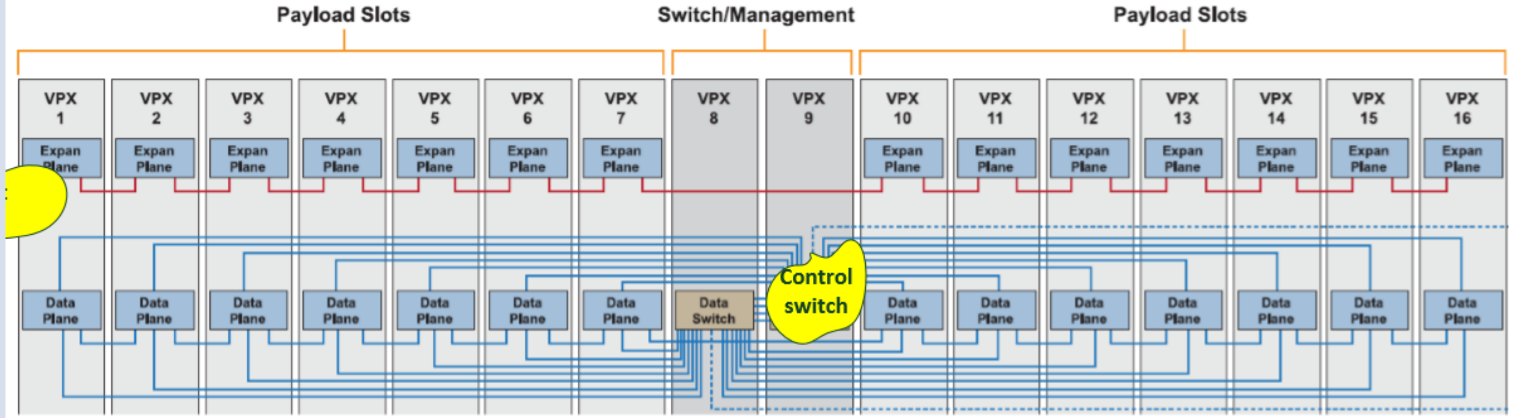
- 40 4-channel ADCs (configurable up to 1 GSPS);
- Single Virtex7 FPGA

- 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

Conclusion

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- 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.

Outlook

- 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 based on physics signatures.

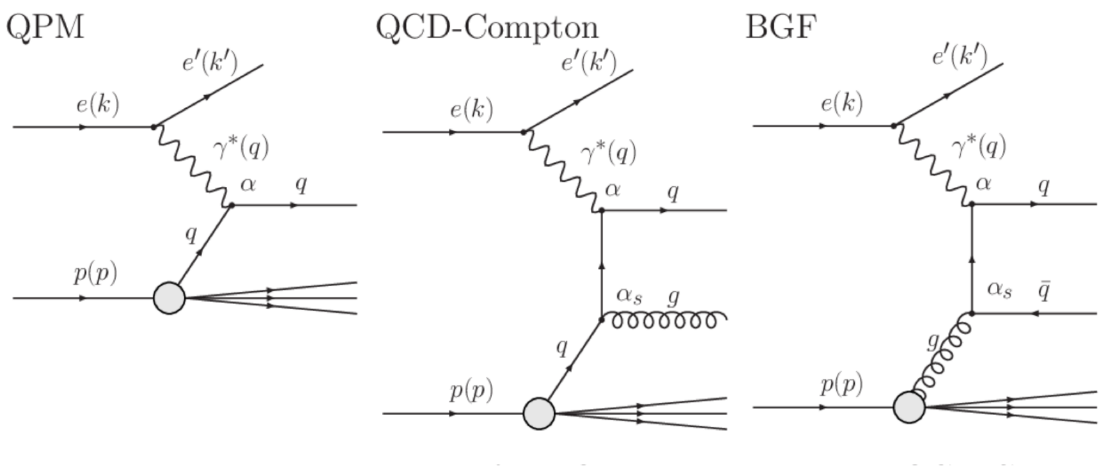
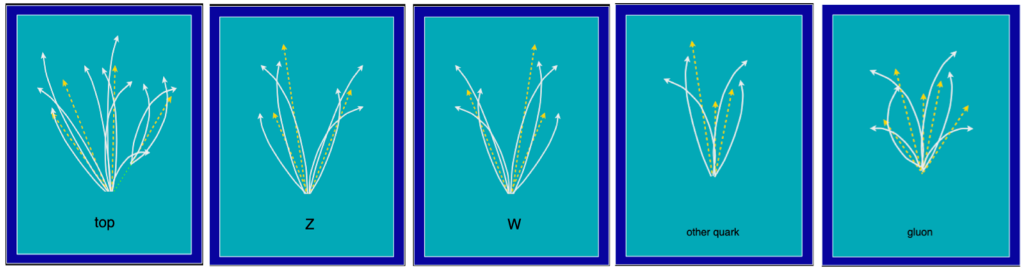


Figure 2.1: Feynman diagrams of the Quark Parton Model, QCD-Compton and Boson Gluon Fusion processes in NC DIS.

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..

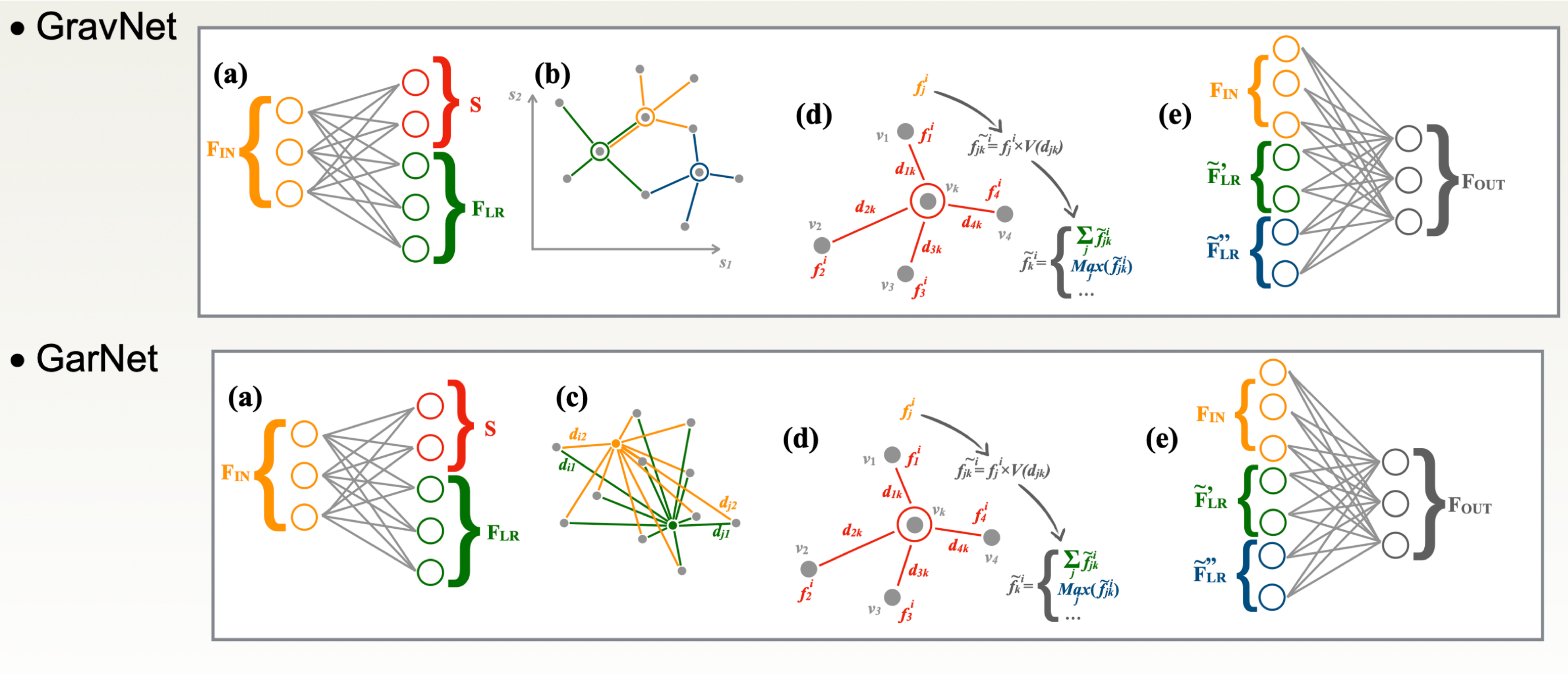
11.01.2019 Jennifer Ngadiuba - hls4ml: deep neural networks in FPGAs 25

Backup

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



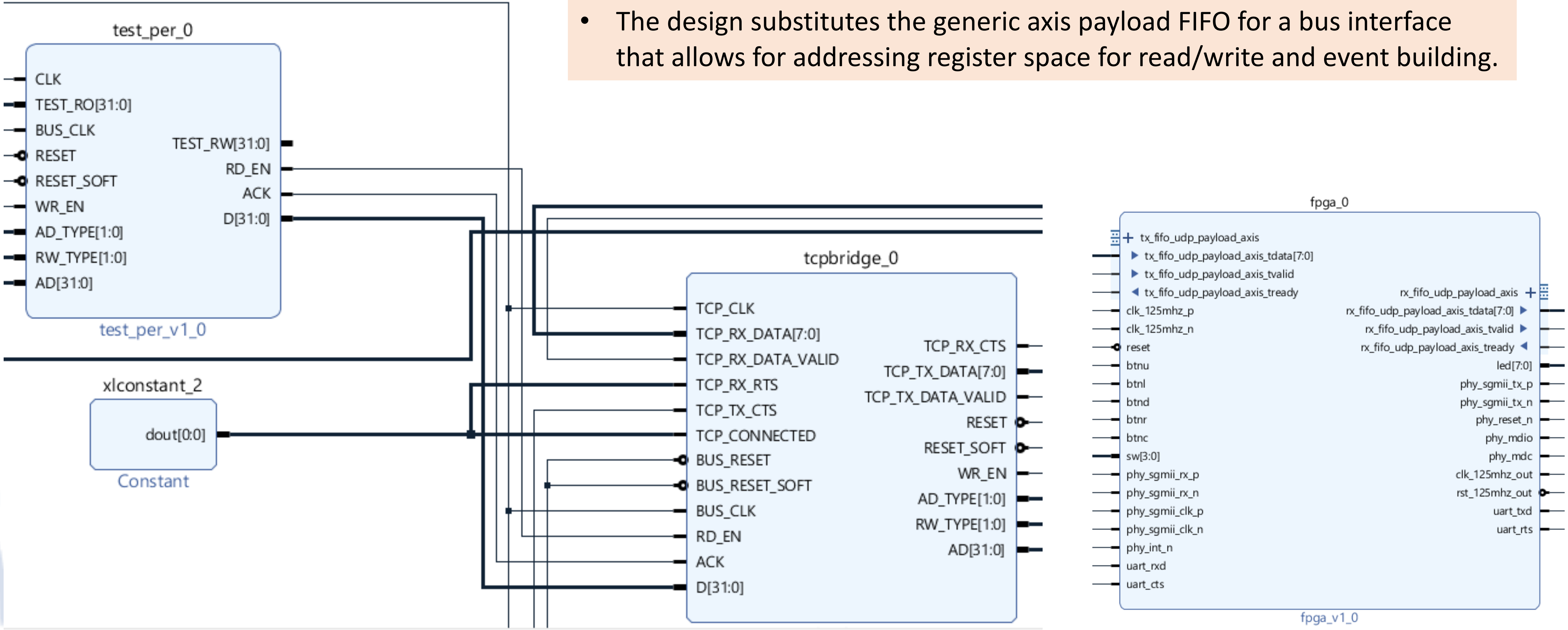
S.R. Qasim, J.K, Y. Iiyama, M Pierini arXiv:1902.07987, EPJC

Developing ethernet interface

Jefferson Lab
Thomas Jefferson National Accelerator Facility

By Cody Dickover

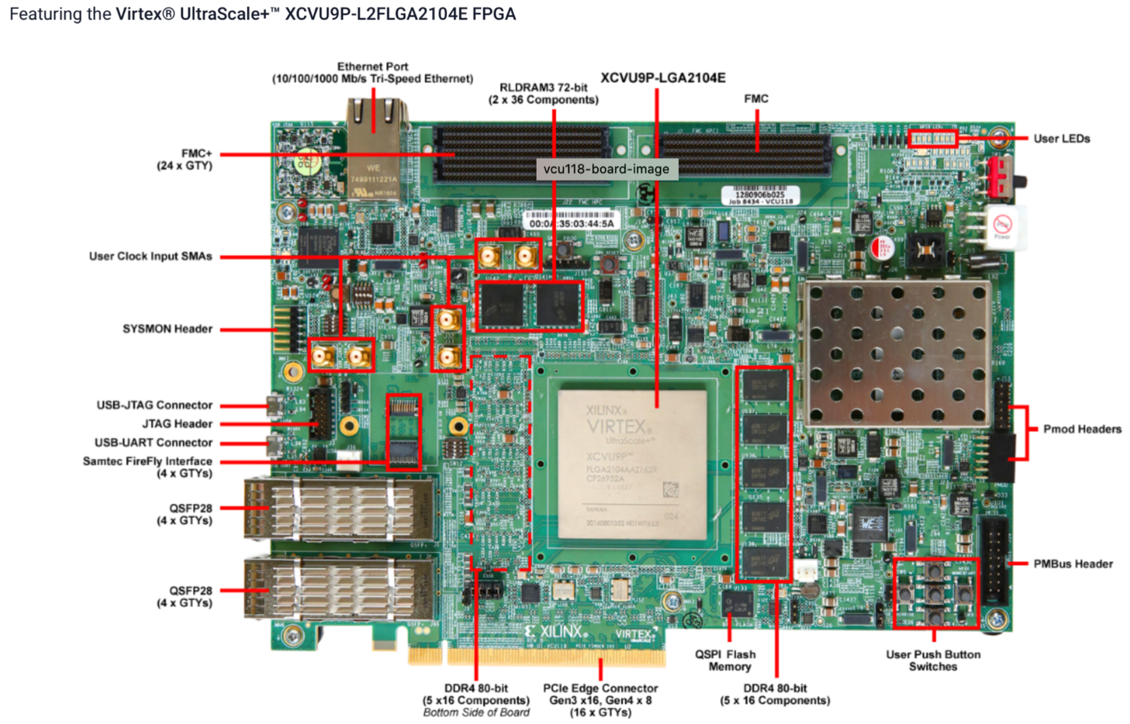
- Currently we using Microblaze setup for tests.
- For the beam test we need high speed interface to Detector/FADC.
- The design substitutes the generic axis payload FIFO for a bus interface that allows for addressing register space for read/write and event building.



FPGA test board for ML

Jefferson Lab
Thomas Jefferson National Accelerator Facility

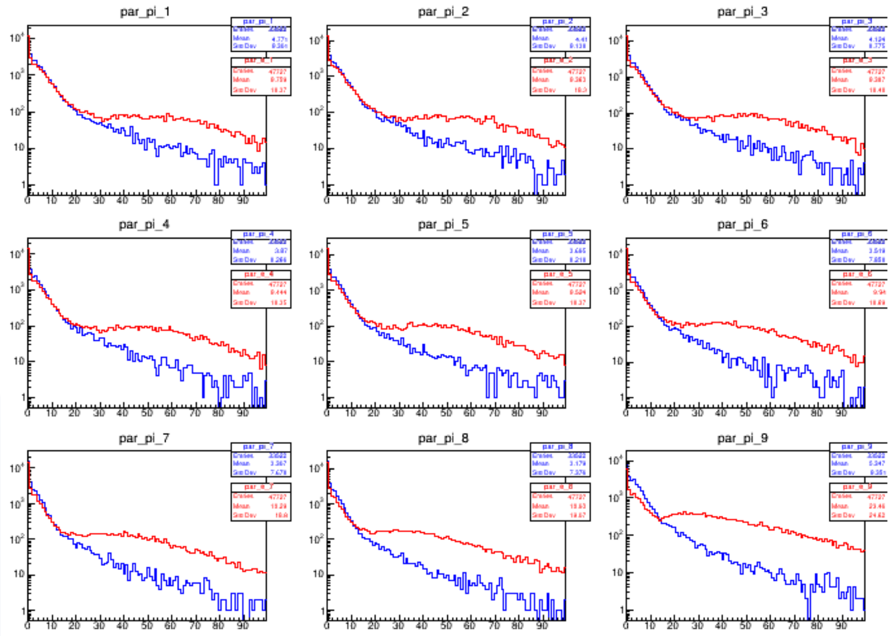
- At an early stage in this project, as hardware to test ML algorithms on FPGA , we use a standard Xilinx evaluation boards rather than developing a customized FPGA board. These boards have functions and interfaces sufficient for proof of principle of ML-FPGA.
- The Xilinx evaluation board includes the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

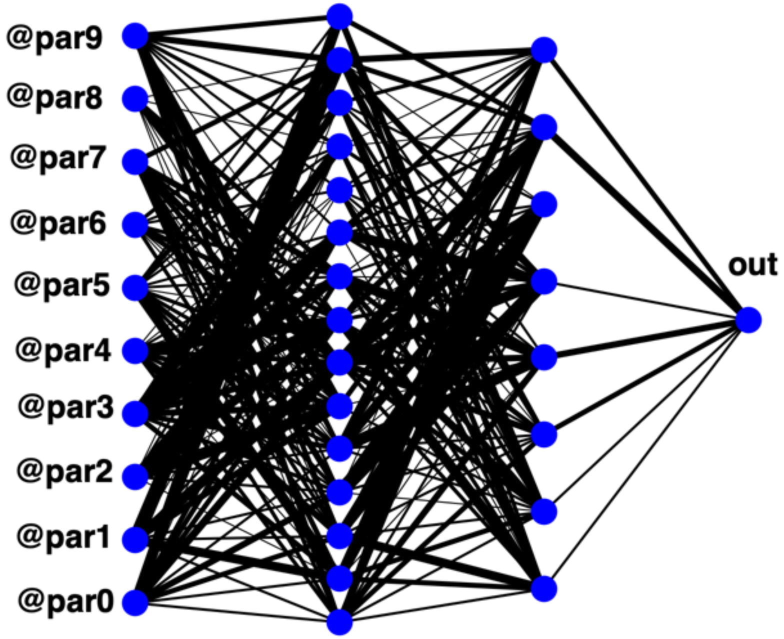


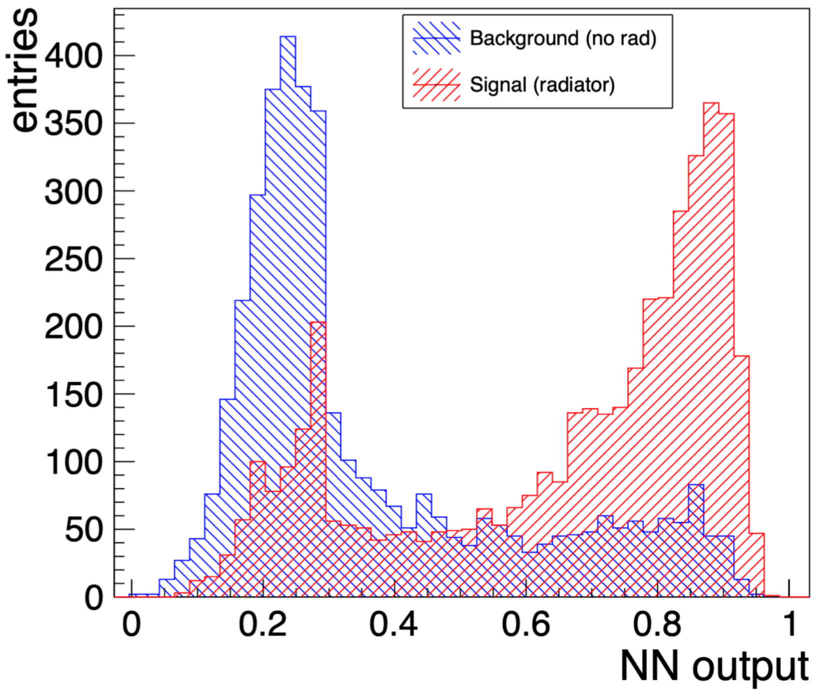
Xilinx Virtex® UltraScale+™

GEM-TRD offline analysis

Jefferson Lab
Thomas Jefferson National Accelerator Facility



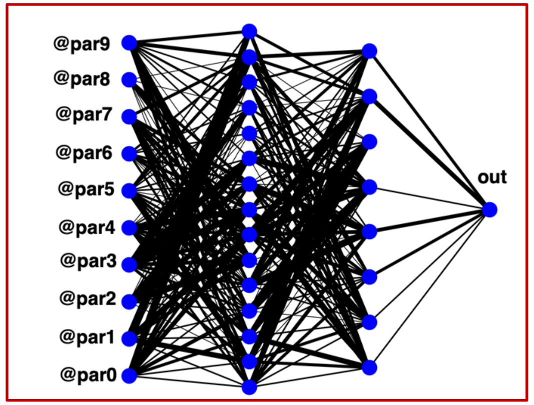




- *For data analysis we used a neural network library provided by root /TMVA package :*
  - MultiLayerPerceptron (MLP)
- *Top left plot shows ionization difference for e/pi in several bins along the track*
- *Top right plot shows neural network output for single TRD module:*
  - Red - electrons with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

Xilinx HLS: C++ to Verilog

Jefferson Lab
Thomas Jefferson National Accelerator Facility



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.

C++

```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;
}
```

Note: fixed point calculation

Thanks to Ben Raydo for help.

Verilog

```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;
```

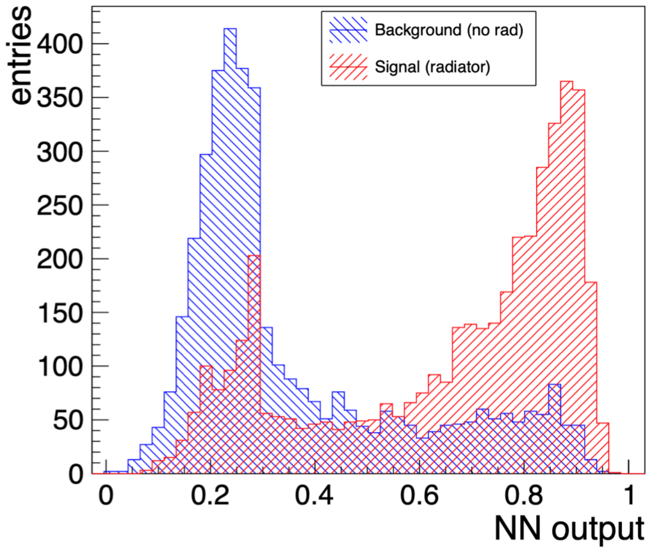
Test NN IP in FPGA

Jefferson Lab
Thomas Jefferson National Accelerator Facility

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 :

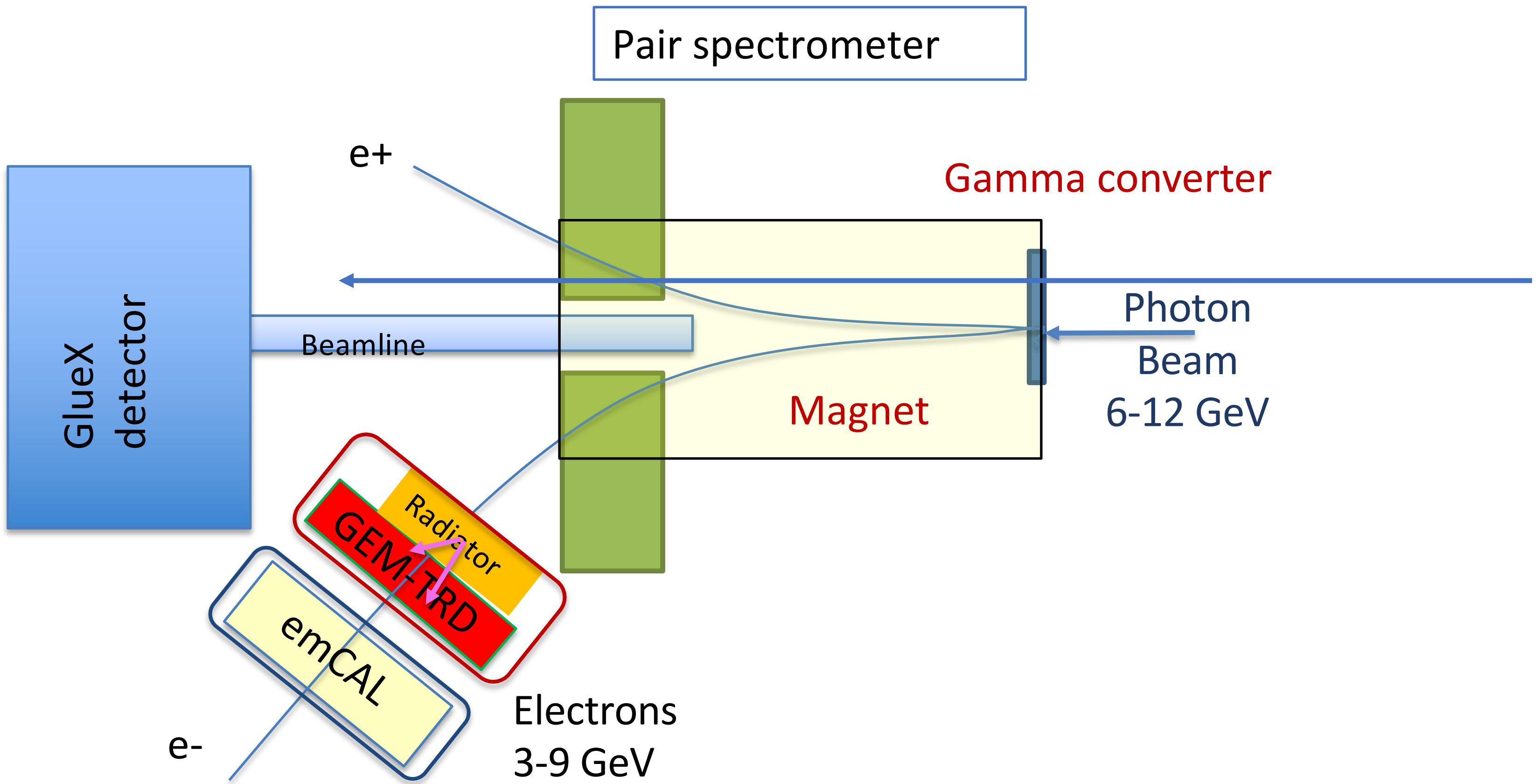
XTrdann ann; // create an instance of ML core.

```
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);
}
```

Beam setup at JLab Hall-D

Jefferson Lab
Thomas Jefferson National Accelerator Facility

- *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