Development of ML FPGA filter for particle identification in real time

Sergey Furletov
Jefferson Lab

<u>Team :</u>
F. Barbosa, L. Belfore, C. Dickover, C. Fanelli, D. Furletov,
S. Furletov, L. Jokhovets, D. Lawrence, D. Romanov

23rd IEEE Real Time Conference

3 Aug 2022

Outline

- *Motivation*
  - Hall-D/GlueX Detector
  - GEM-TRD
  - e/pi identification with TRD
- *Neural network(NN) for offline GEMRD data processing*
- *FPGA*
  - GNN for pattern recognition
  - RNN/LSTM for track fitting
  - DNN/GarNet for particle identification
- *PID with multiple detectors*
- *Outlook*

Motivation

❑ *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.*
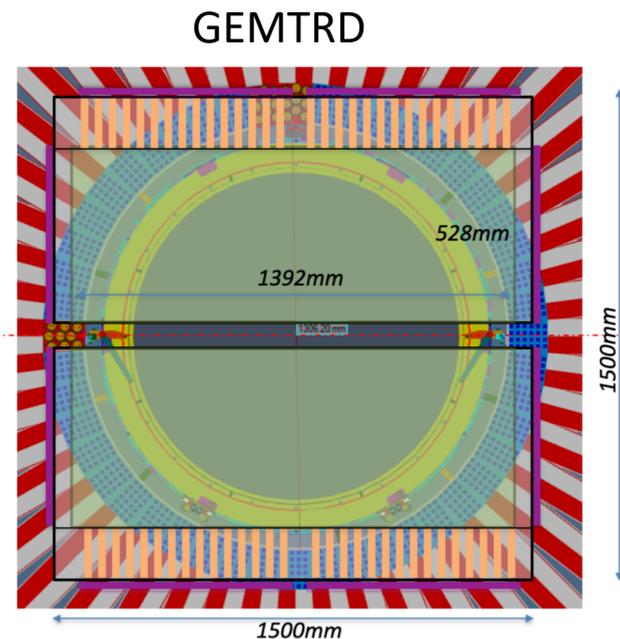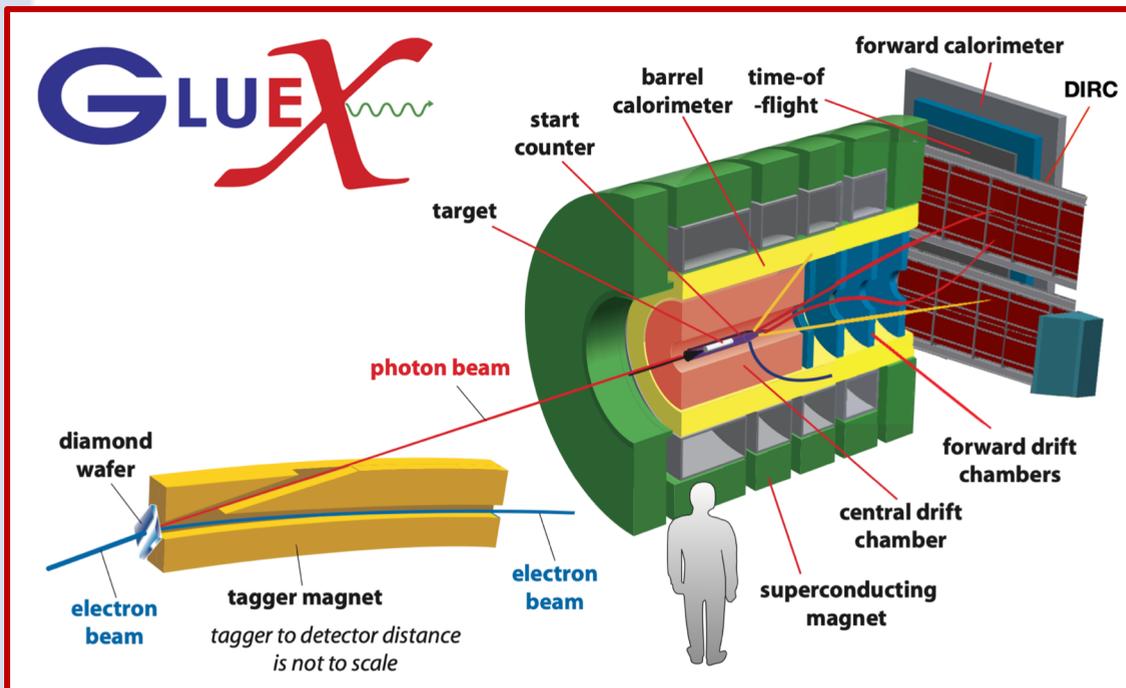
GEMTRD

❑ *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

- 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
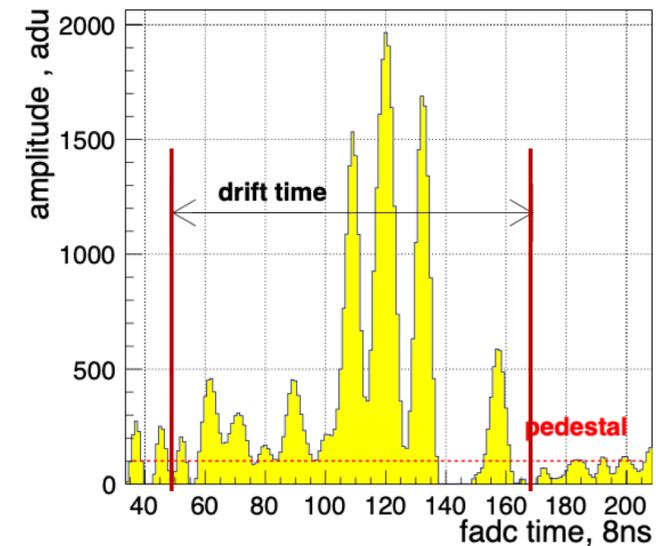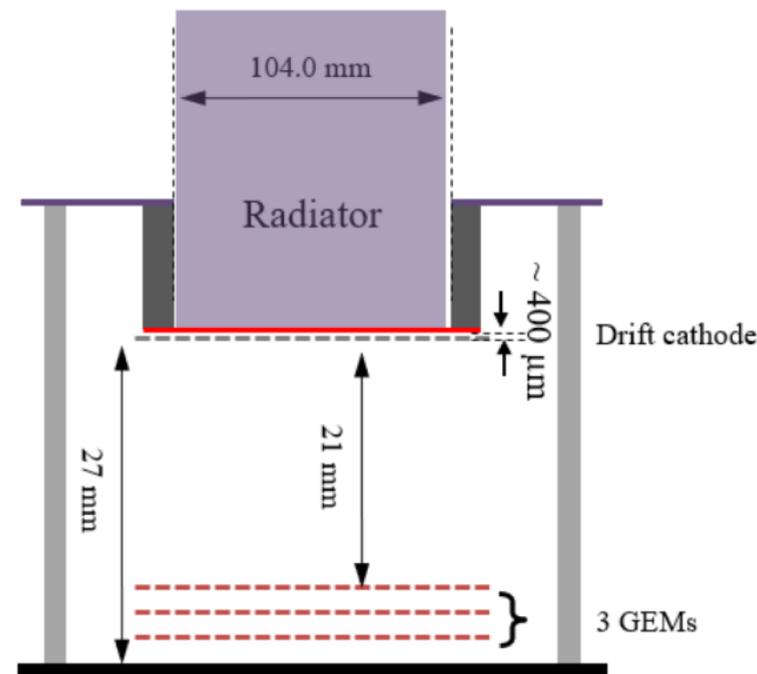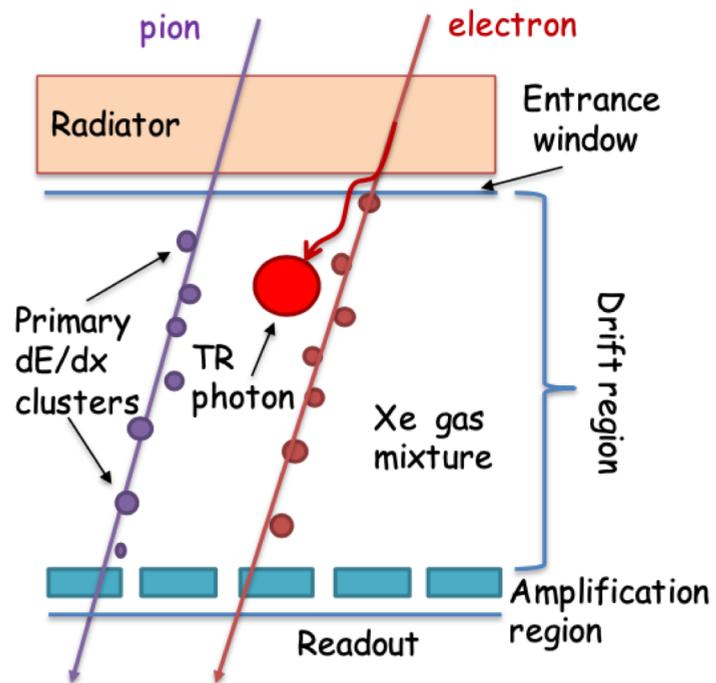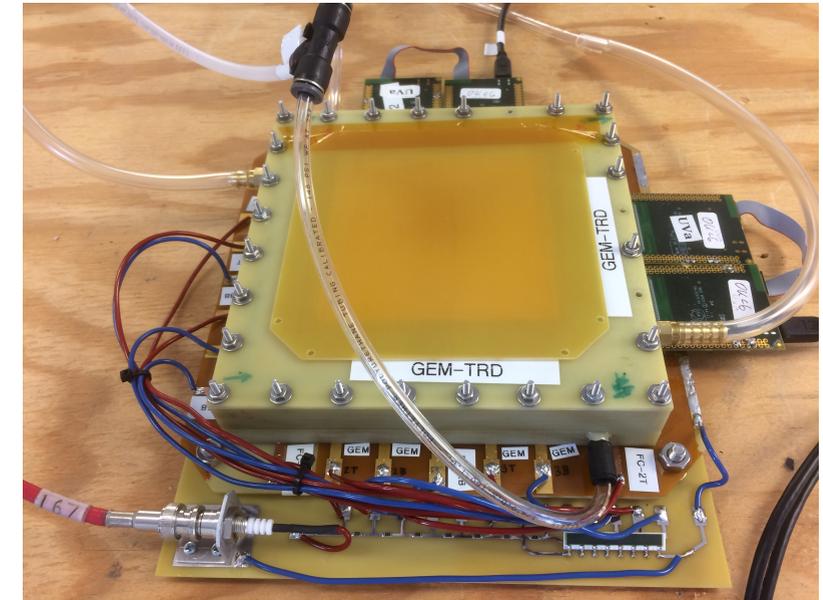
GEM-TRD principle

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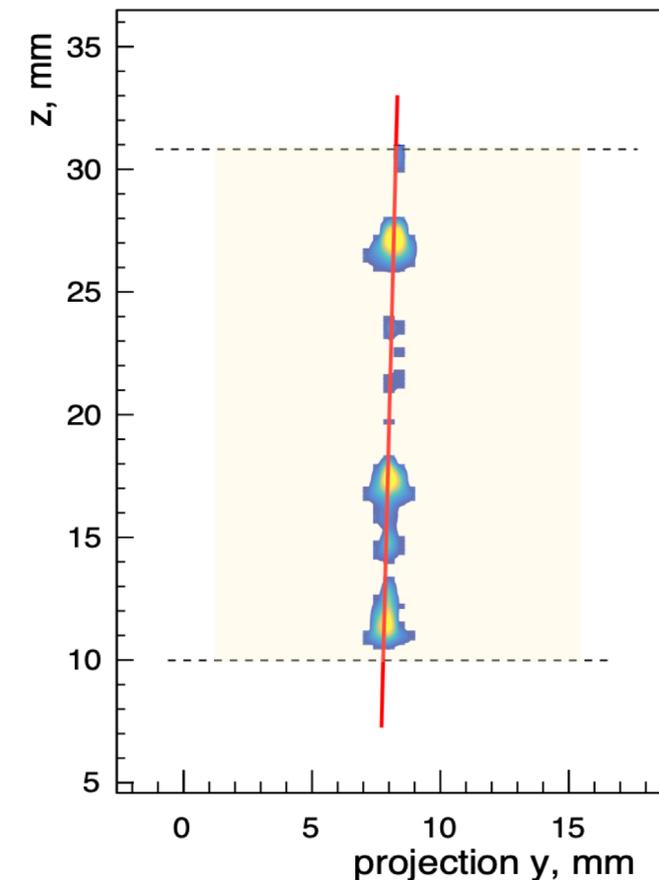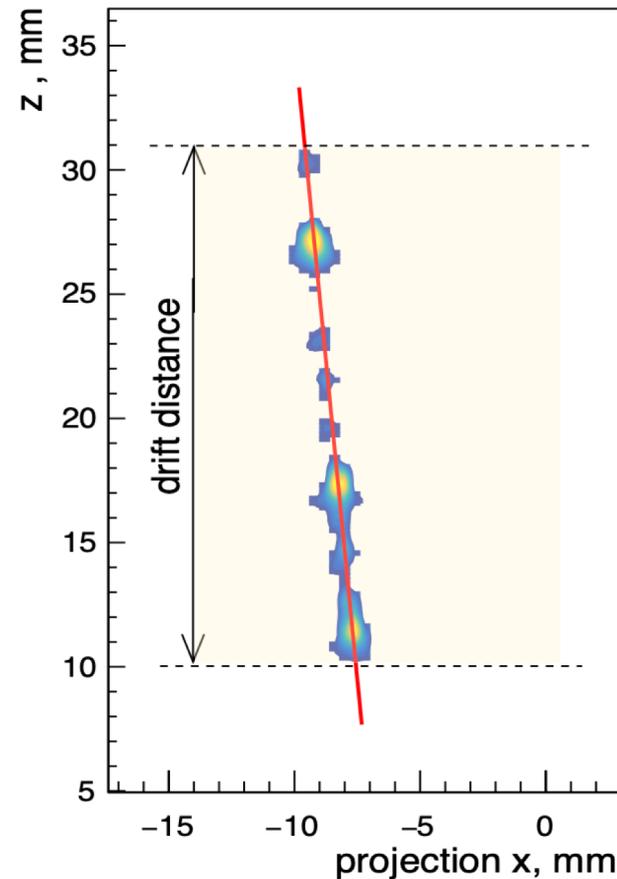
GEM-TRD offline analysis
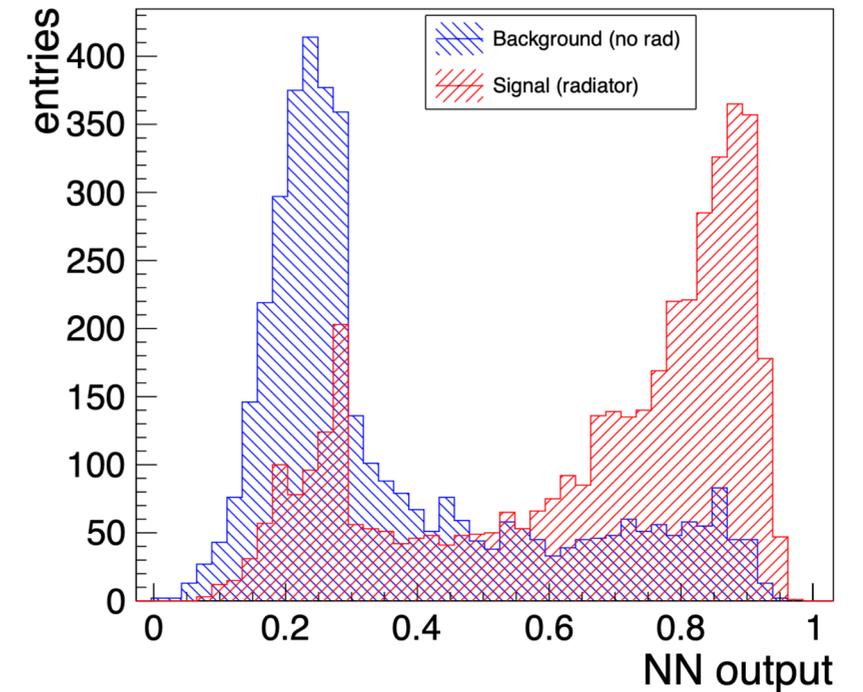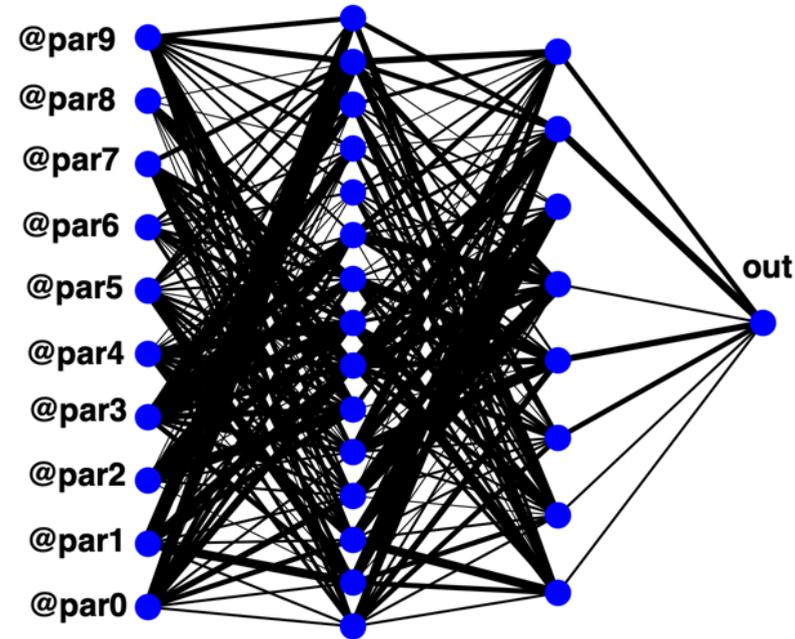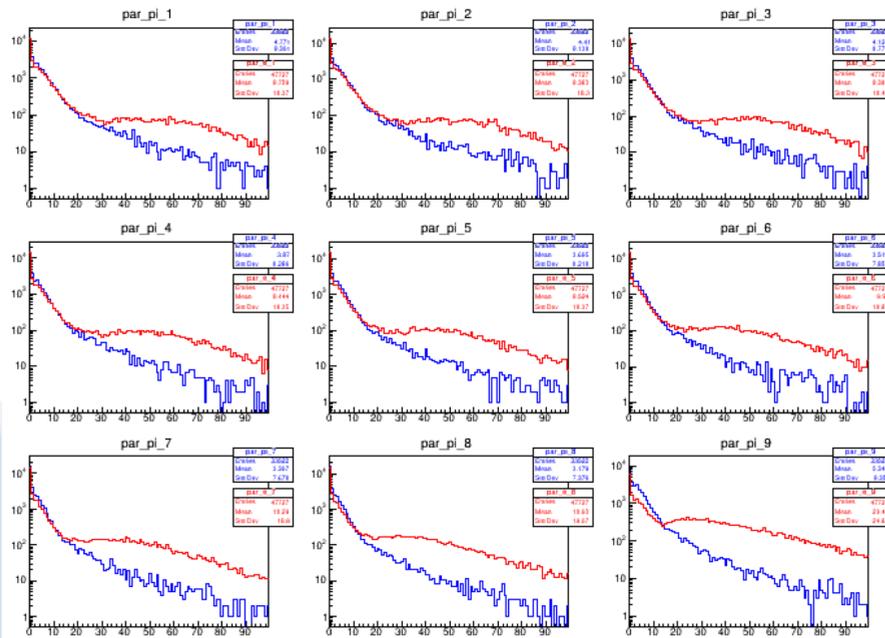
- ❑ *For data analysis we used a neural network library provided by root /TMVA package :*
  - ➢ MultiLayerPerceptron (MLP)
- ❑ *Top left plot shows ionization difference for e/pi in several bins along the track*
- ❑ *Top right plot shows neural network output for single TRD module:*
  - ➢ Red - electrons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

Moving forward : ML on FPGA

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

- *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

Inference on an FPGA
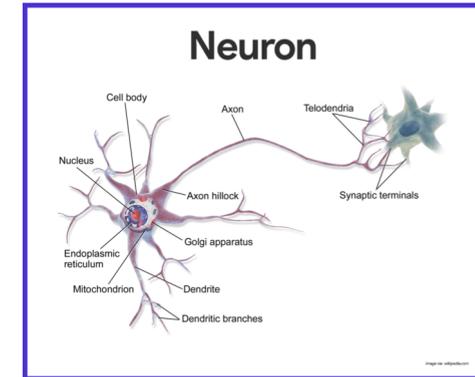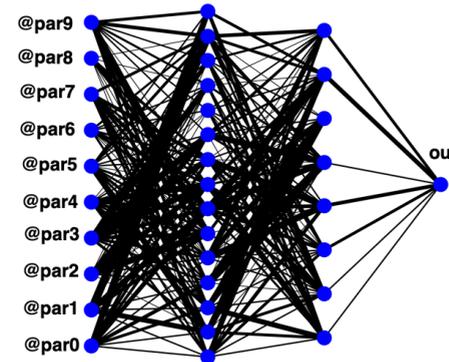
$$\vec{x}_m = g_m \left(W_{m,m-1}\vec{x}_{m-1} + \vec{b}_m \right)$$
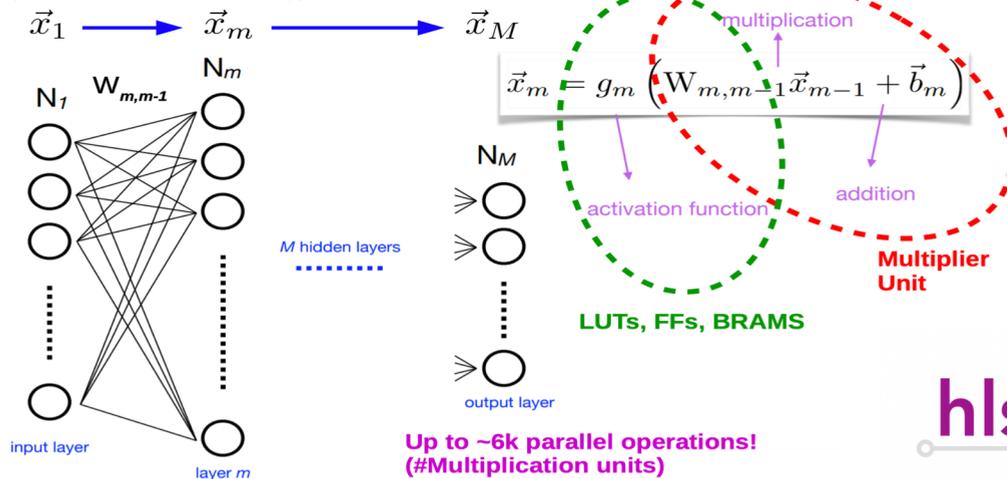
- Modern FPGAs have DSP slices - specialized hardware blocks placed between gateways and routers that perform mathematical calculations.
- The number of DSP slices can be up to 6000-12000 per chip.

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

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.

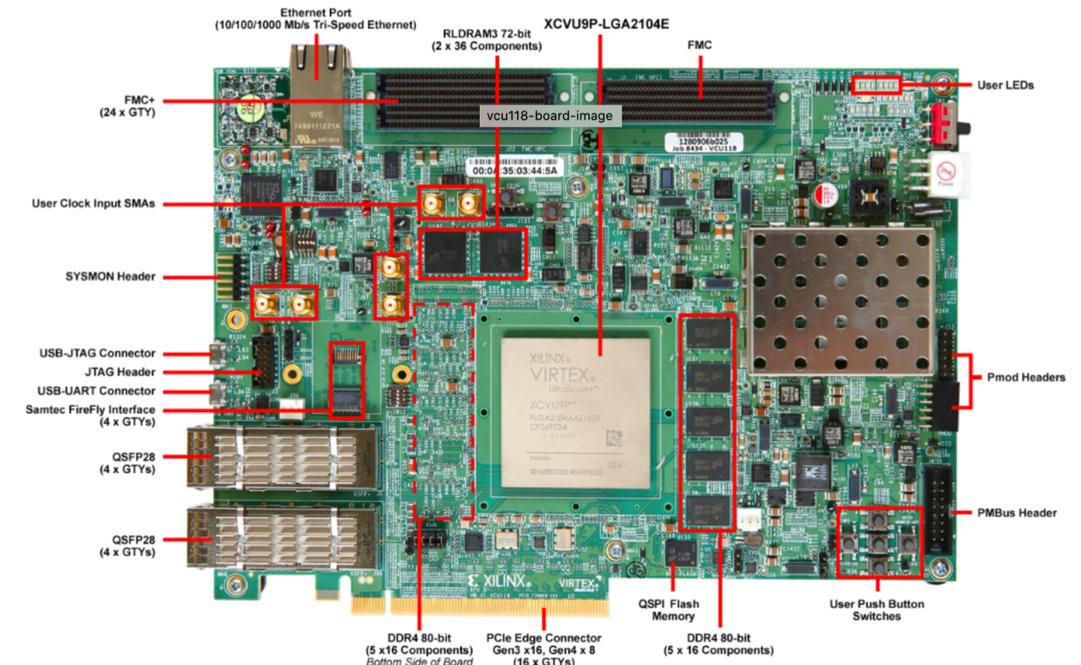- The Xilinx evaluation board includes the Xilinx XCVU9P and 6,840 DSP slices. Each includes a hardwired optimized multiply unit and collectively offers a peak theoretical performance in excess of 1 Tera multiplications per second.

- 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.

Xilinx Virtex® UltraScale+™

Xilinx HLS: C++ to Verilog
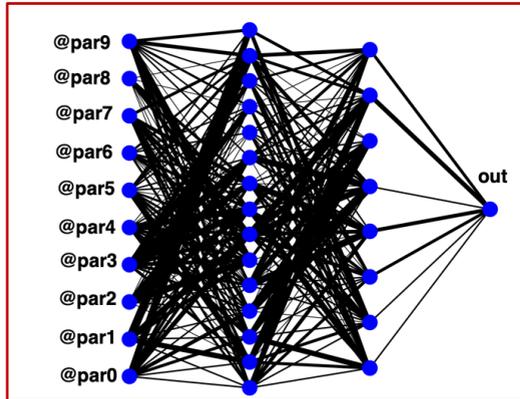
The C/C++ code of the trained network is used as input for Vivado_HLS.

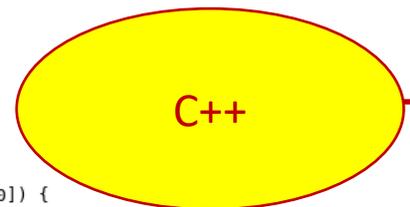The Xilinx Vivado HLS (High-Level Synthesis) tool provides a higher level of abstraction for the user by synthesizing functions written in C,C++ into IP blocks, by generating the appropriate ,low-level, VHDL and Verilog code. Then those blocks can be integrated into a real hardware system.

```
1  //--------------------------------
2  // float_regex.sh:: converted to (tx_t)
3  //--------------------------------
4  //---------- cxx file ---------
5  #include "trd_ann.h"
6  #include <cmath>
7  /*
8  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,
9    input0 = (in0 - (fx_t)1.96805)/(fx_t)7.63362;
10   input1 = (in1 - (fx_t)4.75766)/(fx_t)11.9138;
11   input2 = (in2 - (fx_t)4.40589)/(fx_t)11.4831;
12   input3 = (in3 - (fx_t)4.24519)/(fx_t)11.2533;
13   input4 = (in4 - (fx_t)4.30175)/(fx_t)11.2252;
14   input5 = (in5 - (fx_t)3.87414)/(fx_t)10.1781;
15   input6 = (in6 - (fx_t)3.75959)/(fx_t)9.69367;
16   input7 = (in7 - (fx_t)3.84352)/(fx_t)9.66213;
17   input8 = (in8 - (fx_t)3.65047)/(fx_t)9.09565;
18   input9 = (in9 - (fx_t)5.96775)/(fx_t)11.3203;
19   switch(index) {
20   case 0:
21     return neuron0x32b4c90();
22   default:
23     return (fx_t)0.;
24   }
25  }
26  */
27  fout_t trdann(int index, finp_t input[10]) {
28    input0 = (fx_t(input[0]) - (fx_t)1.96805)/(fx_t)7.63362;
29    input1 = (fx_t(input[1]) - (fx_t)4.75766)/(fx_t)11.9138;
30    input2 = (fx_t(input[2]) - (fx_t)4.40589)/(fx_t)11.4831;
31    input3 = (fx_t(input[3]) - (fx_t)4.24519)/(fx_t)11.2533;
32    input4 = (fx_t(input[4]) - (fx_t)4.30175)/(fx_t)11.2252;
33    input5 = (fx_t(input[5]) - (fx_t)3.87414)/(fx_t)10.1781;
34    input6 = (fx_t(input[6]) - (fx_t)3.75959)/(fx_t)9.69367;
35    input7 = (fx_t(input[7]) - (fx_t)3.84352)/(fx_t)9.66213;
36    input8 = (fx_t(input[8]) - (fx_t)3.65047)/(fx_t)9.09565;
37    input9 = (fx_t(input[9]) - (fx_t)5.96775)/(fx_t)11.3203;
38    switch(index) {
39    case 0:
40      return neuron0x32b4c90();
41    default:
42      return (fx_t)0.;
43    }
44  }
45
46  fx_t neuron0x32bf850() {
47    return input0;
48  }
49
50  fx_t neuron0x32bf190() {
51    return input1;
52  }
53
54  fx_t neuron0x32bf4d0() {
55    return input2;
56  }
```

Note: fixed point calculation

Thanks to Ben Raydo for help.

```
1  // ============================================================
2  // RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
3  // Version: 2019.1
4  // Copyright (C) 1986-2019 Xilinx, Inc. All Rights Reserved.
5  //
6  // ============================================================
7
8  `timescale 1 ns / 1 ps
9
10 (* CORE_GENERATION_INFO="trdann,hls_ip_2019_1,{HLS_INPUT_TYPE=cxx,HLS_INPUT_FLOAT=1
11
12 module trdann (
13         ap_clk,
14         ap_rst_n,
15         s_axi_AXILiteS_AWVALID,
16         s_axi_AXILiteS_AWREADY,
17         s_axi_AXILiteS_AWADDR,
18         s_axi_AXILiteS_WVALID,
19         s_axi_AXILiteS_WREADY,
20         s_axi_AXILiteS_WDATA,
21         s_axi_AXILiteS_WSTRB,
22         s_axi_AXILiteS_ARVALID,
23         s_axi_AXILiteS_ARREADY,
24         s_axi_AXILiteS_ARADDR,
25         s_axi_AXILiteS_RVALID,
26         s_axi_AXILiteS_RREADY,
27         s_axi_AXILiteS_RDATA,
28         s_axi_AXILiteS_RRESP,
29         s_axi_AXILiteS_BVALID,
30         s_axi_AXILiteS_BREADY,
31         s_axi_AXILiteS_BRESP,
32         interrupt
33 );
34
35 parameter    ap_ST_fsm_state1 = 23'd1;
36 parameter    ap_ST_fsm_state2 = 23'd2;
37 parameter    ap_ST_fsm_state3 = 23'd4;
38 parameter    ap_ST_fsm_state4 = 23'd8;
39 parameter    ap_ST_fsm_state5 = 23'd16;
40 parameter    ap_ST_fsm_state6 = 23'd32;
41 parameter    ap_ST_fsm_state7 = 23'd64;
42 parameter    ap_ST_fsm_state8 = 23'd128;
43 parameter    ap_ST_fsm_state9 = 23'd256;
44 parameter    ap_ST_fsm_state10 = 23'd512;
45 parameter    ap_ST_fsm_state11 = 23'd1024;
46 parameter    ap_ST_fsm_state12 = 23'd2048;
47 parameter    ap_ST_fsm_state13 = 23'd4096;
48 parameter    ap_ST_fsm_state14 = 23'd8192;
49 parameter    ap_ST_fsm_state15 = 23'd16384;
50 parameter    ap_ST_fsm_state16 = 23'd32768;
51 parameter    ap_ST_fsm_state17 = 23'd65536;
52 parameter    ap_ST_fsm_state18 = 23'd131072;
53 parameter    ap_ST_fsm_state19 = 23'd262144;
54 parameter    ap_ST_fsm_state20 = 23'd524288;
55 parameter    ap_ST_fsm_state21 = 23'd1048576;
```
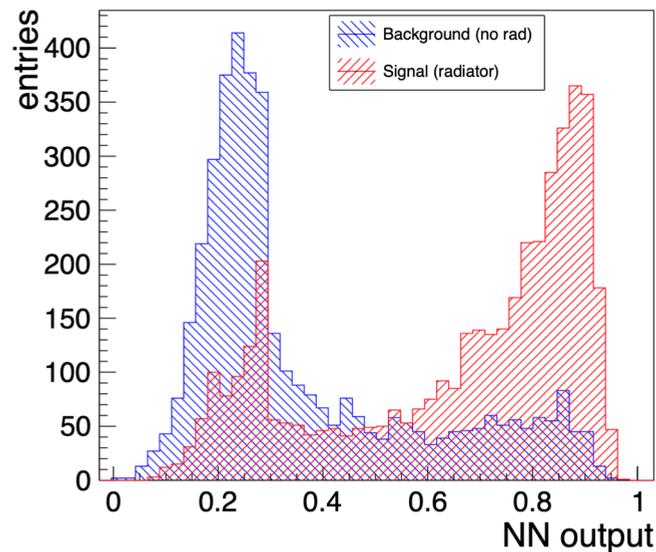
C++ → Verilog

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.

```cpp
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);
}
```
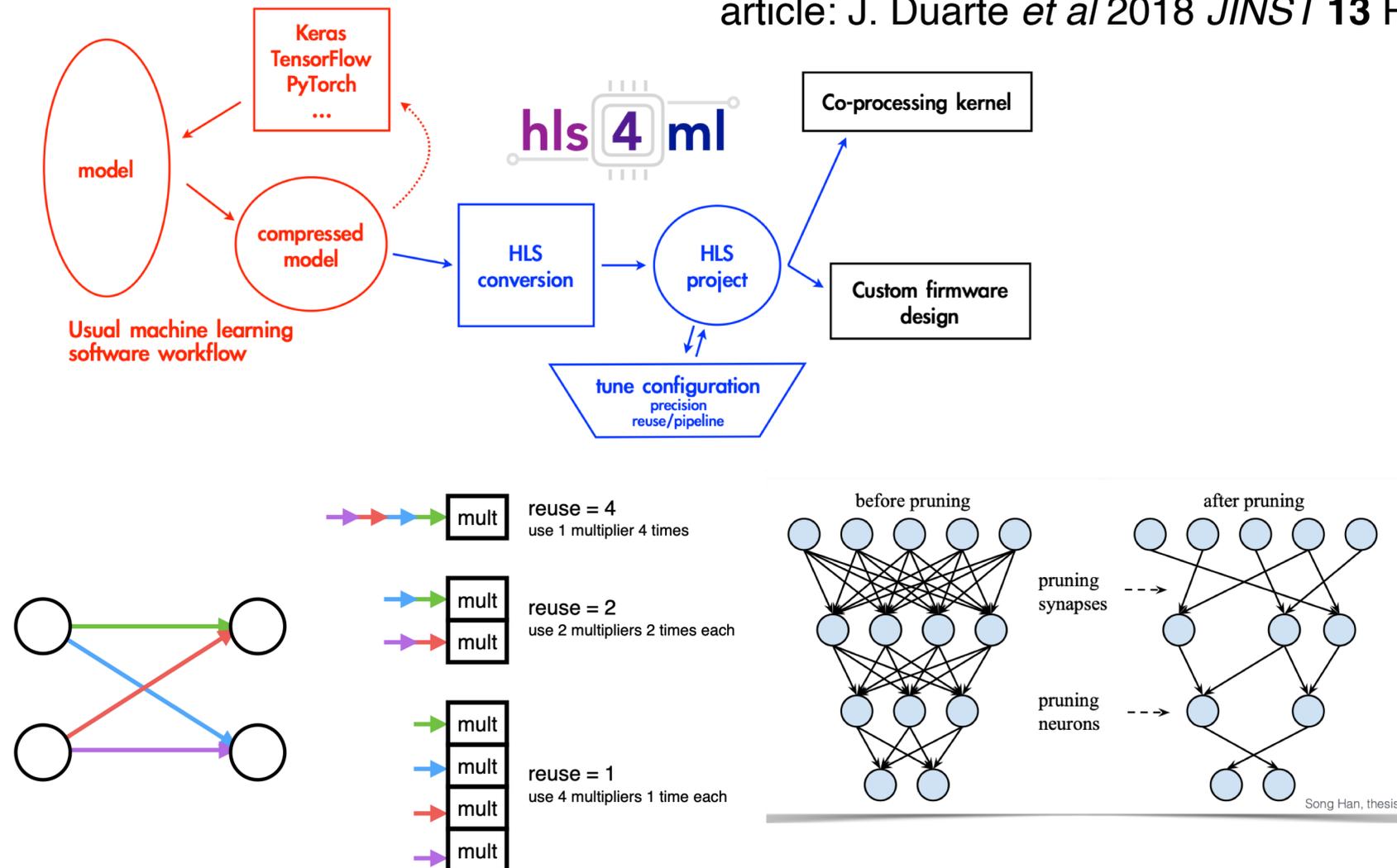
Optimization with hls4ml package

- 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

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Full size neural network, accuracy-optimized.

Size-optimized neural network

```
+ 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    |
    +----------------+----------------+-----------+
    |   15 |   15    |   1  |   1     | function  |
    +----------------+----------------+-----------+
```

Latency = 75ns

```
+--------------+--------+-------+--------+--------+------+
|     Name     |BRAM_18K| DSP48E|   FF   |  LUT   | URAM |
+--------------+--------+-------+--------+--------+------+
|DSP           |      - |     2 |      - |      - |    - |
|Expression    |      - |     - |      0 |     24 |    - |
|FIFO          |      - |     - |      - |      - |    - |
|Instance      |     19 |   692 |   3737 |  16446 |    - |
|Memory        |      2 |     - |      0 |      0 |    - |
|Multiplexer   |      - |     - |      - |     36 |    - |
|Register      |      - |     - |   1532 |      - |    - |
+--------------+--------+-------+--------+--------+------+
|Total         |     21 |   694 |   5269 |  16506 |    0 |
+--------------+--------+-------+--------+--------+------+
|Available SLR |   1440 |  2280 | 788160 | 394080 |  320 |
+--------------+--------+-------+--------+--------+------+
|Utilization SLR (%)|   1 |    30 |    ~0  |      4 |    0 |
+--------------+--------+-------+--------+--------+------+
|Available     |   4320 |  6840 |2364480 |1182240 |  960 |
+--------------+--------+-------+--------+--------+------+
|Utilization (%)|  ~0   |    10 |    ~0  |      1 |    0 |
+--------------+--------+-------+--------+--------+------+
```

DSP utilization 10%

```
+ Timing (ns):
  * Summary:
    +--------+-------+----------+------------+
    | Clock  | Target| Estimated| Uncertainty|
    +--------+-------+----------+------------+
    |ap_clk  |  5.00 |    3.883 |      0.62  |
    +--------+-------+----------+------------+

+ Latency (clock cycles):
  * Summary:
    +----------------+----------------+-----------+
    |    Latency     |    Interval    | Pipeline  |
    | min  |  max    | min  |  max    |   Type    |
    +----------------+----------------+-----------+
    |   17 |   17    |   3  |   3     | function  |
    +----------------+----------------+-----------+
```

Latency = 85ns

```
+--------------+--------+-------+--------+--------+------+
|     Name     |BRAM_18K| DSP48E|   FF   |  LUT   | URAM |
+--------------+--------+-------+--------+--------+------+
|DSP           |      - |     2 |      - |      - |    - |
|Expression    |      - |     - |      0 |     24 |    - |
|FIFO          |      - |     - |      - |      - |    - |
|Instance      |      - |   177 |   3132 |  10696 |    - |
|Memory        |      2 |     - |      0 |      0 |    - |
|Multiplexer   |      - |     - |      - |     81 |    - |
|Register      |      - |     - |   1423 |      - |    - |
+--------------+--------+-------+--------+--------+------+
|Total         |      2 |   179 |   4555 |  10801 |    0 |
+--------------+--------+-------+--------+--------+------+
|Available SLR |   1440 |  2280 | 788160 | 394080 |  320 |
+--------------+--------+-------+--------+--------+------+
|Utilization SLR (%)|  ~0 |    7 |    ~0  |      2 |    0 |
+--------------+--------+-------+--------+--------+------+
|Available     |   4320 |  6840 |2364480 |1182240 |  960 |
+--------------+--------+-------+--------+--------+------+
|Utilization (%)|  ~0   |     2 |    ~0  |    ~0  |    0 |
+--------------+--------+-------+--------+--------+------+
```

DSP utilization 2%

GEM-TRD tracks

❑ *In a real experiment, GEMTRD will have multiple tracks.*

❑ *So we also need a fast algorithm for pattern recognition in the FPGA*

❑ *As well as for track fitting.*
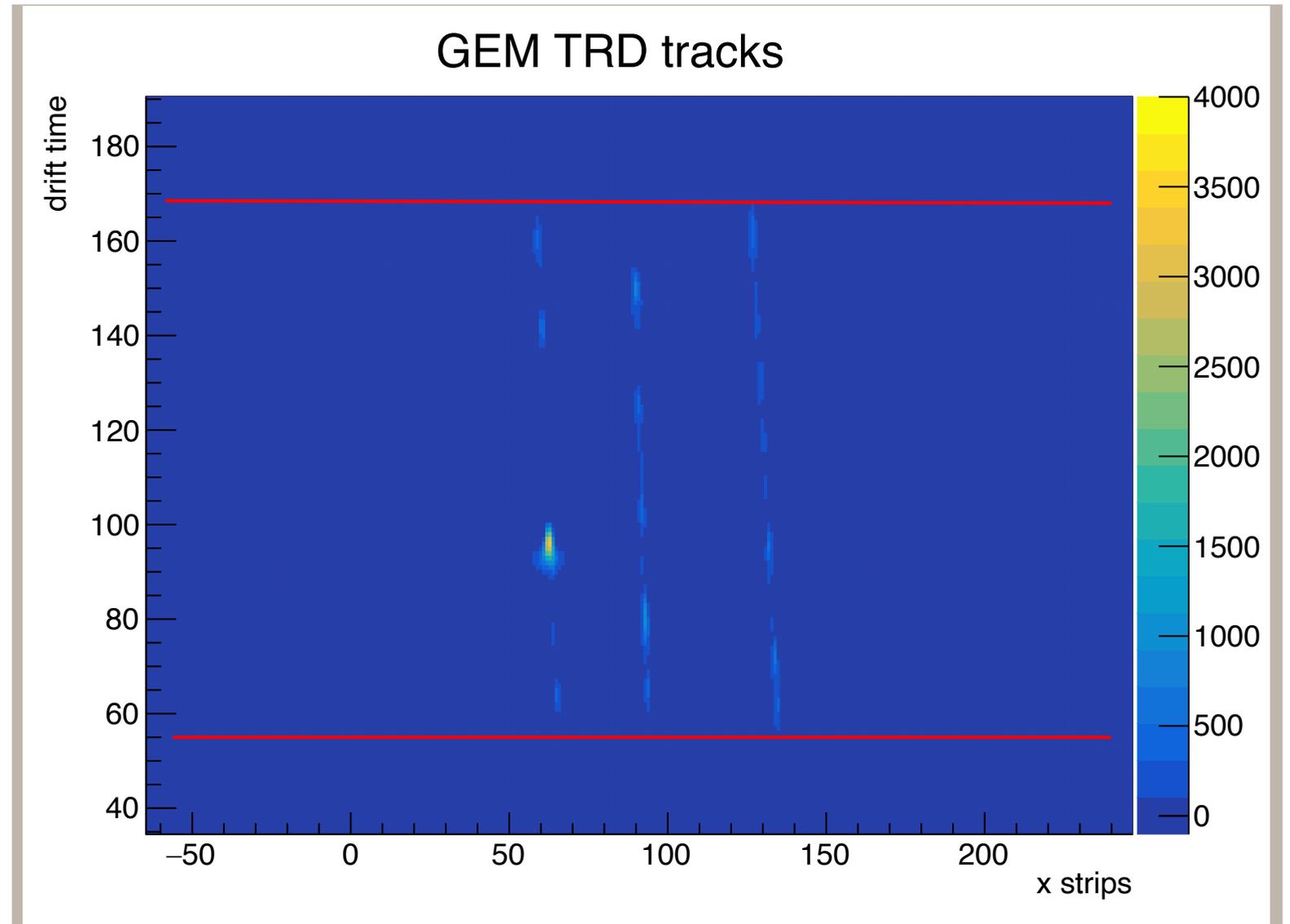
GEM TRD tracks

GEMTRD tracks

- ❑ *In a real experiment, GEMTRD will have multiple tracks.*
- ❑ *So we also need a fast algorithm for pattern recognition in the FPGA*
- ❑ *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

GEM TRD tracks
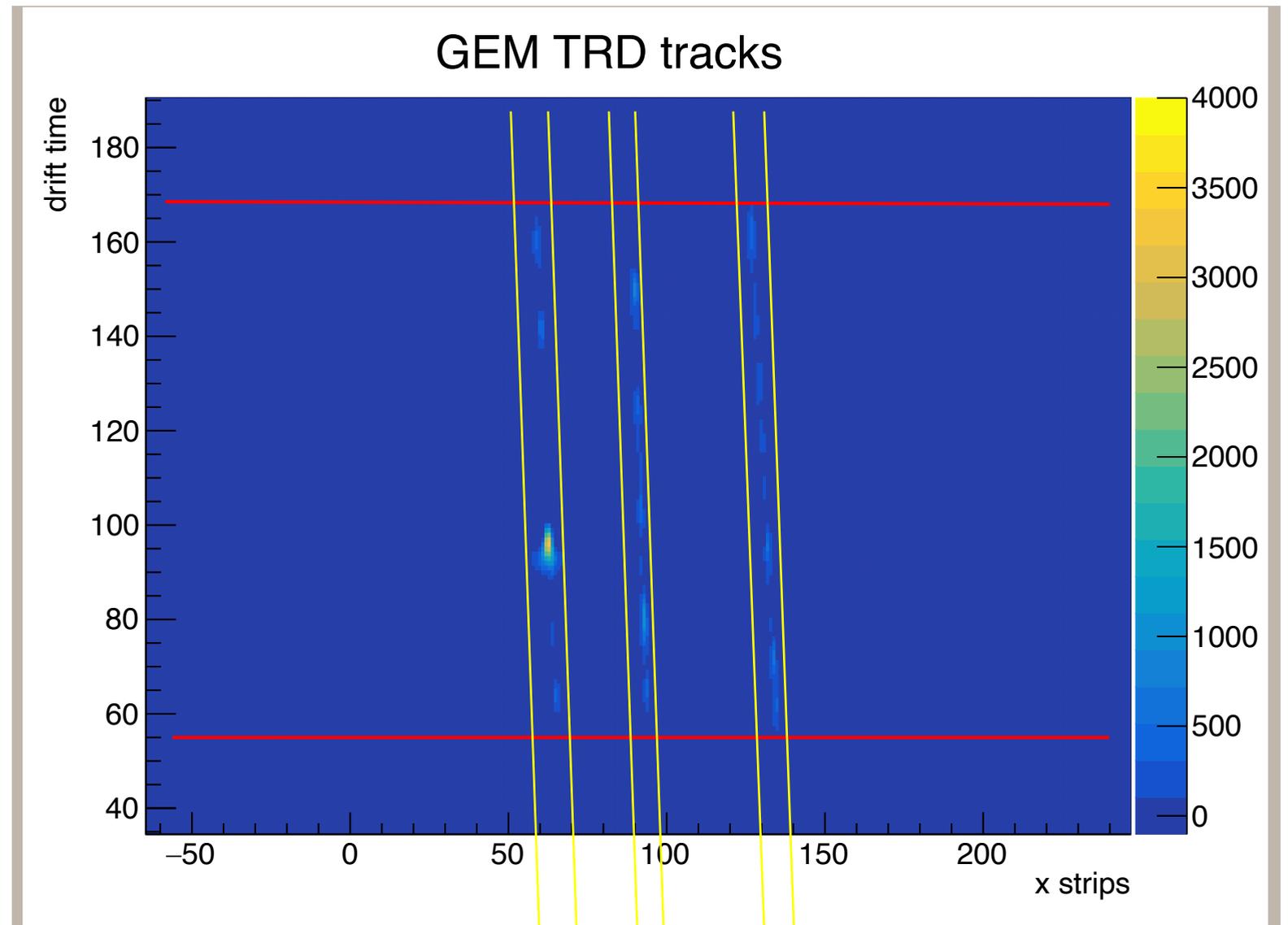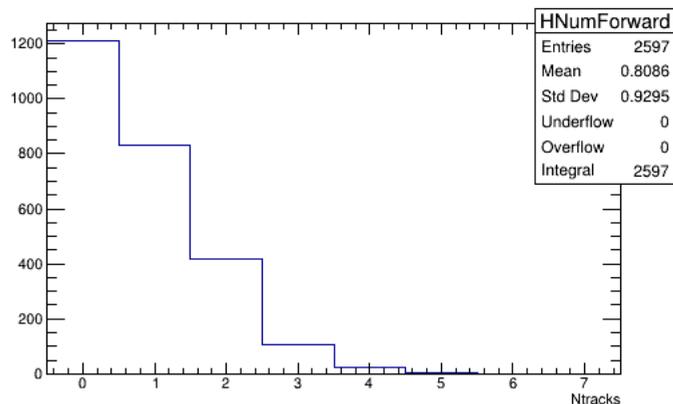
Existing GNN tracking projects

❑ **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

(v_i, e_k)

GNN

(e_k''')
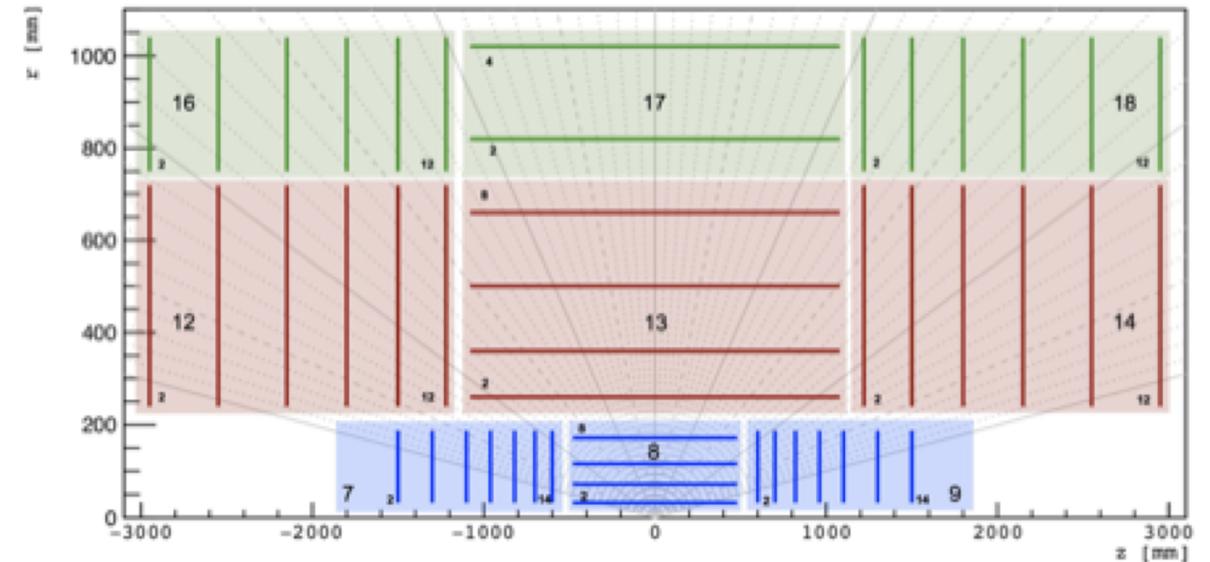
So we decided to start by evaluating an Exa.TrkX solution

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*
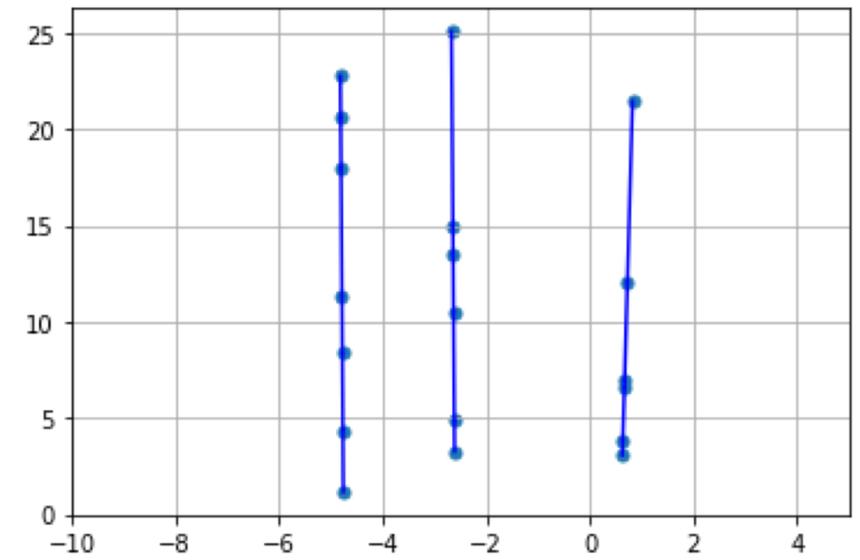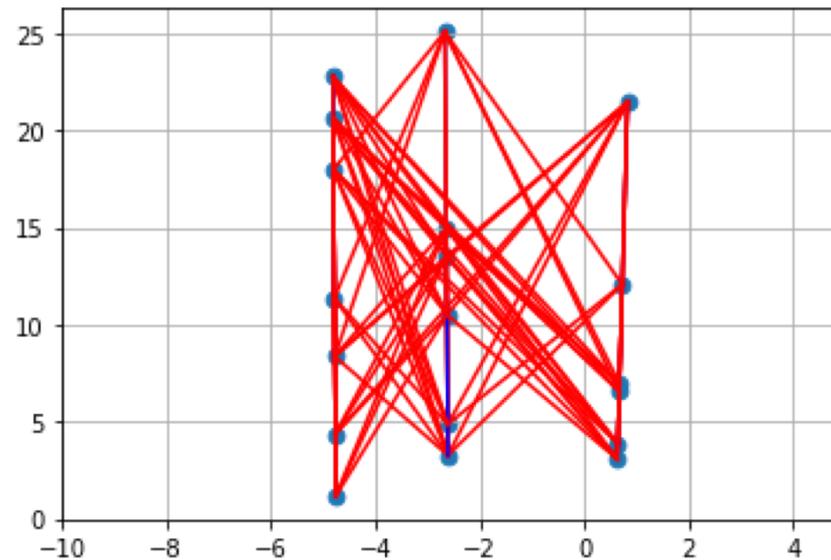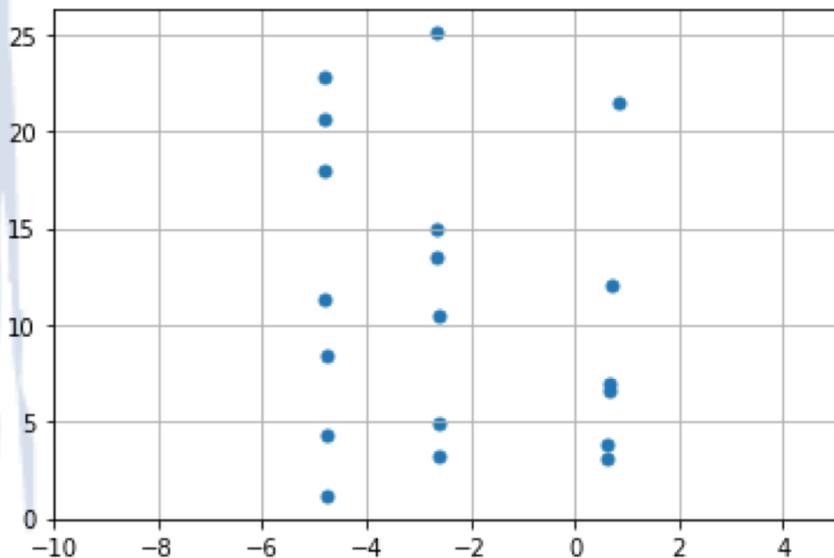
- ❑ *This type of graph neural network is not yet supported in HLS4ML.*
- ❑ *So we did a manual conversion first to C++ and then to Verilog using Vitis_HLS.*
- ❑ *This neural network has not been optimized, so it consumes a lot of resources - 70% of DSPs, (4858 of 6840).*
  - ➢ At the moment it can serve 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 µs (left plot) (thanks to Ben Raydo), providing good purity and efficiency (right plot).*

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.*
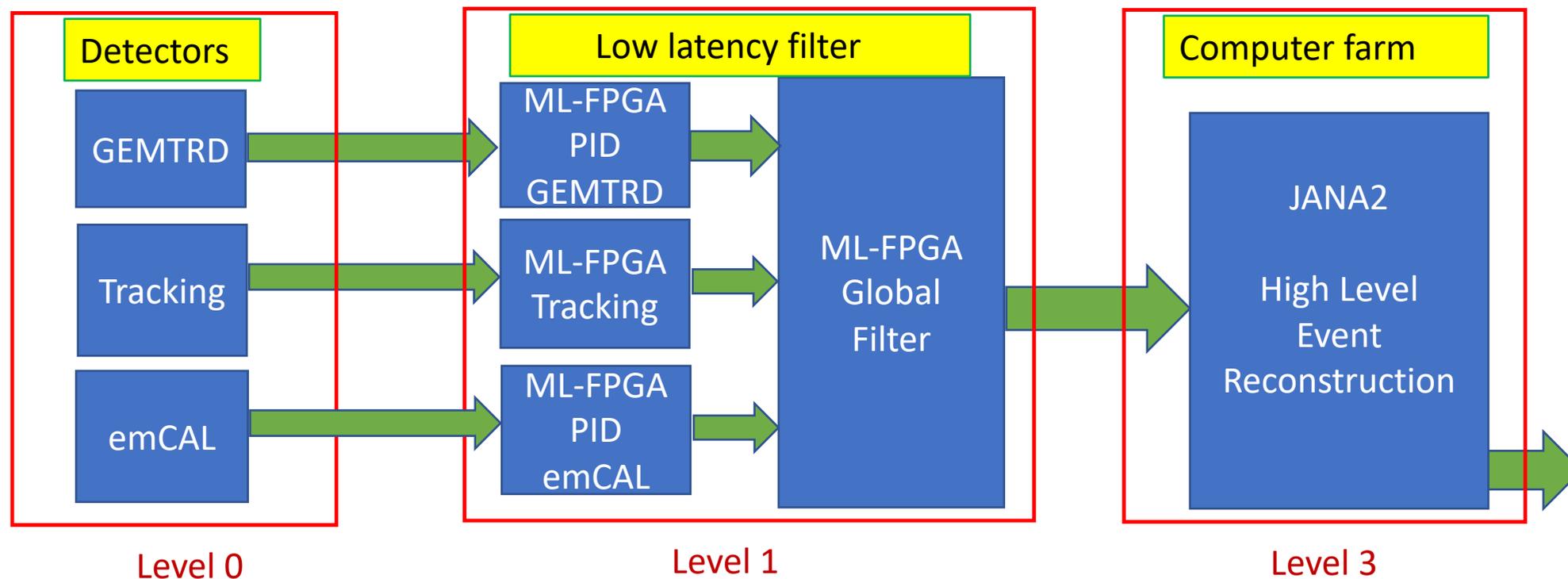
FPGA test bench

- ❏ *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..*

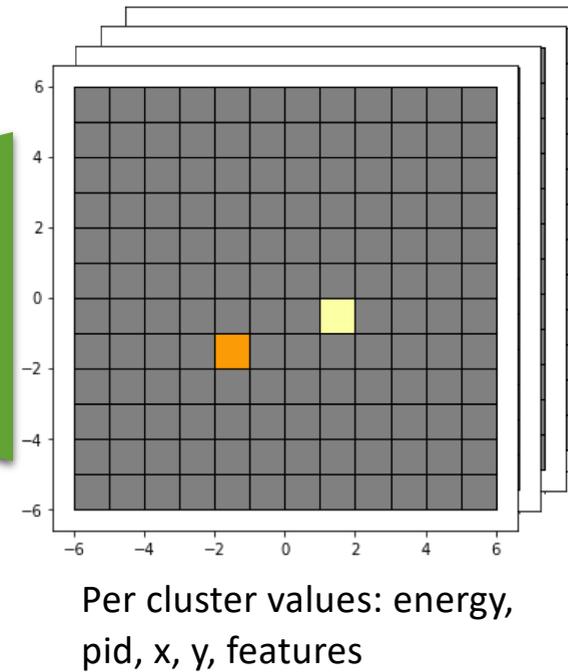GNN Pattern recognition

DNN/LSTM track fit

DNN PID module
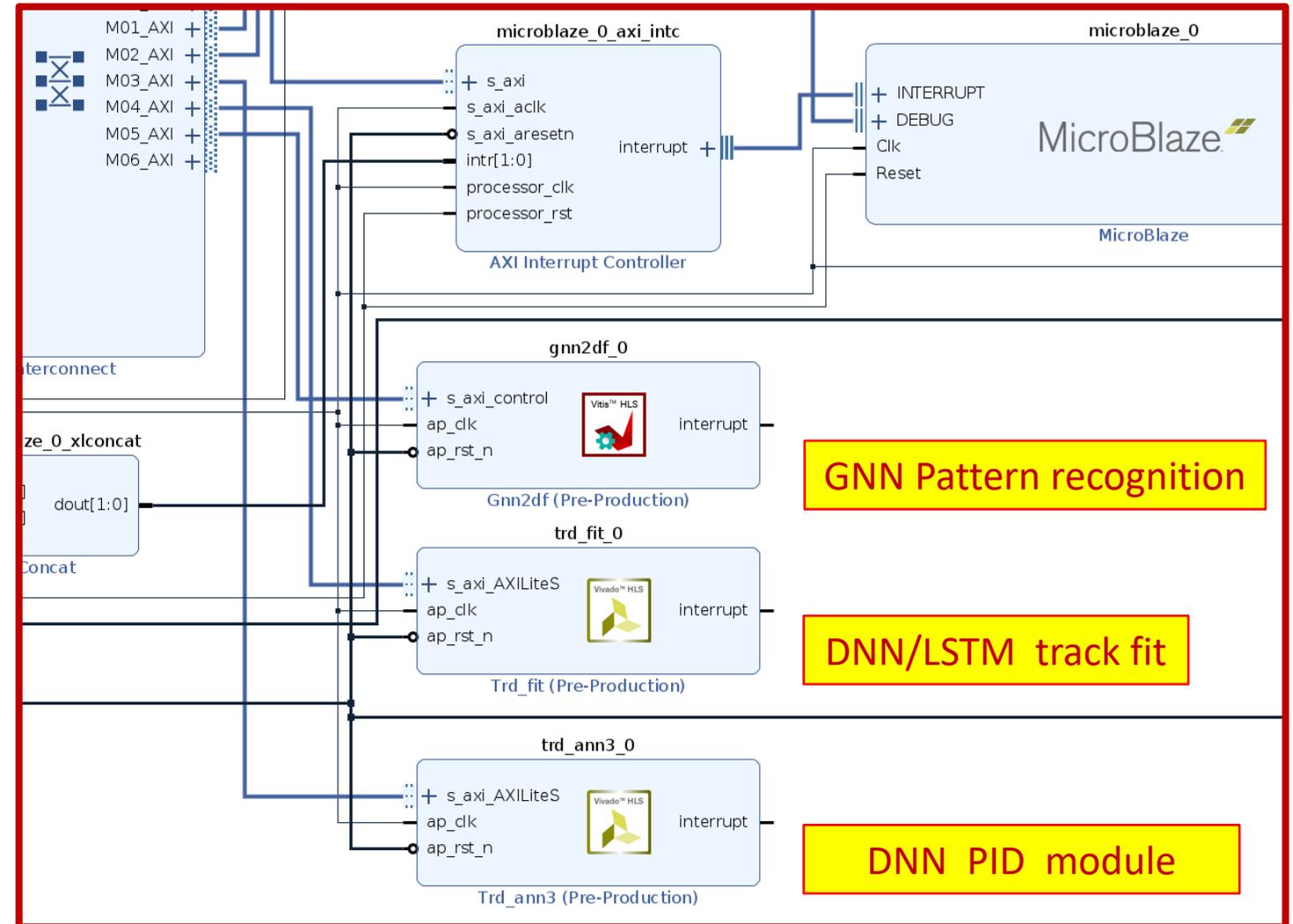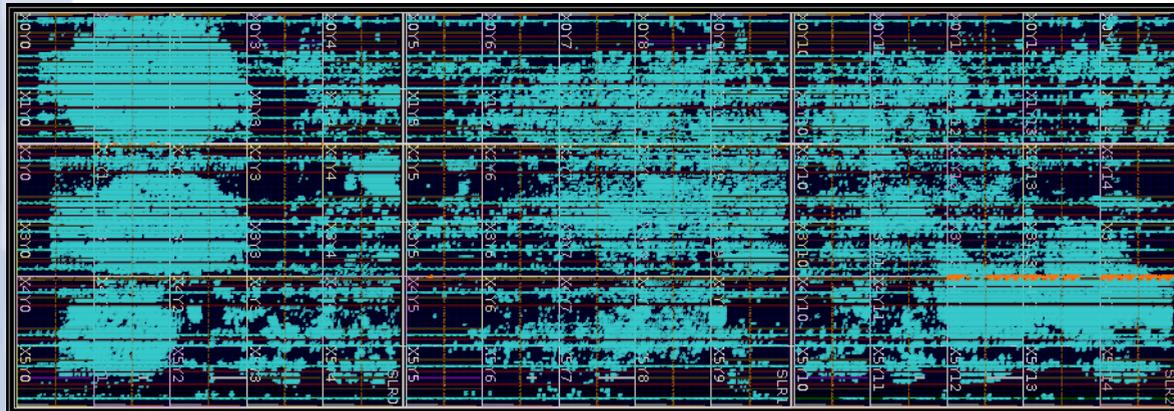
Global PID

❑ *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

Modules deposits

Clusters

Encoder

CR

Decoder

Convolutional variational autoencoder

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

Geant 4 simulation

π^-

e

μ^-

$PbWO_4$ 20 cm

Examples of events with e and π^- showers and μ^- passing through.
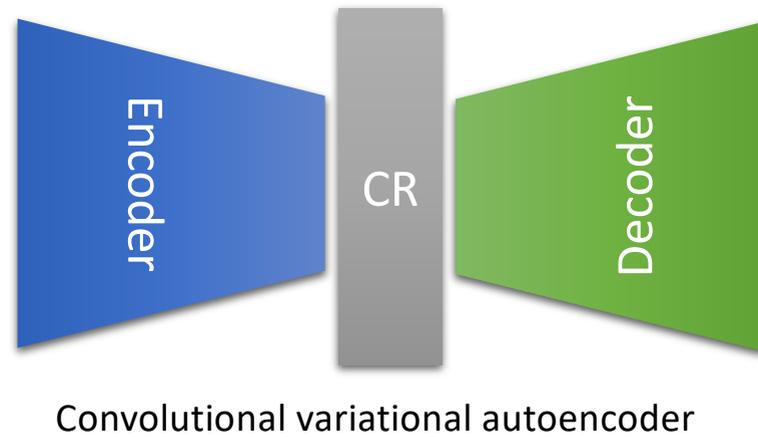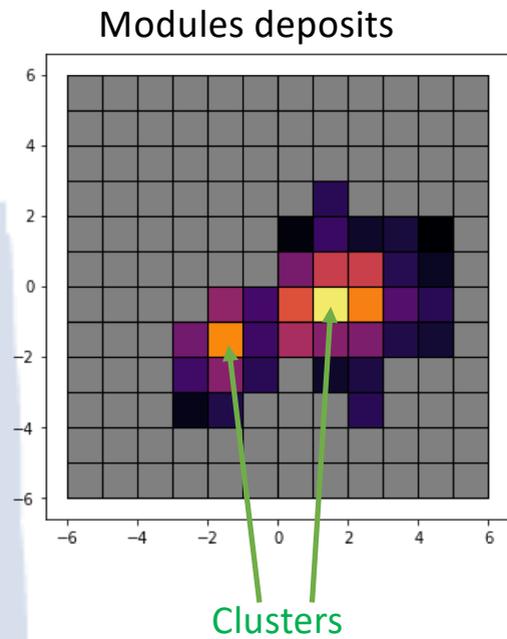
- Convolutional VAE as a backbone
- Modules deposits as inputs
- Per cluster output of multiple values:
- Energy, e/π, coordinates, features

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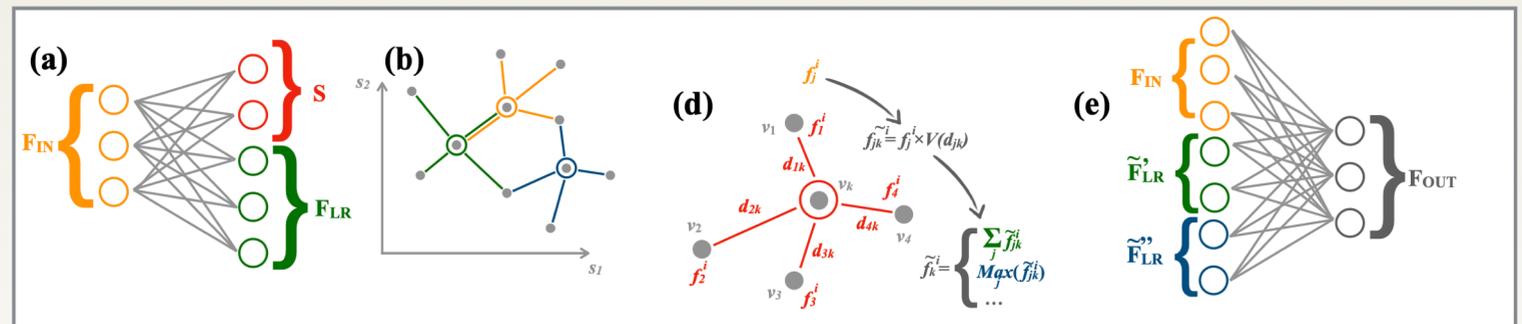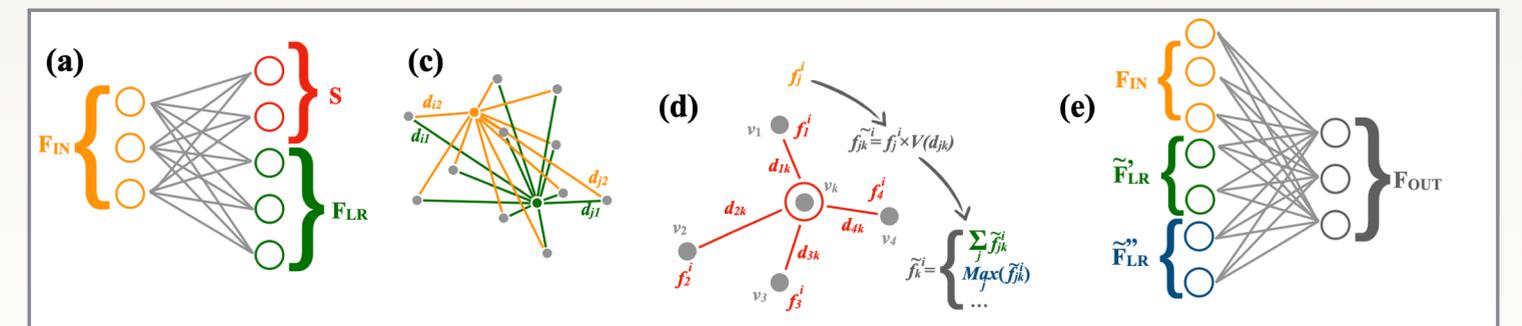
S.R. Qasim, J.K, Y. Iiyama, M Pierini arXiv:1902.07987, EPJC

Developing ethernet interface

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.

Outlook

✦ Machine learning methods are widely used and have proven to be very powerful in particle physics.

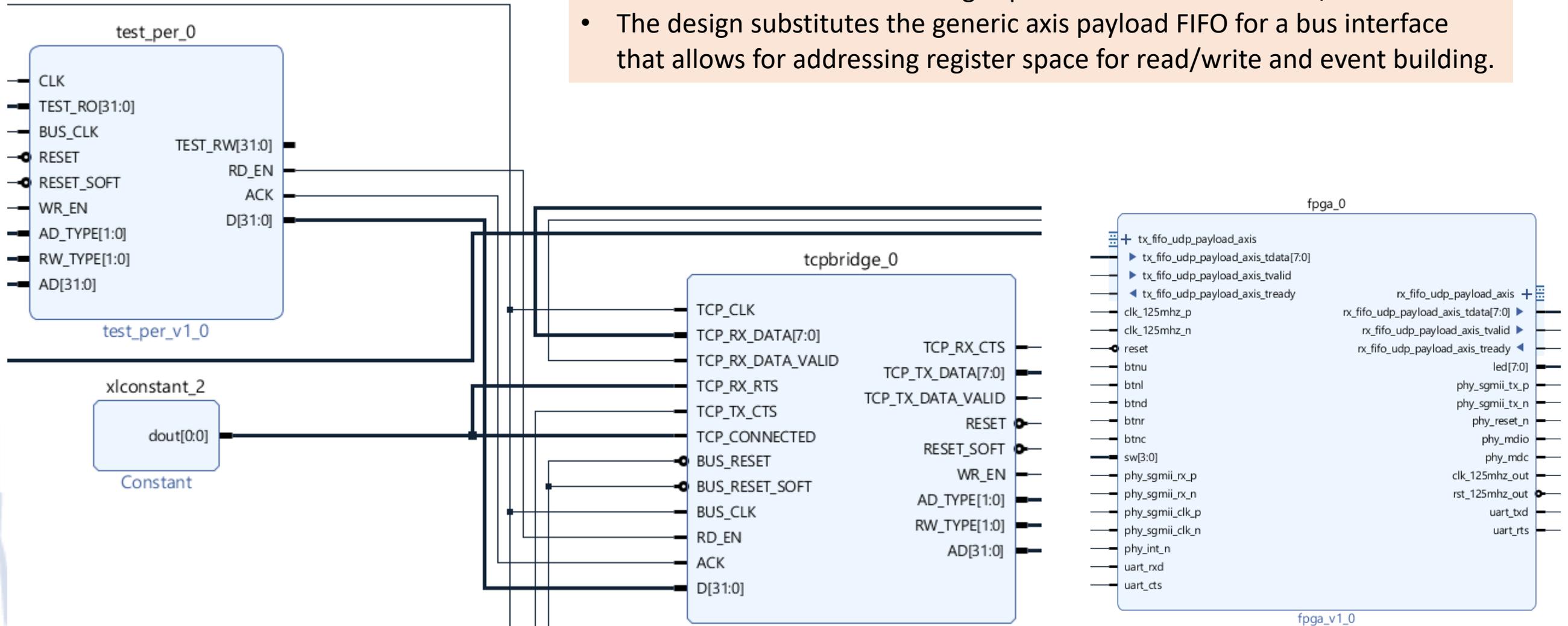✦ Although the methods of machine learning and artificial intelligence are developed by many groups and have a lot in common, nevertheless, the hardware used and performance is different.

✦ 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.*
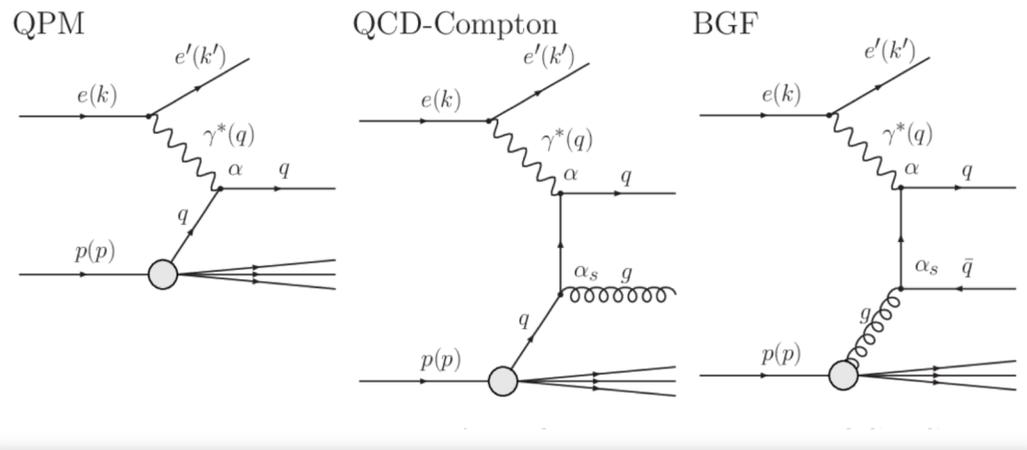

Figure 2.1: Feynman diagrams of the Quark Parton Model, QCD-Compton and Boson Gluon Fusion processes in NC DIS.
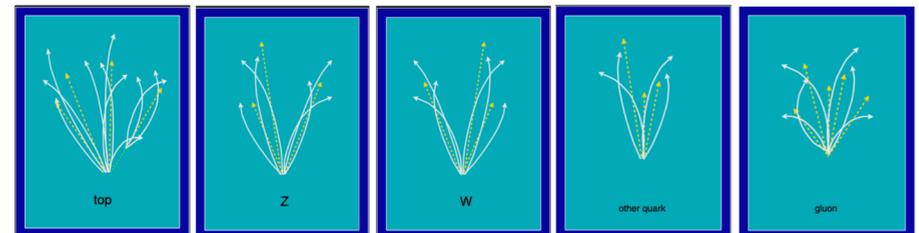
Published in 2007
Measurement of multijet events at low x_{Bj} and low Q^2 with the ZEUS detector at HERA
T. Gosau

Case study: jet tagging

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

| top | Z | W | other quark | gluon |

| 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

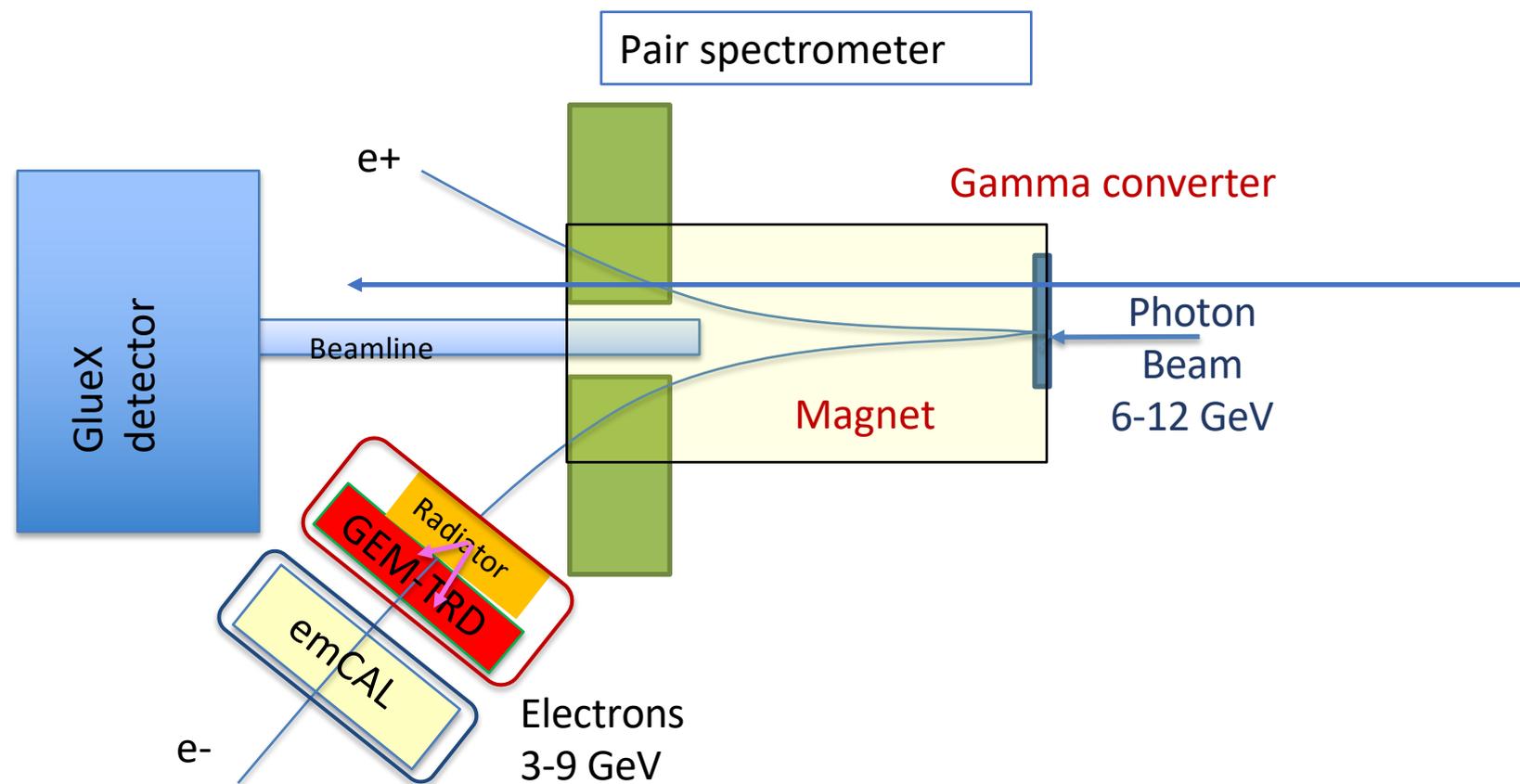**Jet substructure observables used to distinguish signal vs background** [*]

[*] 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

Sergey Furletov   23rd IEEE Real Time Conference
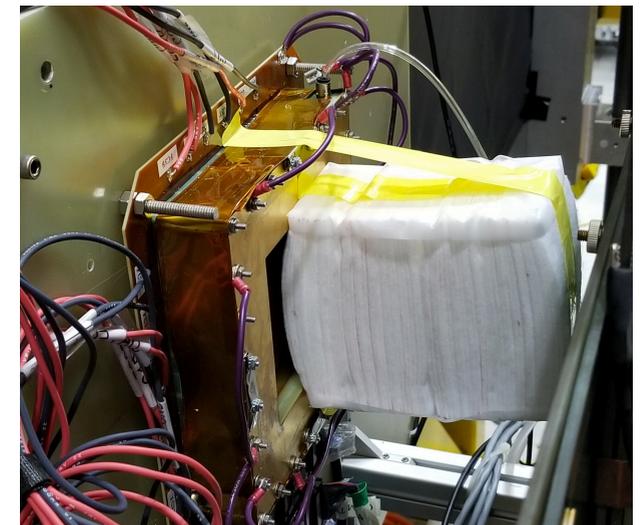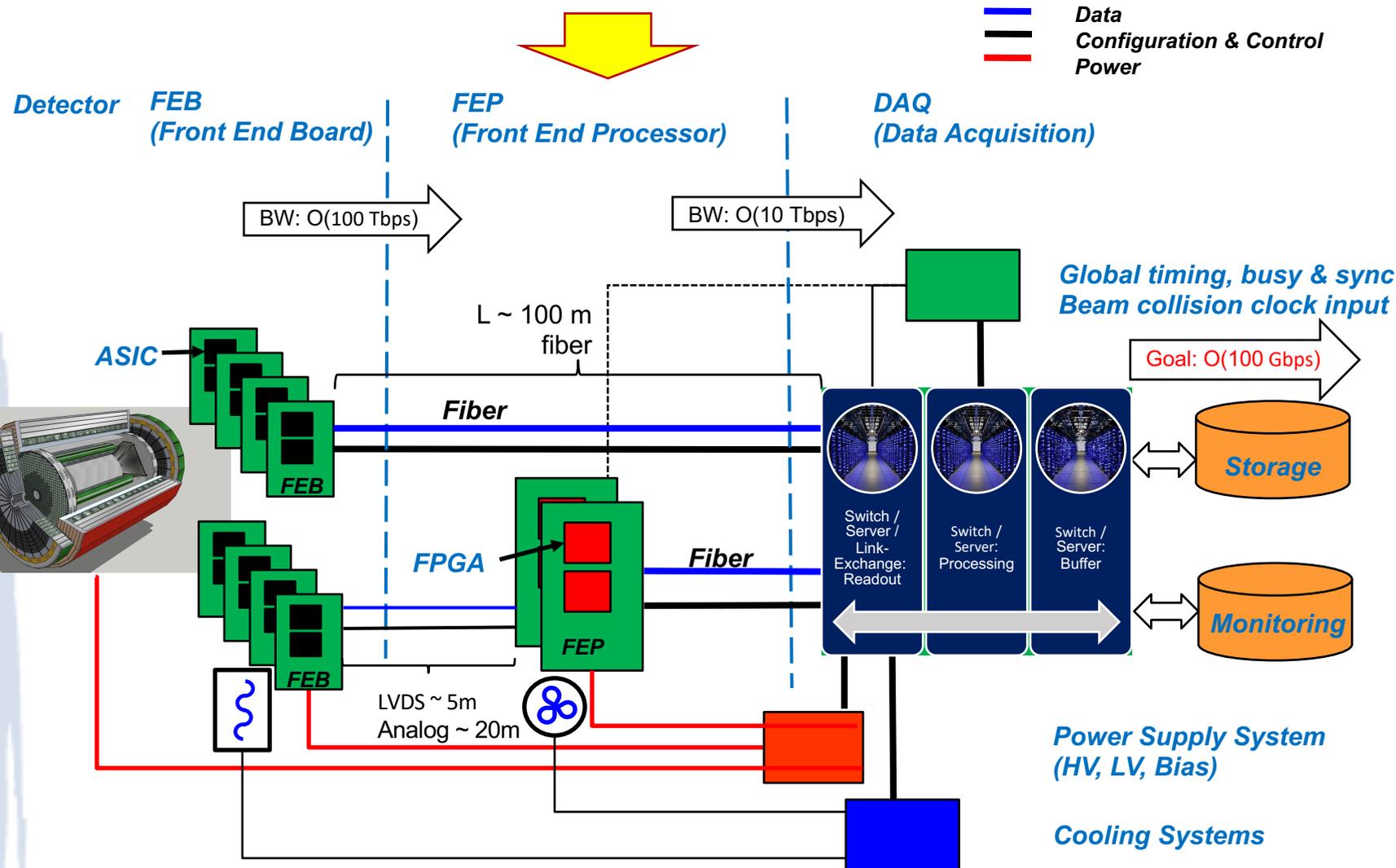
- *Tests were carried out using electrons with an energy of 3-6 GeV, produced in the converter of a pair spectrometer at the upstream of GlueX detector.*

Pair spectrometer

e+

Gamma converter

GlueX detector

Beamline

Photon Beam 6-12 GeV

Magnet

Radiator

GEM-TRD

emCAL

Electrons 3-9 GeV

e-

GEMTRD prototype

EIC readout as motivation

Jefferson Lab — Thomas Jefferson National Accelerator Facility

Legend:
- Data (blue)
- Configuration & Control (black)
- Power (red)

Detector

FEB (Front End Board)

FEP (Front End Processor)

DAQ (Data Acquisition)

BW: O(100 Tbps)

BW: O(10 Tbps)

ASIC

L ~ 100 m fiber

Fiber

FEB

FPGA

FEP

Fiber

LVDS ~ 5m
Analog ~ 20m

Switch / Server / Link-Exchange: Readout

Switch / Server: Processing

Switch / Server: Buffer

Global timing, busy & sync
Beam collision clock input

Goal: O(100 Gbps)

Storage

Monitoring

Power Supply System (HV, LV, Bias)

Cooling Systems

- 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.

ADC based DAQ for PANDA STT

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Level 0 Open VPX Crate
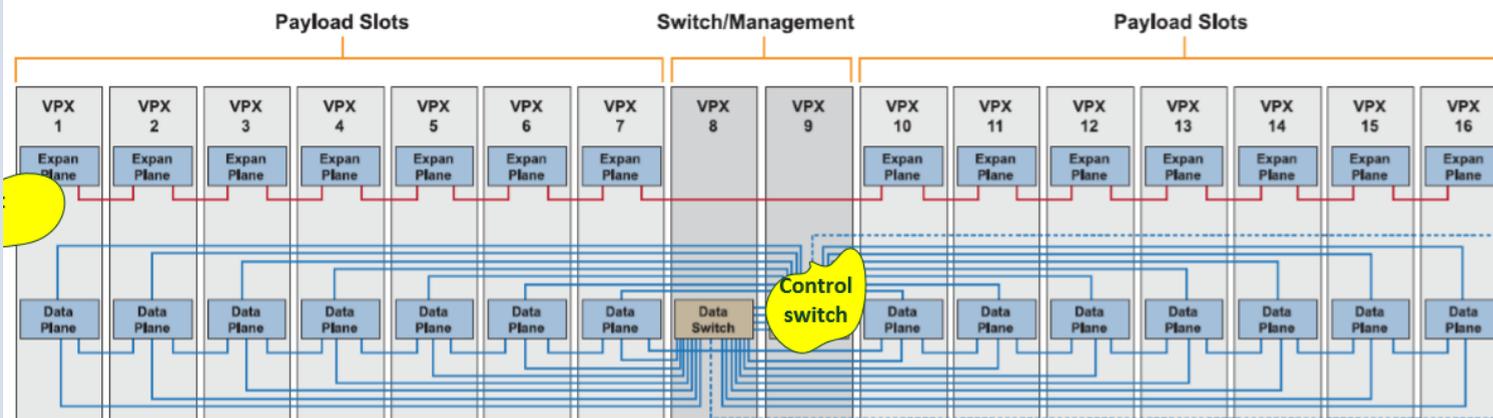
ADC based DAQ for PANDA STT (one of approaches):
- 160 channels (shaping, sampling and processing) per payload slot, 14 payload slots+2 controllers;
- **totally 2200 channels per crate**;
- time sorted output data stream (arrival time, energy,...)
- noise rejection, pile up resolution, base line correction, ..
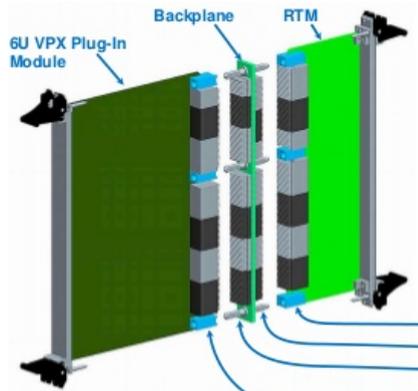
- 40 4-channel ADCs (configurable up to 1 GSPS);
- Single Virtex7 FPGA

- 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