Real time and AI integrated data processing system

Zhao-Zhi Liu, Yuxin Liu(KEK), Shuangshuang Zhang, Xiaoqian Hu, Yunqing Mao
Xingtao Huang, Qi-Dong Zhou

2025.11.09

The 2025 International Workshop on the High Energy Circular Electron Position Collider

The Belle II experiment

Belle II detector including:

Tracking: Vertex detectors and CDC.

Particle identification: TOP and ARICH

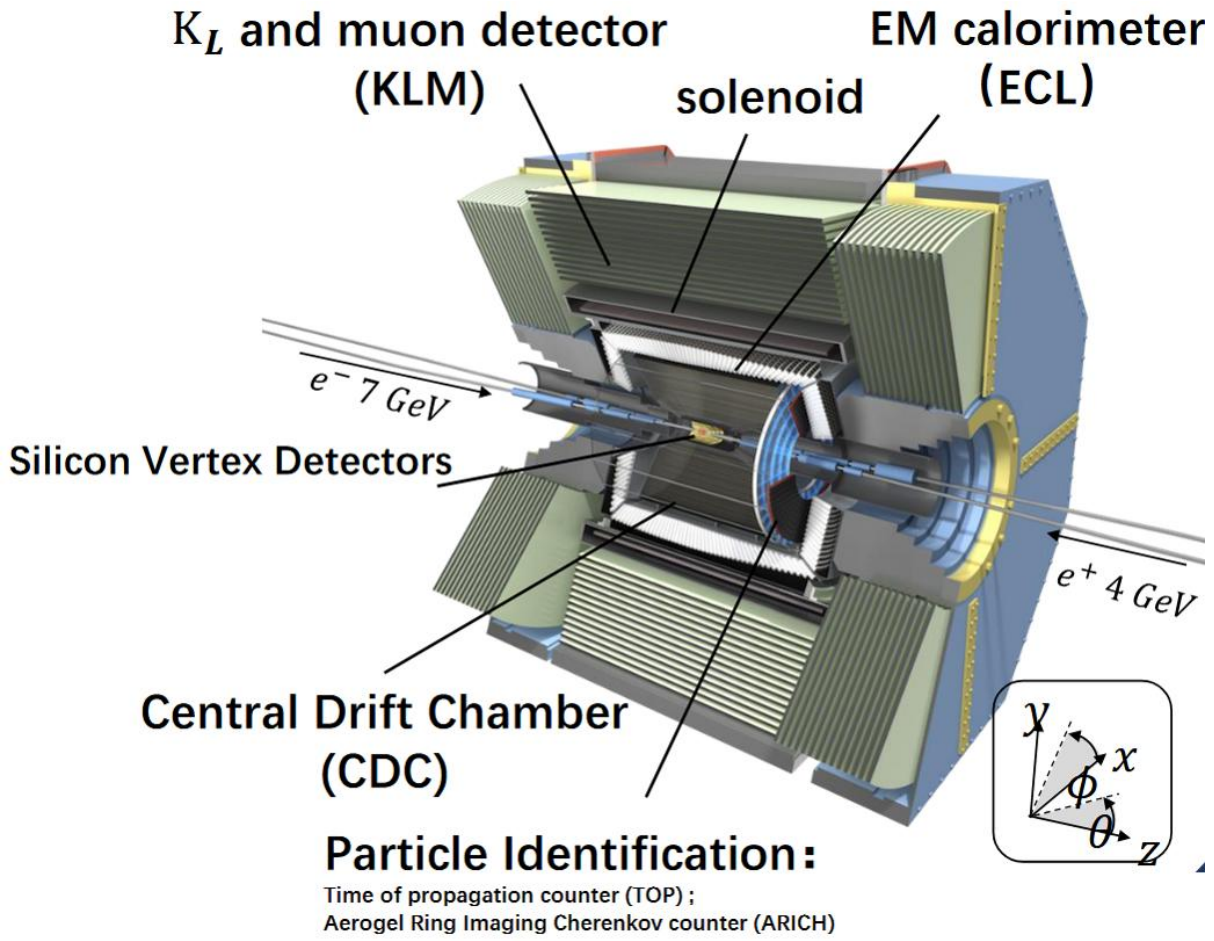
Calorimeter: ECL.

KL and muon detector.

First level (L1) trigger, High level trigger (HLT) and DAQ.

Target luminosity: $60 \times 10^{34}\ cm^{-2}s^{-1}$

Achieved luminosity: $5.1 \times 10^{34}\ cm^{-2}s^{-1}$



Machine Learning for Real-Time Data Processing

The background condition getting worse when the luminosity increased
Real-time data reduction technique is essential

Two ML implementations:

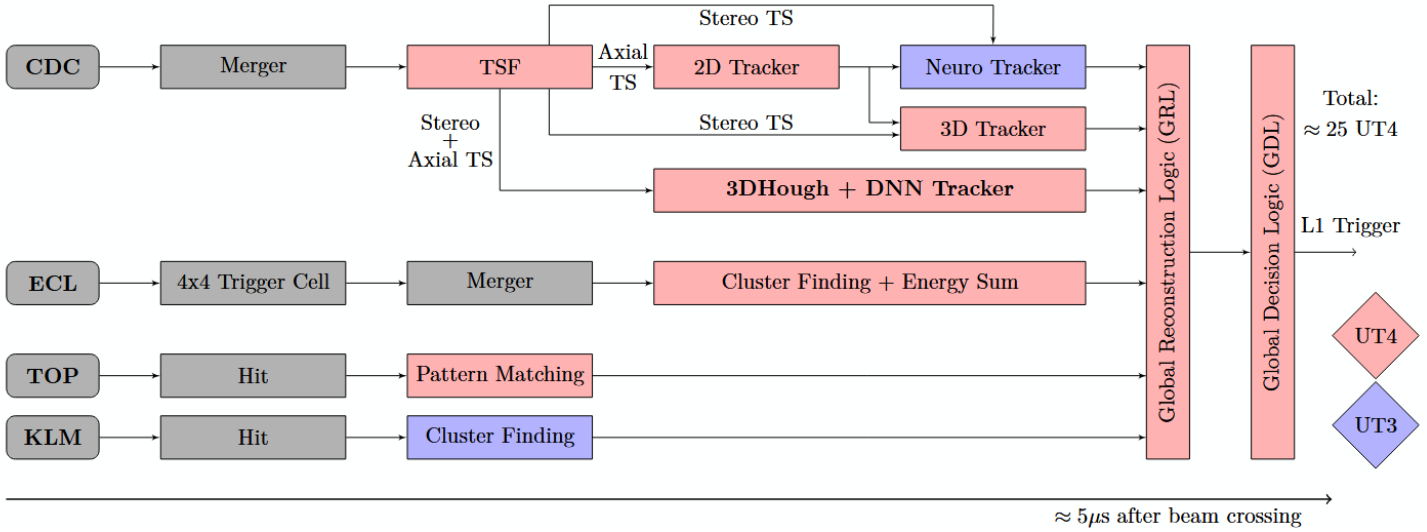
3D tracker with DNN-based Z trigger

L1 trigger

Maximum average trigger rate of 30 kHz,
Fixed latency of about 4.3 us

DNN-based Z trigger

Build **3D track** in CDC with **2D track** and **stereo hit**
Less 600ns latency



L1 trigger

Denoising GNN in High Level Trigger:

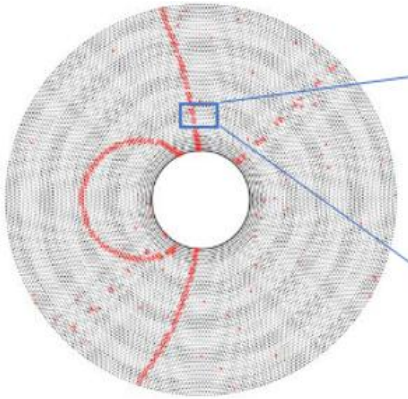
High Level Trigger

Designed to further suppress the background

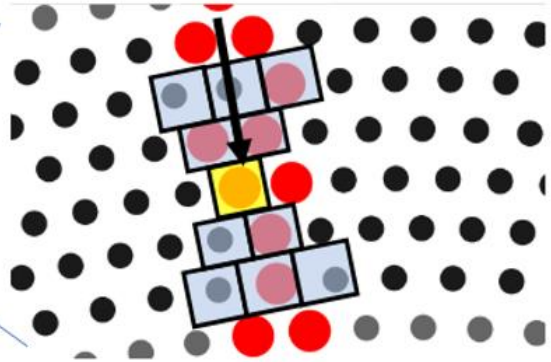
Denoising GNN

Online algorithm to filter out background CDC hits for track reconstruction

DNN for Z Trigger of L1



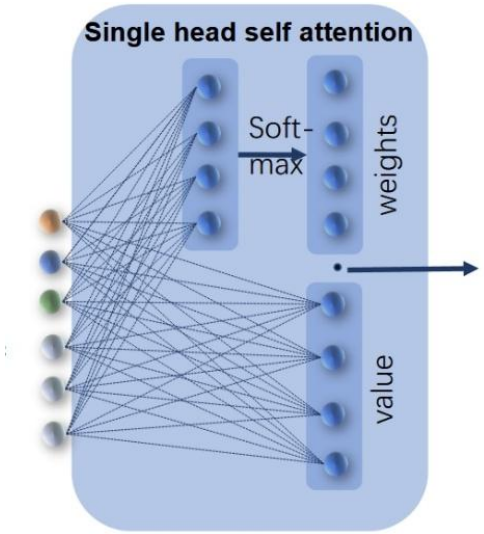
CDC profile

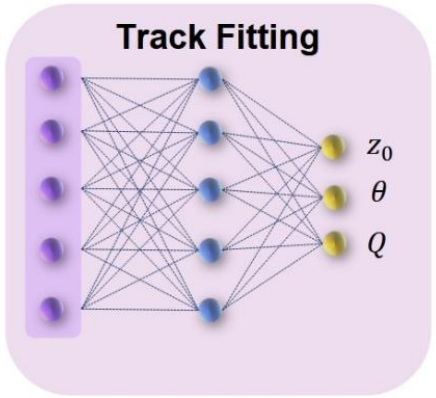


Track Segment(TS)

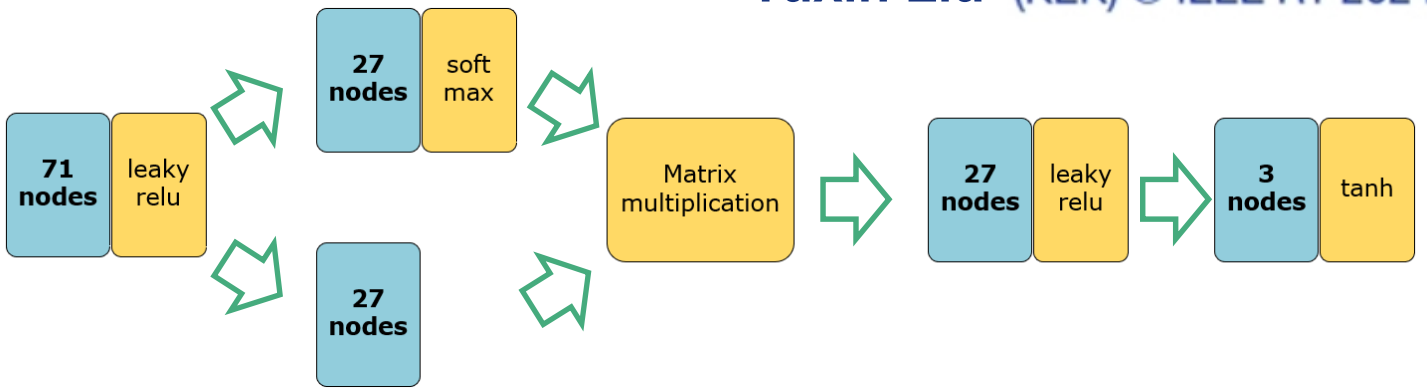
1. One TS uses 5 layers of wire.
2. Each SL(super layer) has 6 layers
3. Variables of each TS:
   track crossing angle α,
   drift time t
   relative angle φ

One TS in one SL





Yuxin Liu (KEK) @ IEEE RT 2024

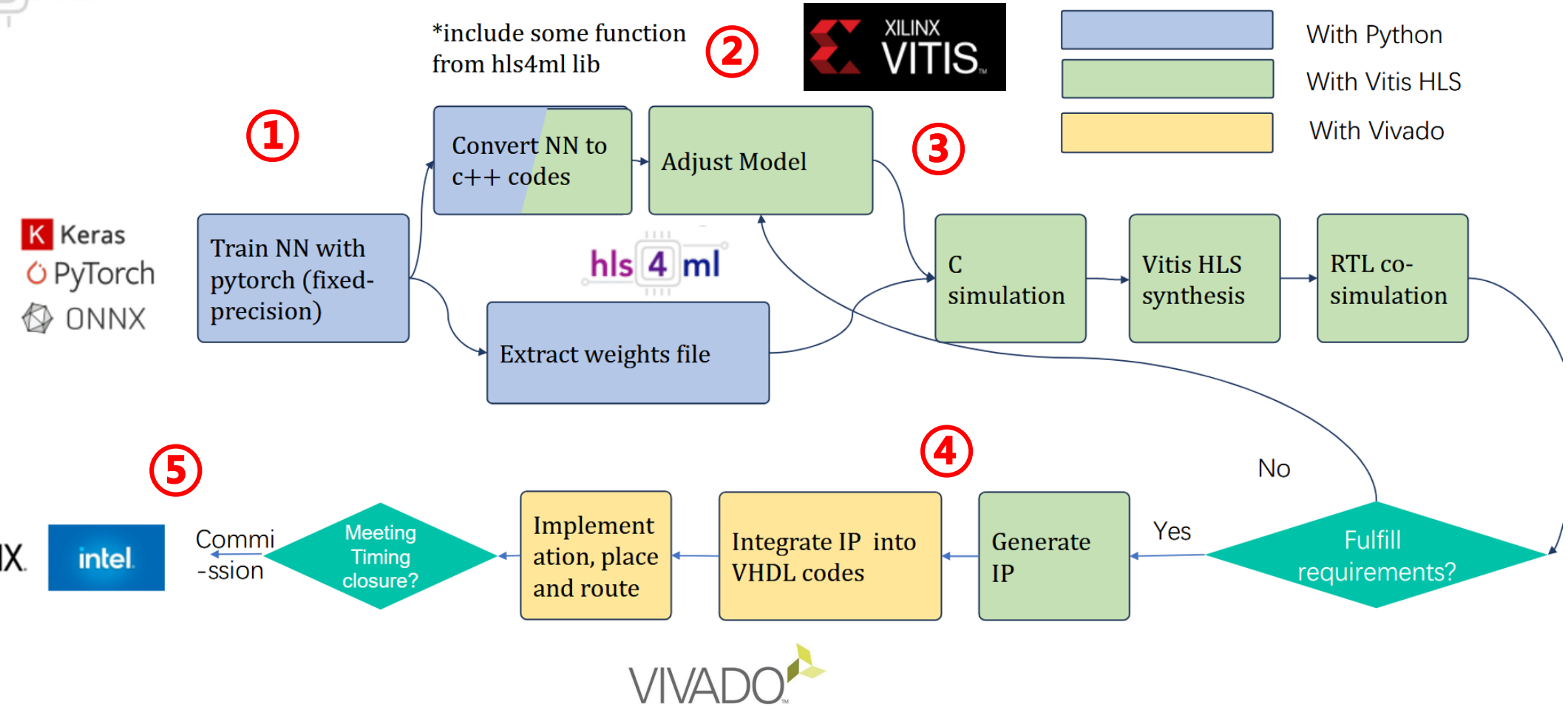


Input drift time t, wire relative location φ, crossing angle α
Output track vertex Z_0, track ɵ, signal/background classifier output Q

DNN implementation by HLS

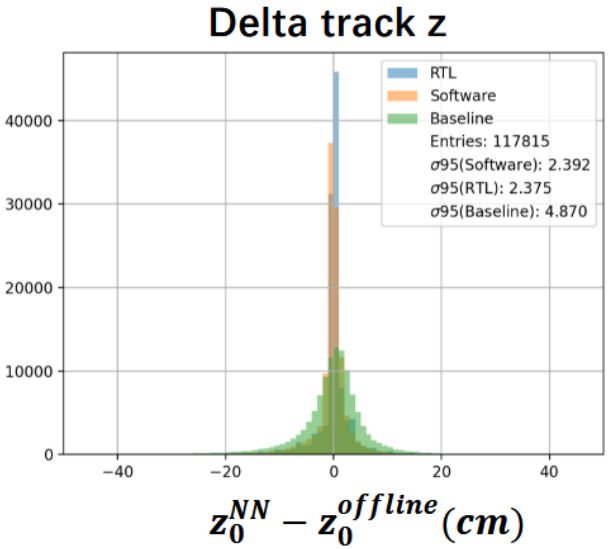
hls4ml

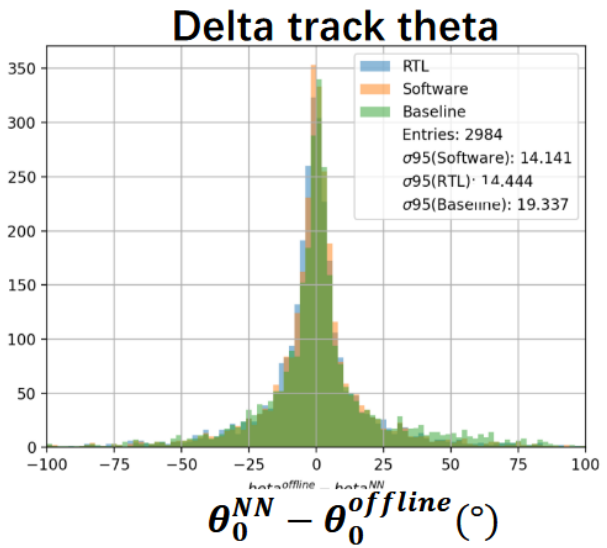
open-source project serving for full procedure from python model → C/C++ code project



①train model in python → ②use hls4ml to export model in C/C++ → ③ import to Vitis HLS and compile and generate IP core → ④import the IP core in Vivado → ⑤download to hardware

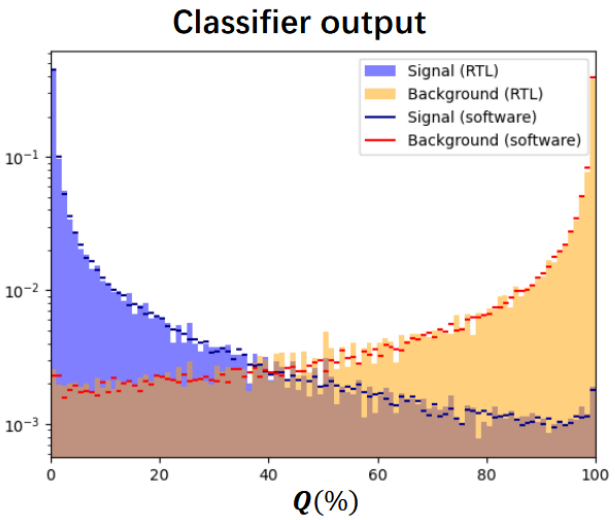
Performance of DNN on FPGA

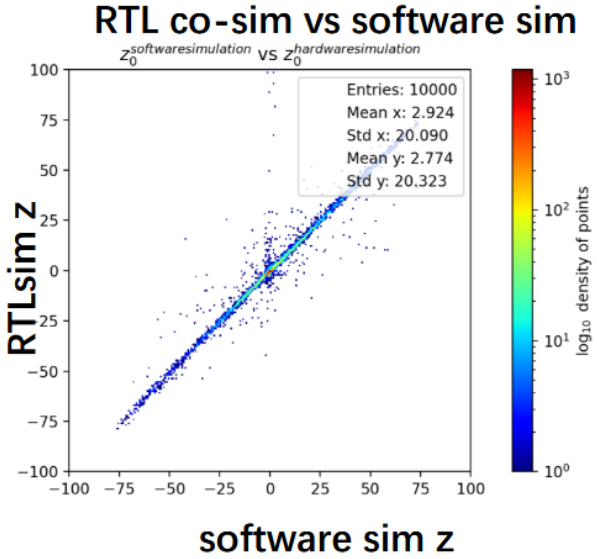




Y. X. Liu(KEK)
S. S. Zhang(SDU)

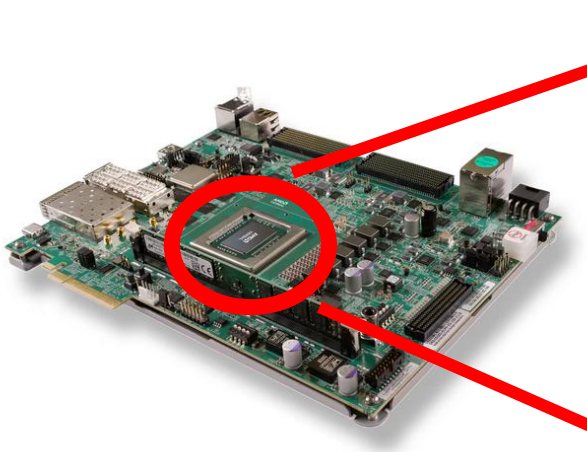
Latency: 76 clock = **551.2ns**(required < 600ns); σ^{z0}=2.7cm; σ^{θ}=14°



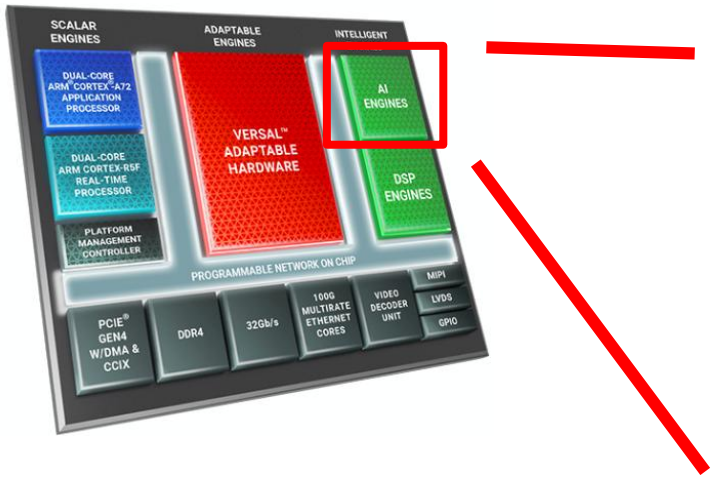


Signal efficiency ~95%, Background reject rate ~85%

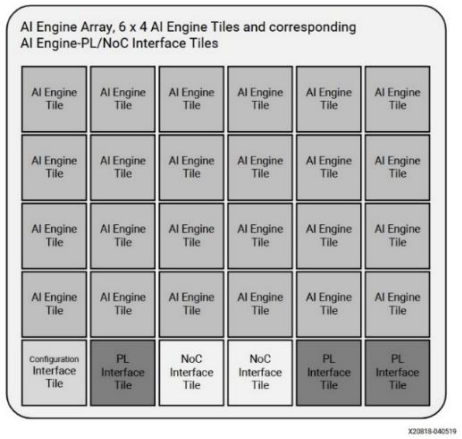
AMD Versal ACAP



Versal VCK190



System On Chip(SOC)



AI engine array

SOC:

Programmable Logic(PL):

FPGA, programmable digital circuit

AI Engine(AIE): (highly optimized for compute-intensive applications)

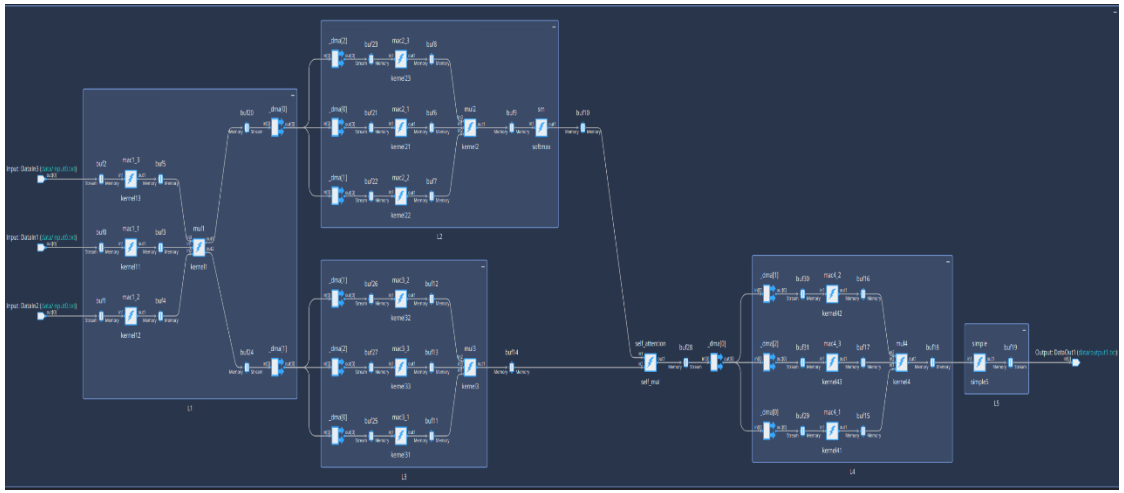
Very-Long Instruction Word(**VLIW**) processor with Single Instruction Multiple Data(**SIMD**) vector unit

Process System(PS): Two Arm cortex processor

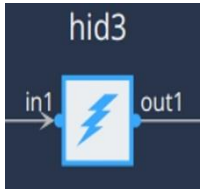
Network on chip(NOC): Designed to handle data transfer with high speed and reliability

Others: DDR; PCIE; Ethernet core

DNN deployment on AI engine

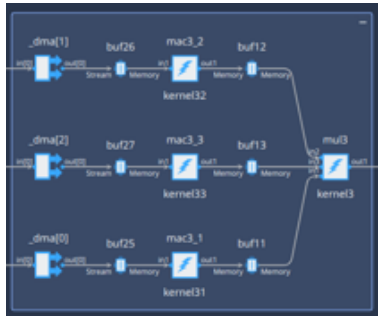


DNN algorithm structure in AI engine



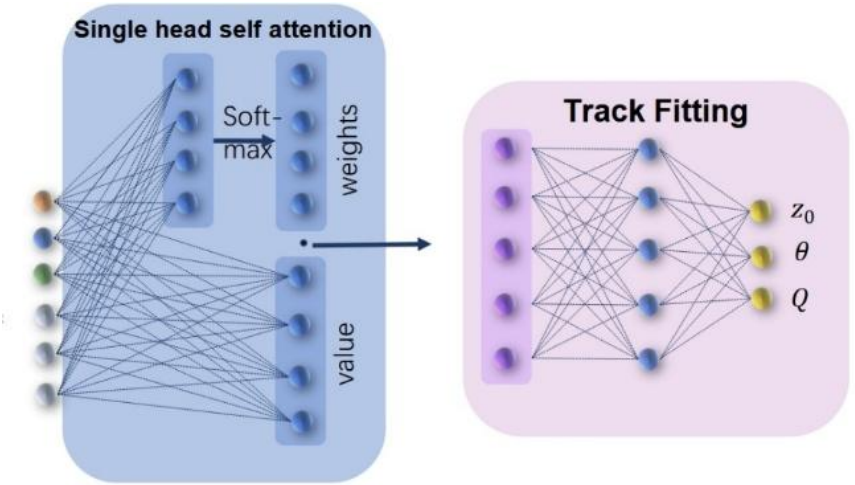
Kernel

Highly optimized computational function that runs on AI Engine hardware

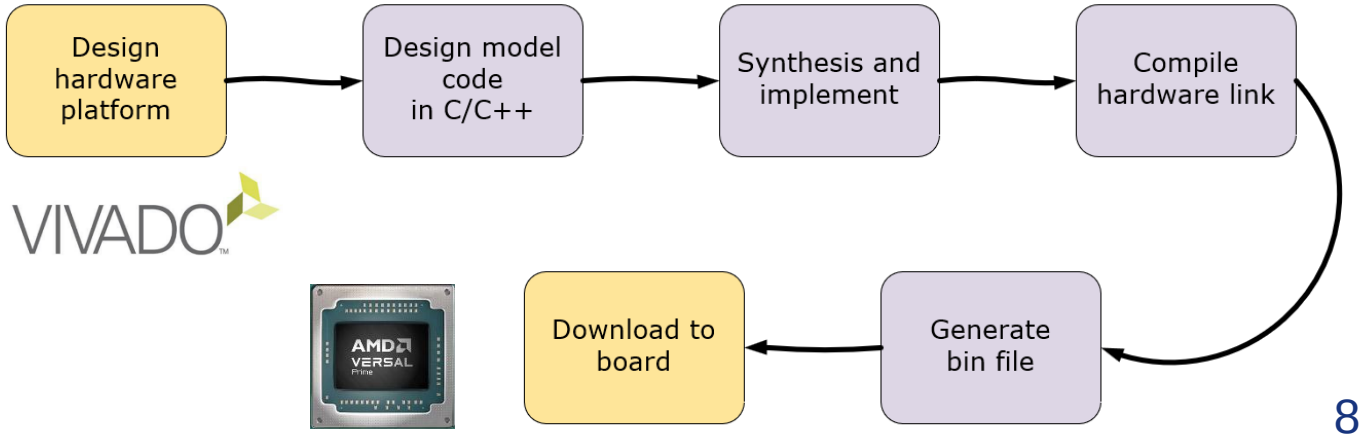


Graph

Connect multiple AI Engine Kernels to build a complete high-performance application

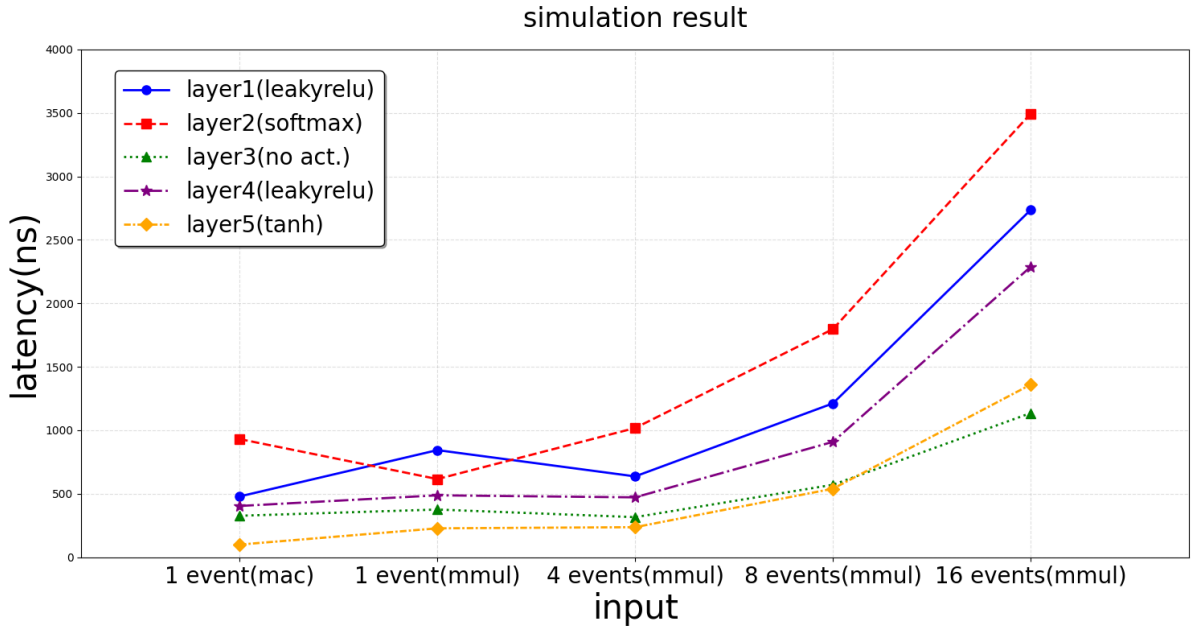


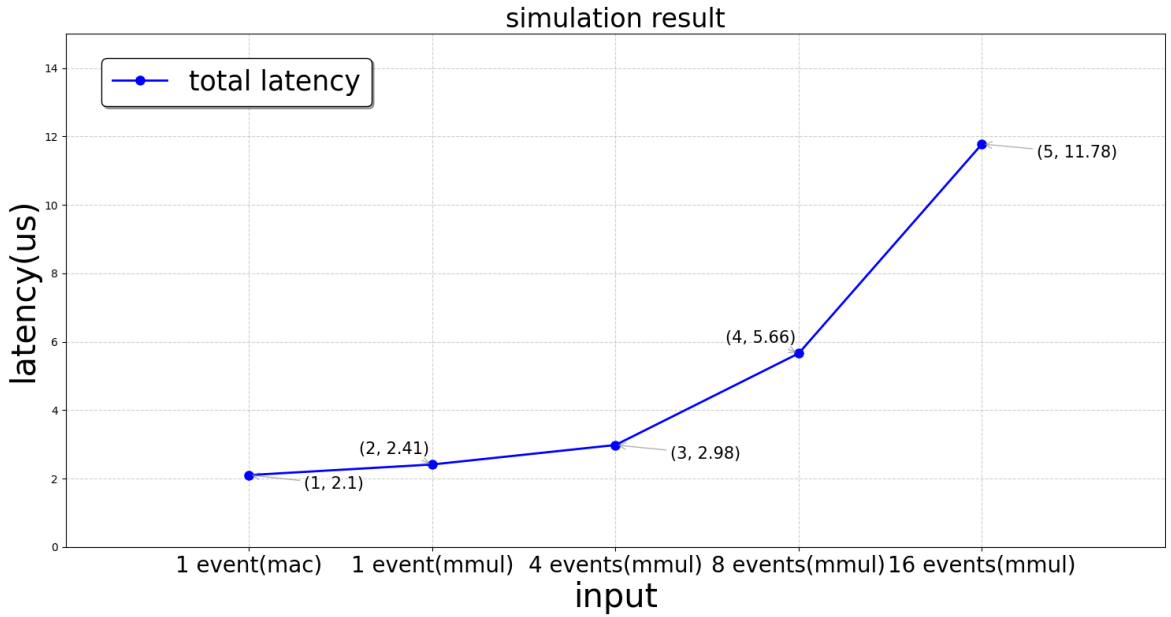
XILINX VITIS



Simulation result of DNN deployed on Versal ACAP

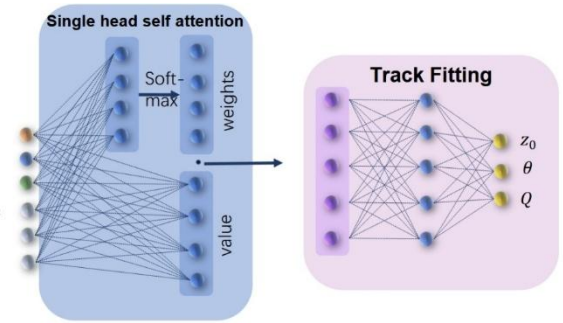
In AIE API, ① **multiply-accumulate(MAC)** ② **matrix-multiplication(MMUL)** can be used. MAC is simpler and MMUL is complex





Best latency is about 2.1us

1. All data in AI engine is float32
2. Softmax: 1event(mac) use approximations; others use LUT
3. If vector is float32 and width 8, one AIE can do 8 times mac in one sys clk at most
4. MMUL is more potential for compute-dense situation compared to MAC.

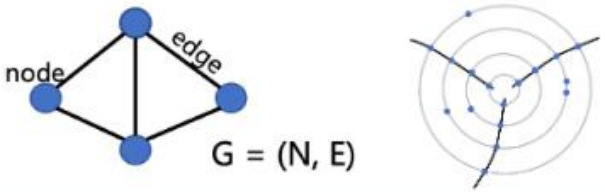


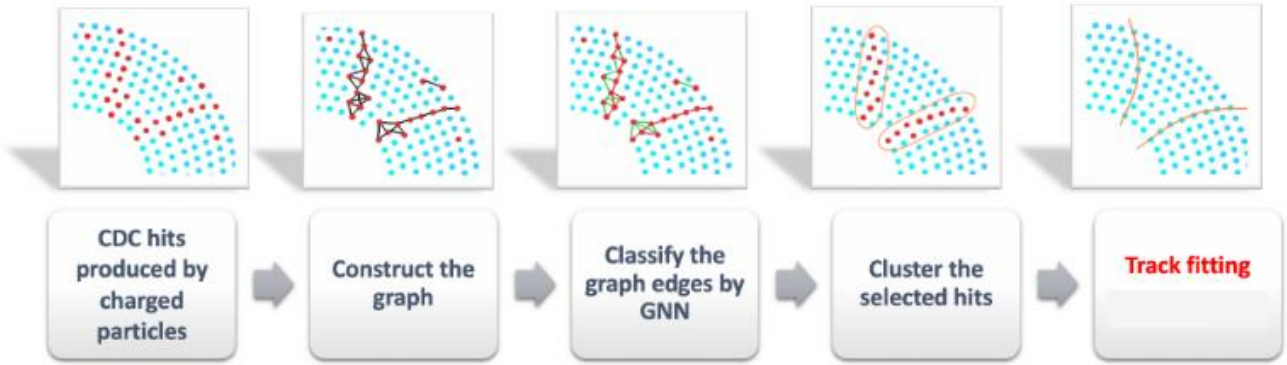
GNN for CDC background filter

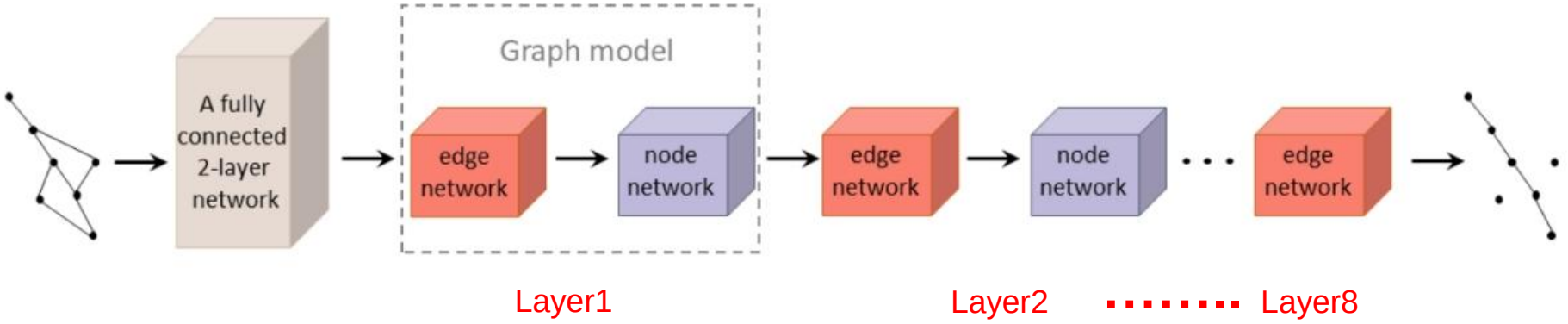
- Developed a GNN algorithm (based on BESIII's algorithm) for Belle II CDC hits clean up
  - Inputs: TDC, position coordinates r, ϕ

X. Q. Jia, X. Q. Hu et al (SDU)

Noise filtering → Clustering → Fitting



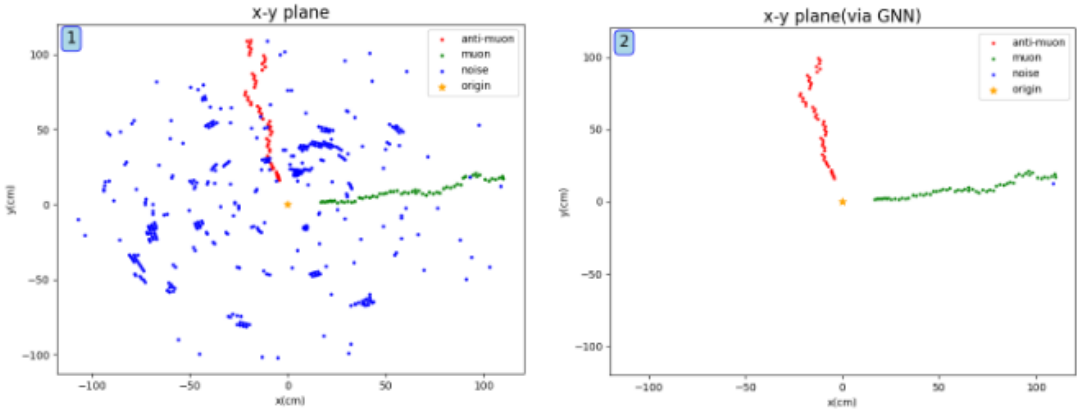




Node:
Wires of CDC
Edge:
Connect nodes
Edge classifier:
Connects nodes → true
Connects background → false

Performance

GNN X. Q. Hu et al (SDU)

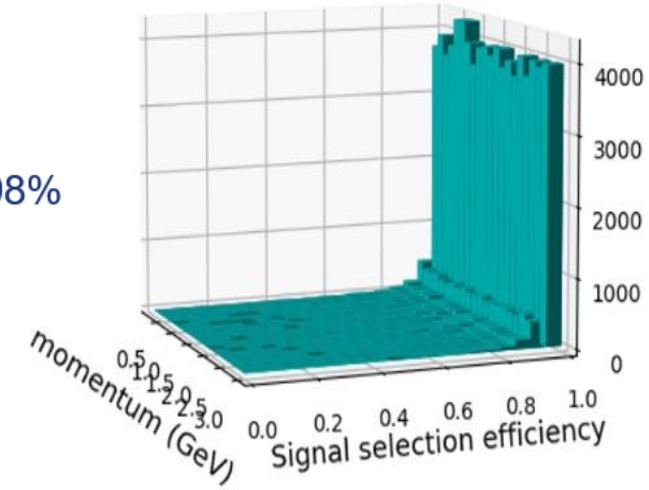


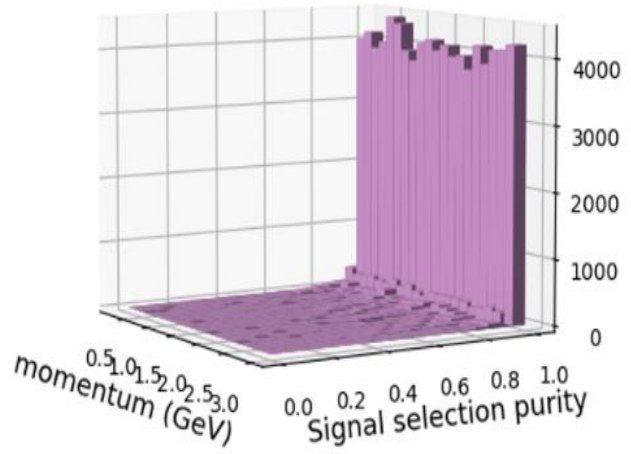
Before After

Simulation data from belle II particle gun

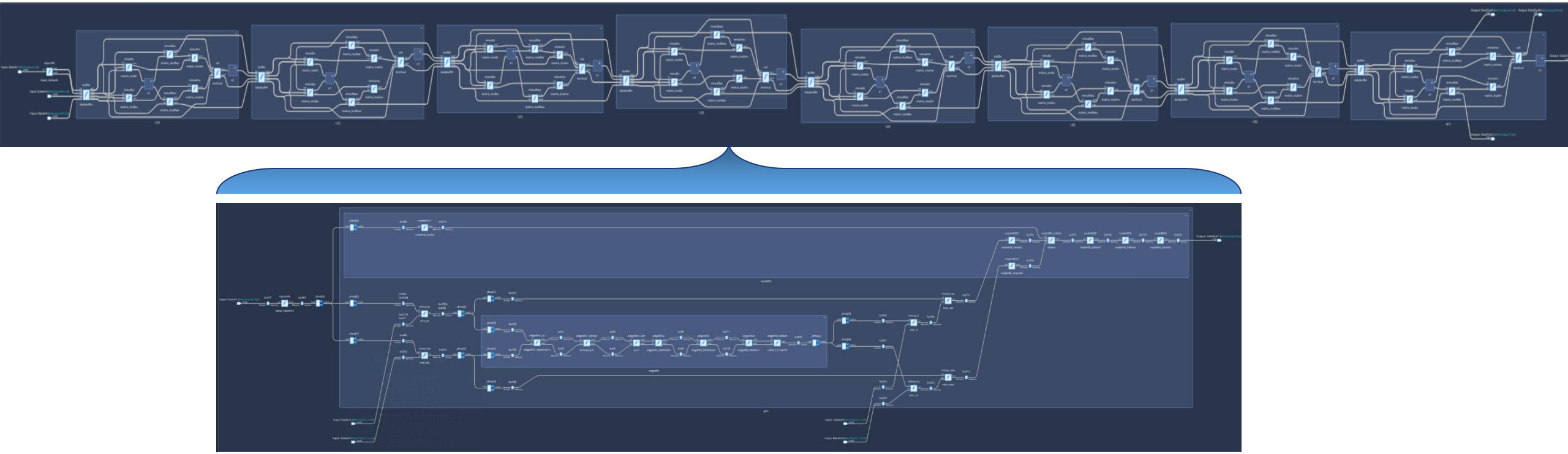
Efficiency $\frac{N_{signal}^{predicted}}{N_{signal}^{real}}$ = ~ 98%

Purity $\frac{N_{signal}^{predicted}}{N_{all}^{predicted}}$ = ~ 98%



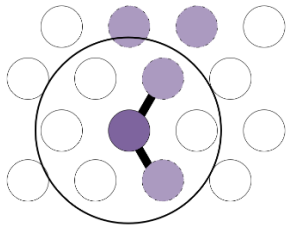


8 layer denoising GNN structure



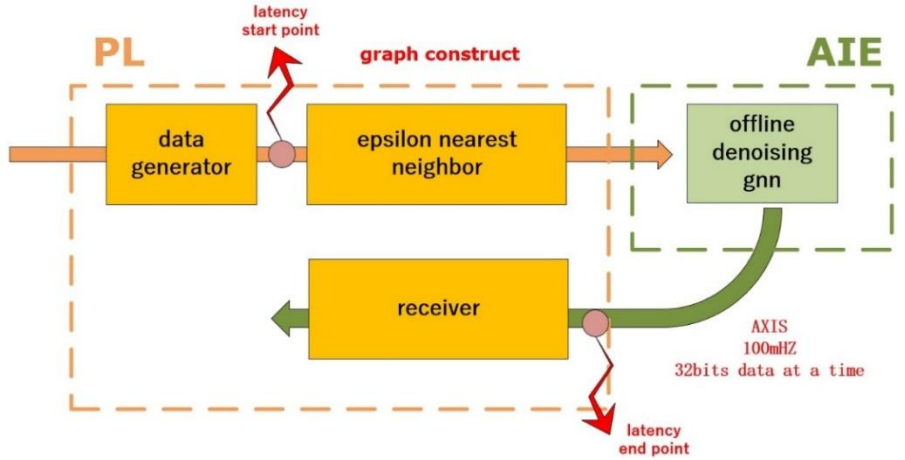
1. Offline GNN algorithm constructed by 8 layers of graph model
2. For the demo propose, one layer of graph model implemented on AIE
3. Futher model optimization needed for training a online GNN algorithm

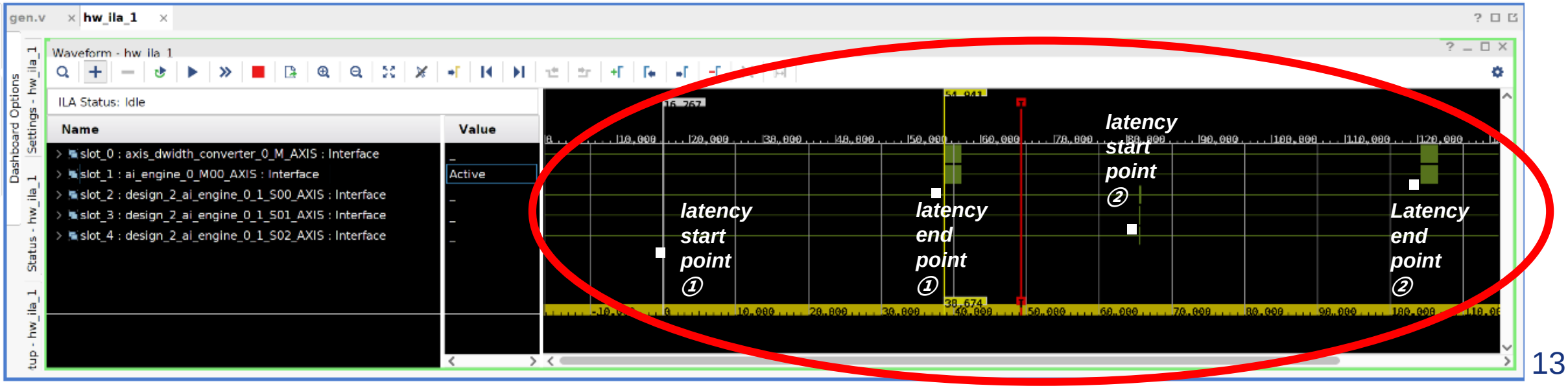
Test bench setup for GNN deployment



ε nearest neighbor

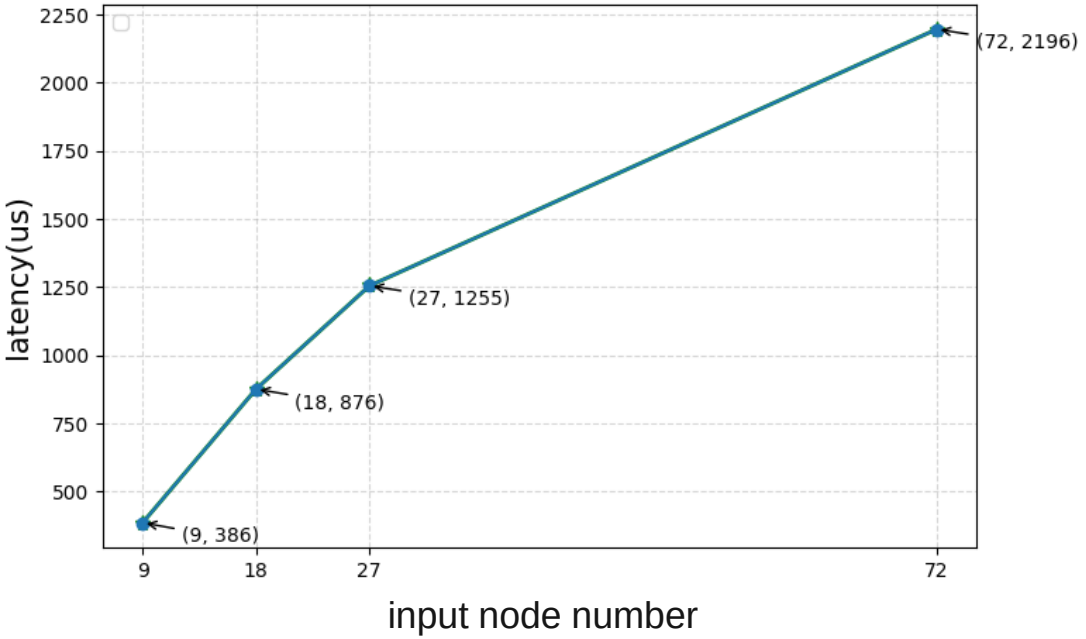
1. Data generator in PL output raw data
2. Use εNN to construct the graph in PL.
3. Denoising GNN deploy on AI engine which result will send back to PL
4. An ila at PL side will monitor the GNN result
5. **ILA**(Integrated Logic Analyzer), sample signal inside the FPGA

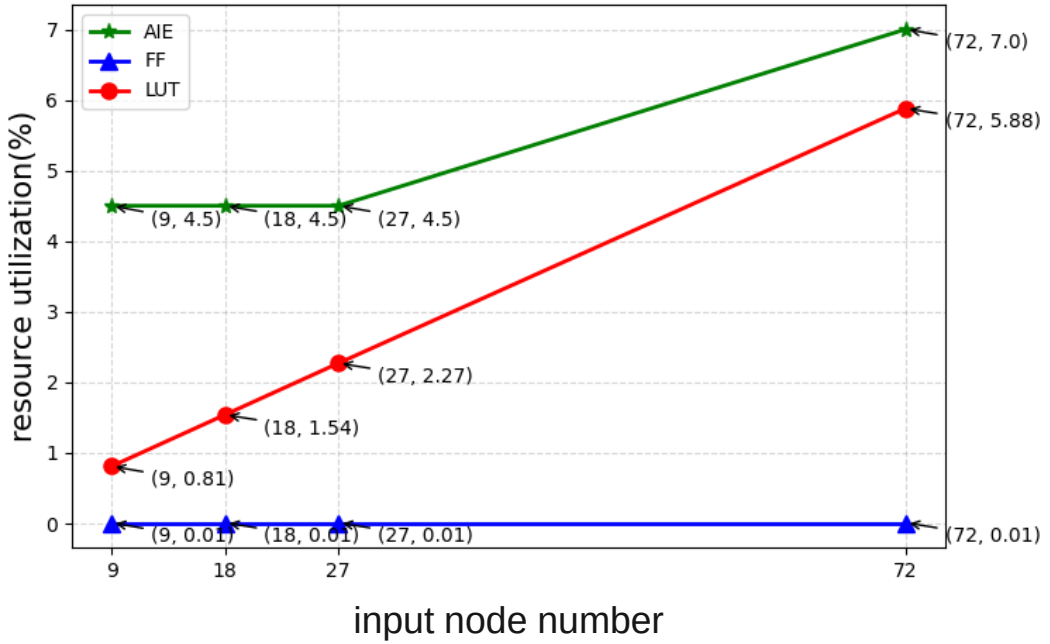




Real Latency and Resource Result

The structure of GNN is more complex, and the difficulty of deployment increases accordingly





One iteration is implemented for this demo, instead of 8 iteration in the real model

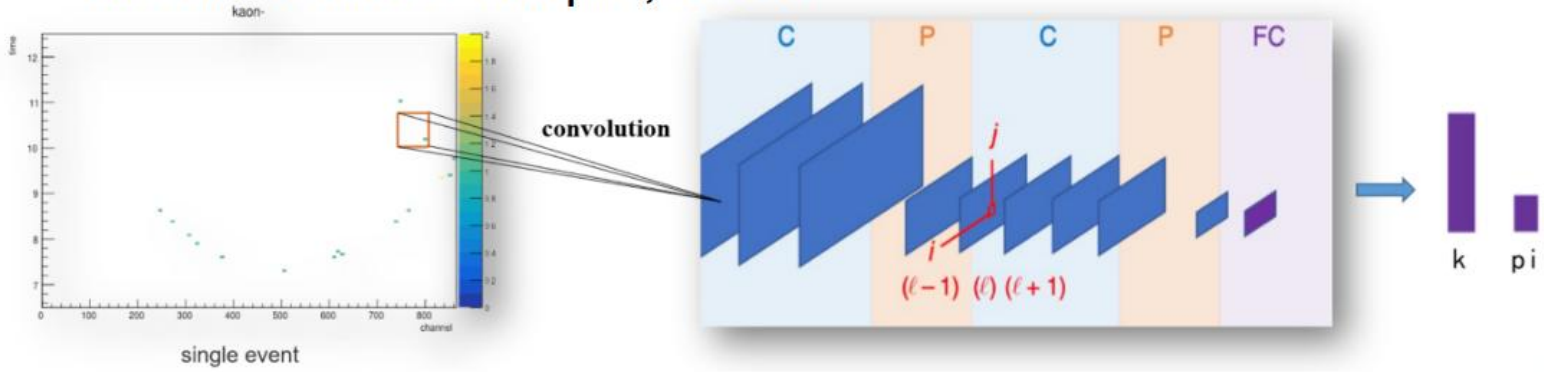
Latency: 2.2ms for 72 input nodes

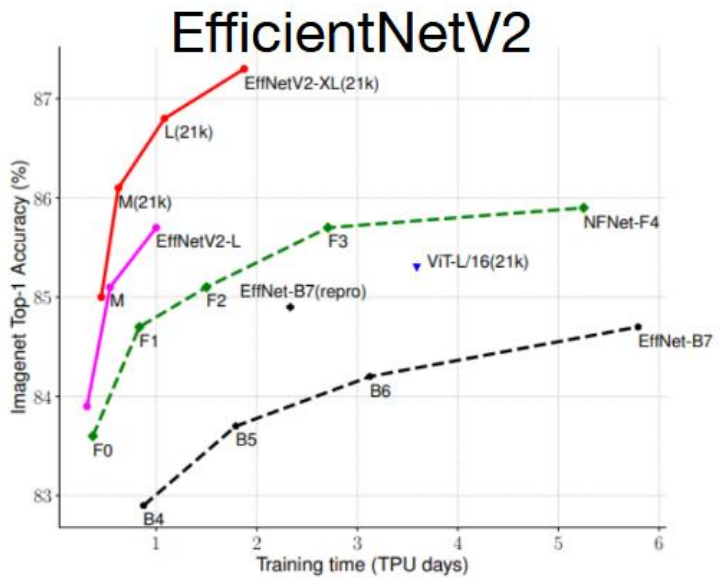
Resource usage: AIE: 7% LUT: 5.88% FF: 0.01%

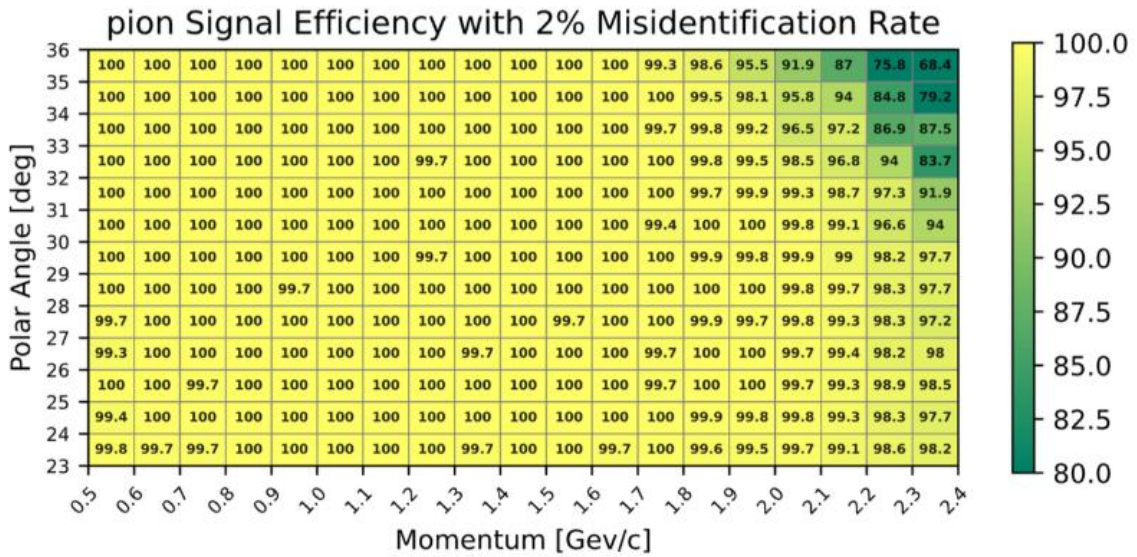
CNN algorithm for STCF PID

- DTOF as a PID subdetector of STCF
- CNN algorithm developed for Kaon/pion identification
- Kaon/Pion MC simple, 800w

Z. Yao et al.@SDU

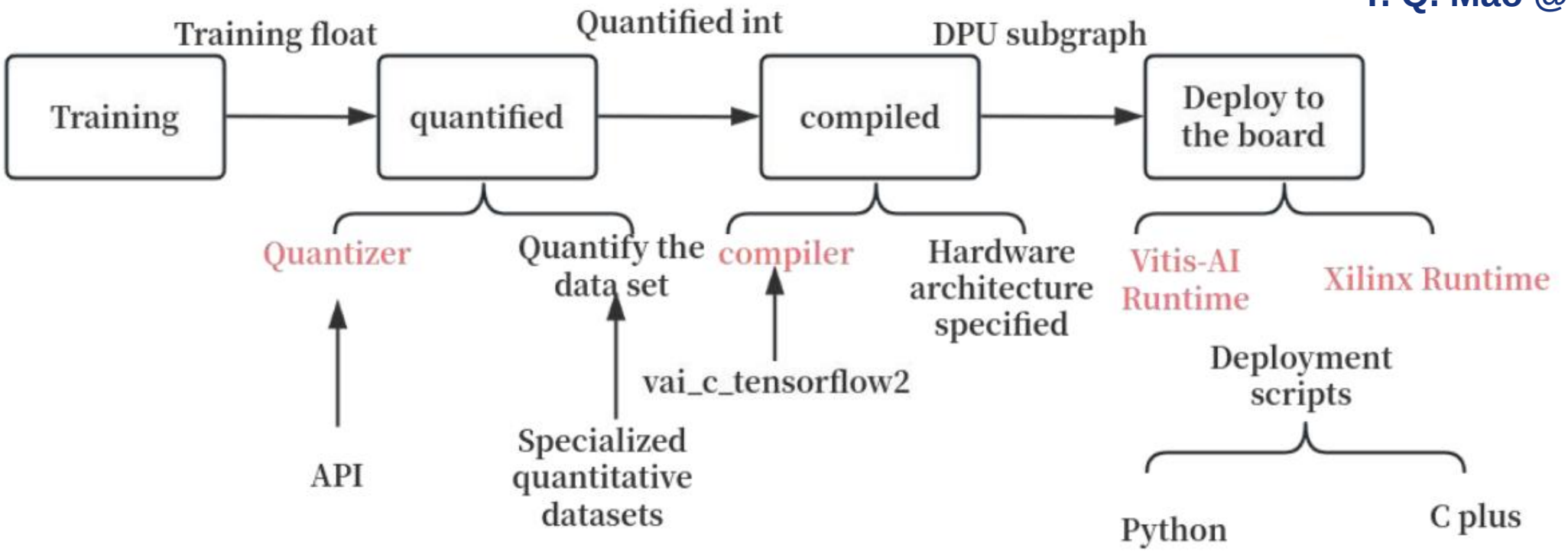






CNN implementation by Vitis AI

Y. Q. Mao @(SDU)



1.Training at the software level

2.Quantization and compilation using the Vitis-AI

3.Vitis-AI Runtime and Xilinx Runtime tools were used for deployment

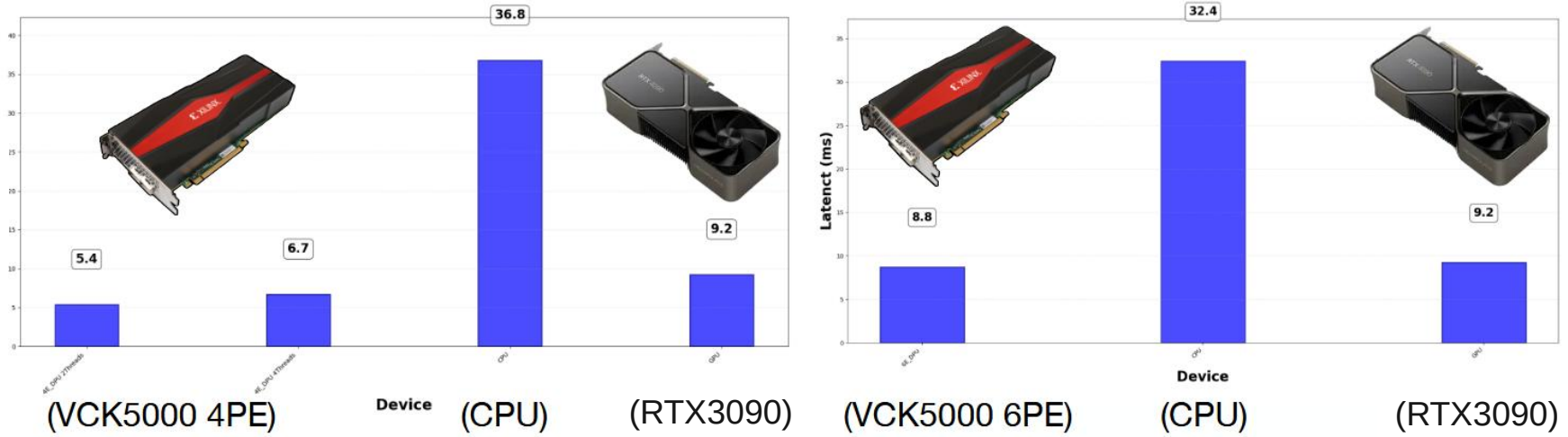
CNN algorithm inference

Inference result based on 10000 sample

- Batch size 4 or 6 due to DPU PE limitation

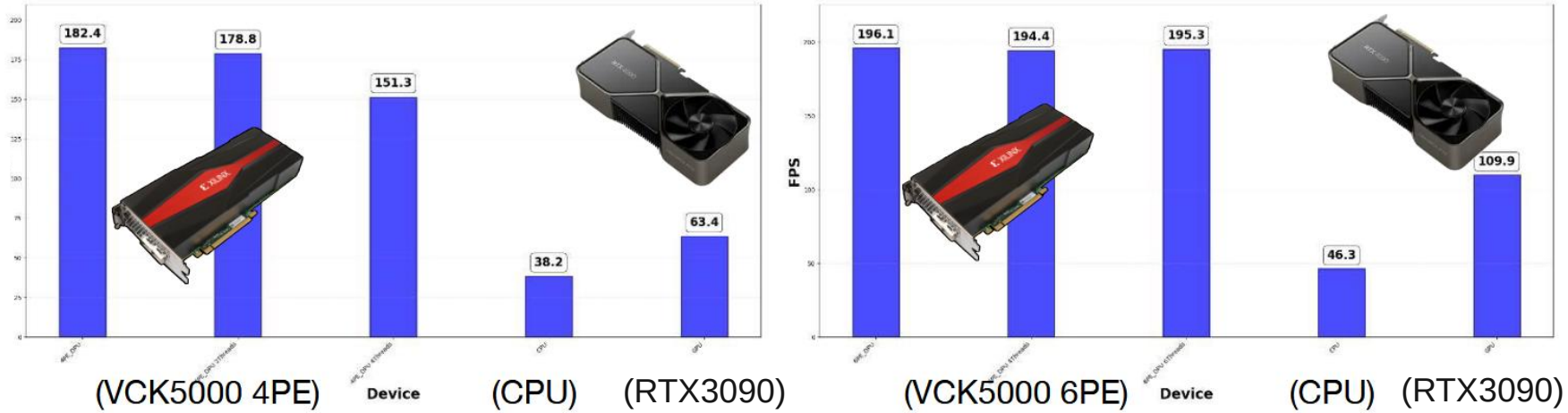
Y. Q. Mao @(SDU)

Latency (ms)



DPU based on Versal ACAP shows ~7 times(CPU)/ ~2 times(GPU) faster inference time

FPS



DPU based on Versal ACAP shows ~7 times(CPU)/ ~3 times(GPU) higher throughput

Summary

The neural network algorithms deployments on Versal ACAP has been made.

For DNN deployment

Best AIE DNN latency is about 2 us(MAC)

MMUL method shows more potential when input event number increases.

For GNN deployment

When input 72 nodes, latency is about 2.2ms

Resource usage is low(7% for AI engine, 5.88% for LUT)

For CNN deployment

EfficientNet b0 of model zoo deployed on the DPU of Versal VCK5000

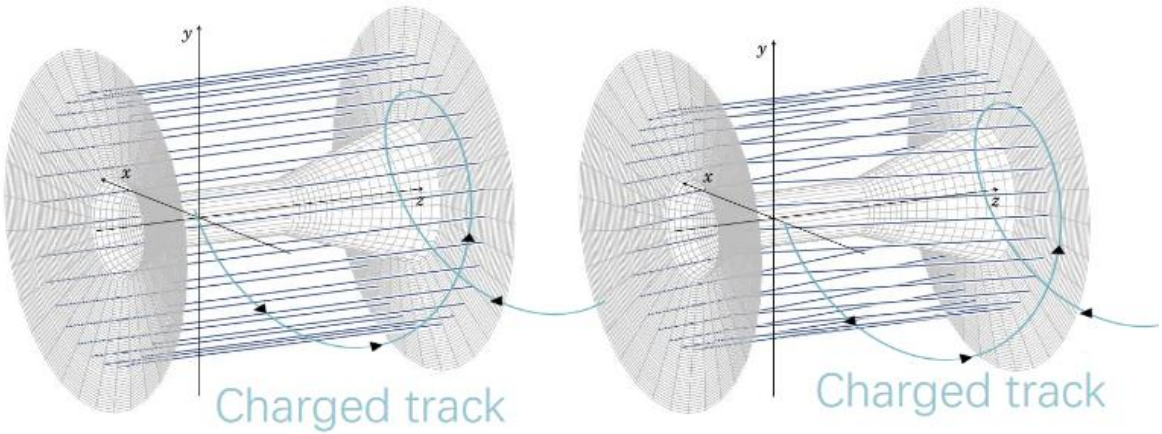
2 times faster inference time compared with the RTX 3090

3 times higher throughput compared with the RTX 3090

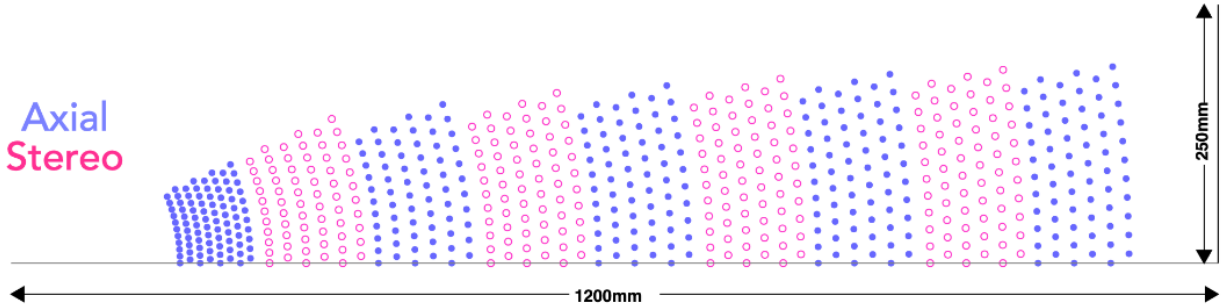
END

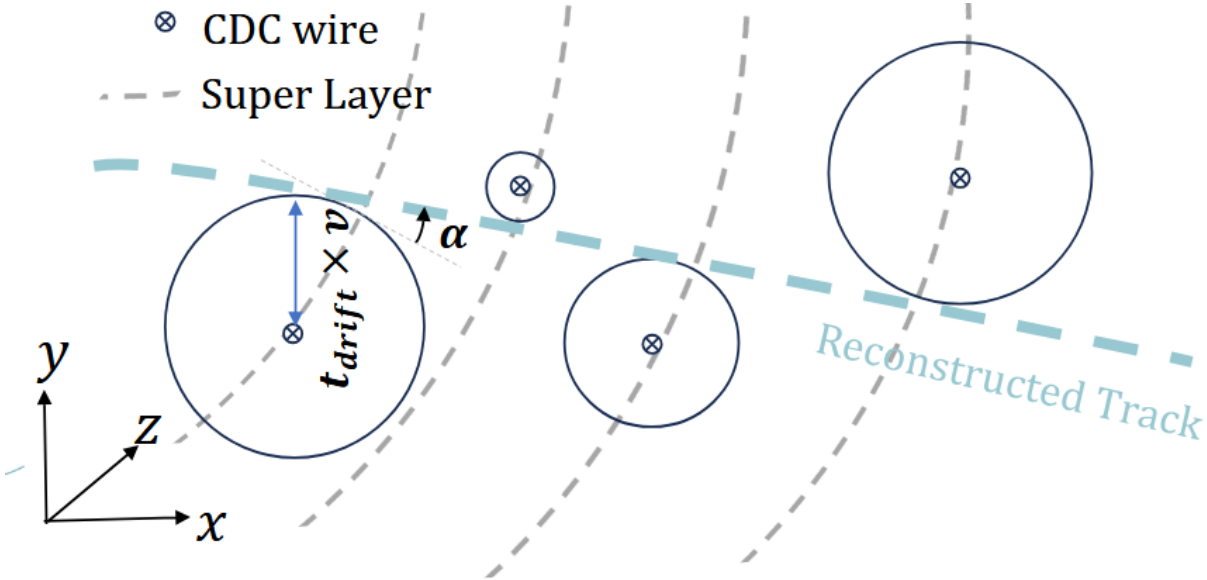
BACK UP

Central Drift Chamber(CDC)



Axial wires　　Stereo wires





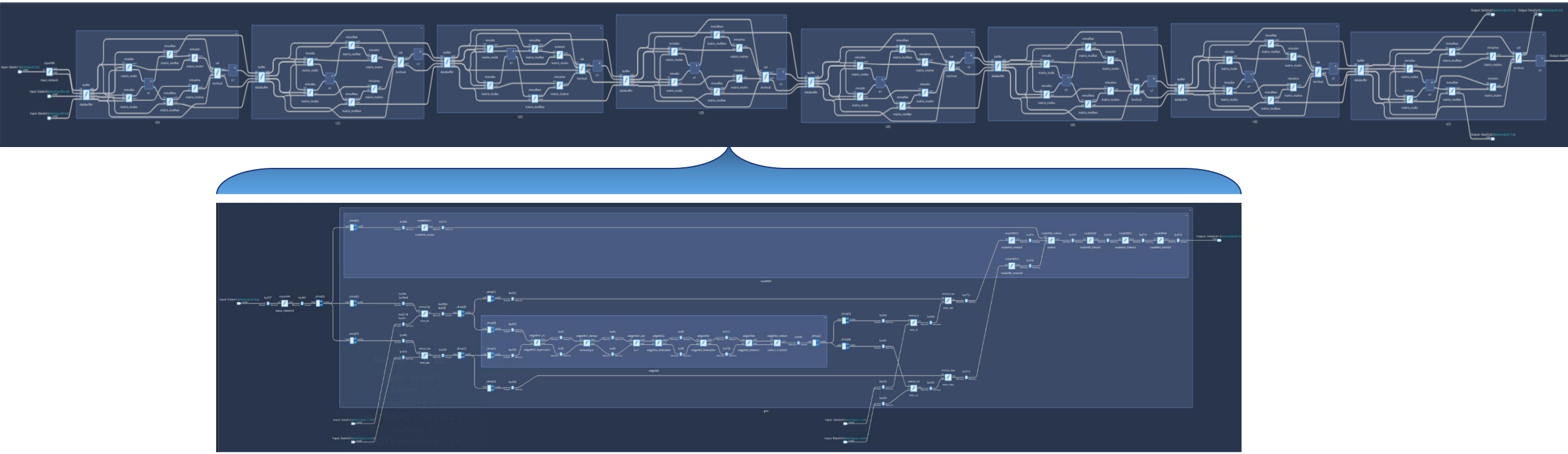
CDC

1. CDC is a gas detector
2. Electric field on the wire
3. Particle ionized gas
4. Electron after ionization drift towards the wire

Track information:

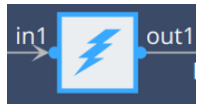
Location for CDC hits(x, y, z, layer)
TDC, ADC

8 layer denoising GNN structure



1. Offline GNN algorithm constructed by 8 layers of graph model
2. For the demo propose, one layer of graph model implemented on AIE
3. Futher model optimization needed for training a online GNN algorithm

AIE deployment

Kernel: 

basic coding unit

C++/C or **AIE API(vector datatype)** which width is fixed to 4, 8, 16, 32

In/out ports of kernel:

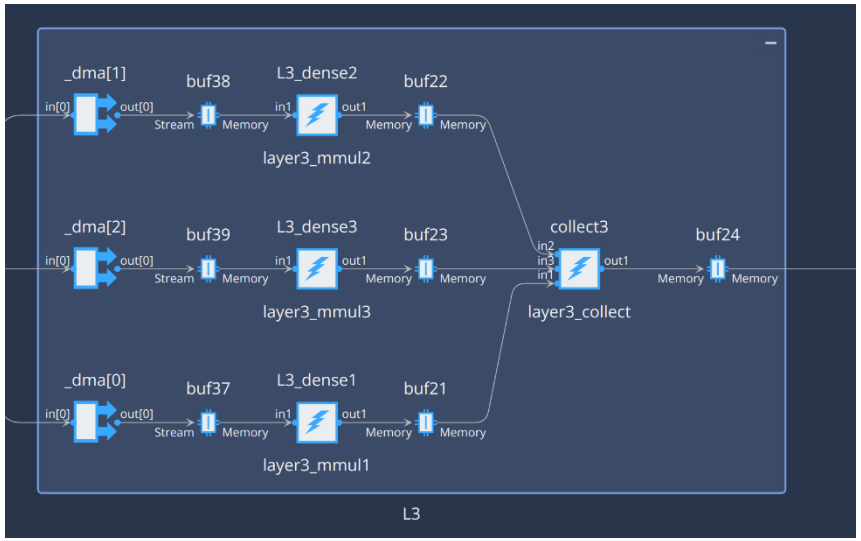
Buffer type:

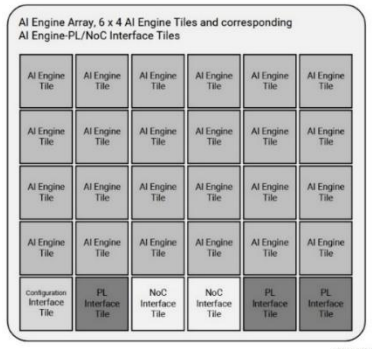
Read after store (extra time needed for storage)

Stream type:

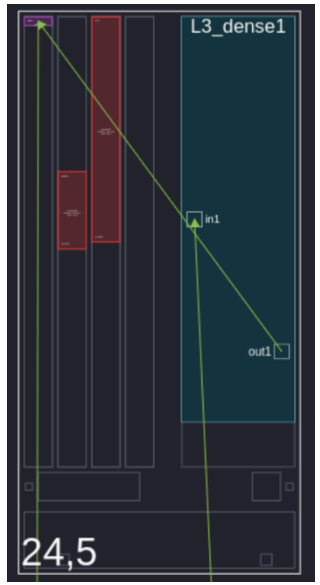
Free running if not stall

Graph:





AIE tile



AI Engine(AIE):

Where kernels placed(16KB program memory)

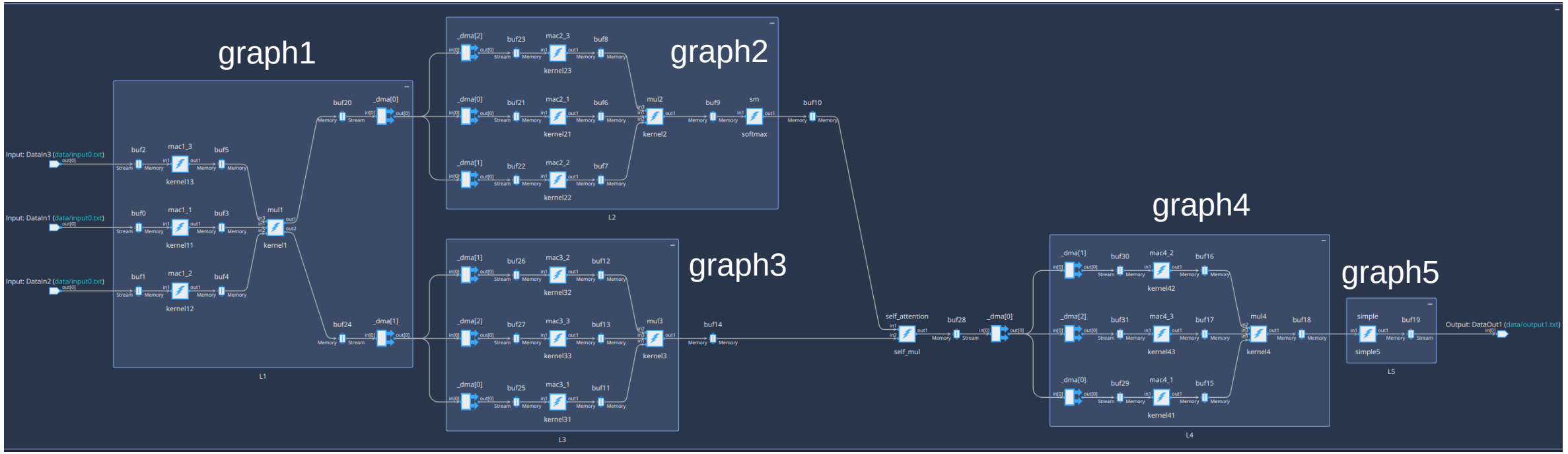
Memory Bank:

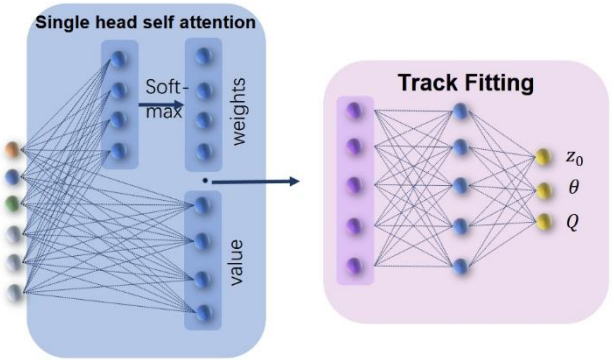
Place buffer (32KB data memory) (weight & bias)

Direct Memory Access(DMA) & Switch

Communication mechanism for non-neighboring AIE

Deployment on VCK190



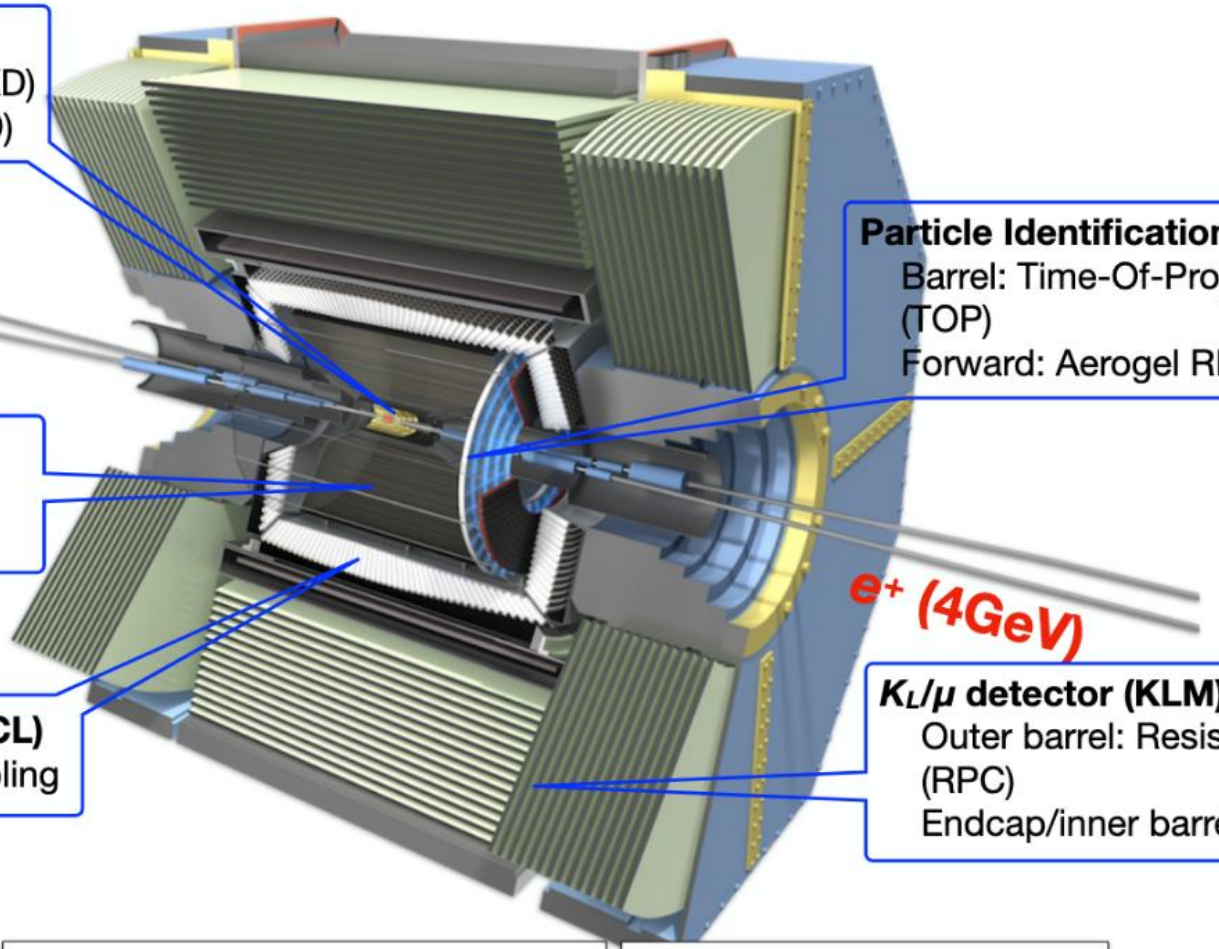


① To improve the parallelization, one graph represents one dnn layer.

② All math operation use the AIE API

③ AIE API is linear operation, so non-linear operation use approximations or LUTs

The Belle II detector



Vertex detector (VXD)
Inner 2 layers: pixel detector (PXD)
Outer 4 layers: strip sensor (SVD)

Particle Identification
Barrel: Time-Of-Propagation counters (TOP)
Forward: Aerogel RICH (ARICH)

e^- (7GeV)

Central Drift Chamber (CDC)
He (50%), C_2H_6 (50%), small cells, long lever arm

e^+ (4GeV)

ElectroMagnetic Calorimeter (ECL)
Barrel: CsI(Tl) + waveform sampling

K_L/μ detector (KLM)
Outer barrel: Resistive Plate Counter (RPC)
Endcap/inner barrel: Scintillator

Level-1 trigger system
CDC+ECL+TOP+KLM
L1 trigger latency 5 μsec

Data acquisition (DAQ) system
Maximum 30 kHz L1 trigger
1MB/event

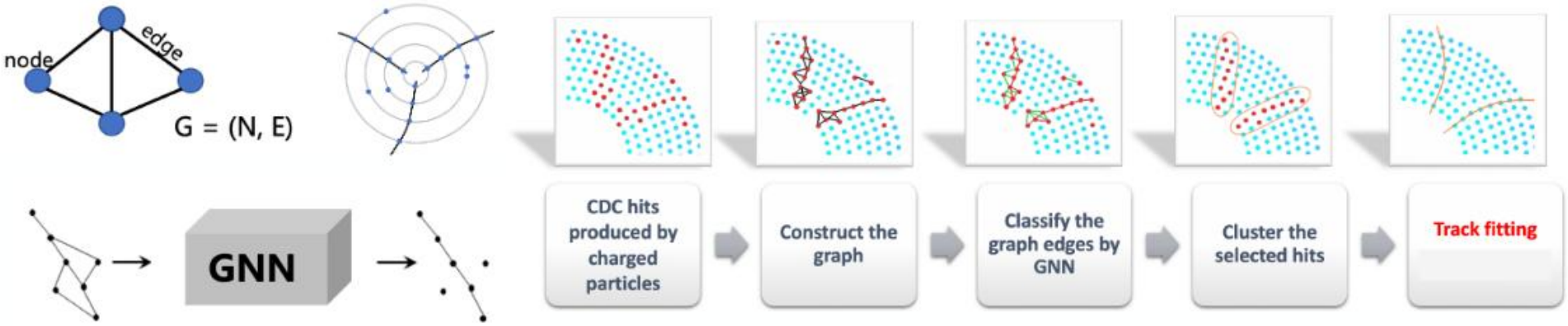
Computing system
GRID
Tens of PB / year

GNN for CDC background filter

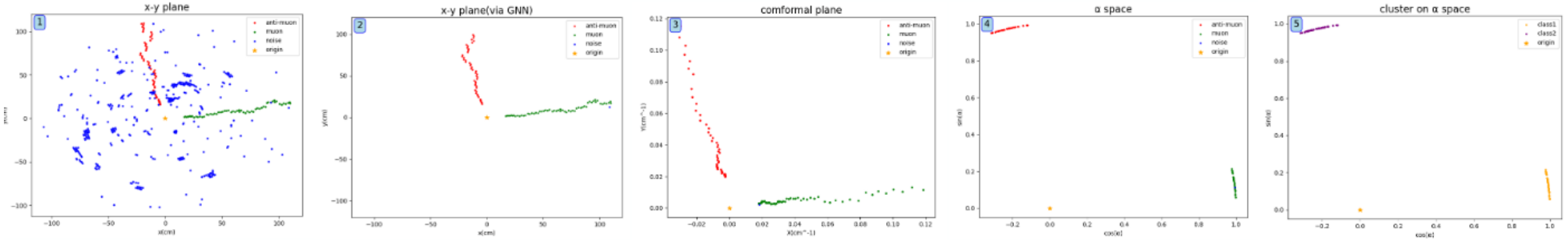
- Developed a GNN algorithm (based on BESIII's algorithm) for Belle II CDC hits clean up
  - Inputs: TDC, position coordinates r, ϕ

X. Q. Jia, X. Q. Hu et al (SDU)

Noise filtering → Clustering → Fitting



Simulation (own work)



μ+ μ- (particle gun) | GNN noise filtering | Transform space | Transform α space | DBSCAN clustering

Denoising GNN structure

Denoising GNN:

Online algorithm to filter background and keep signal

[X. Q. Jia(SDU) et al.](#)

Offline algorithm

Node:

Signal or background on wire

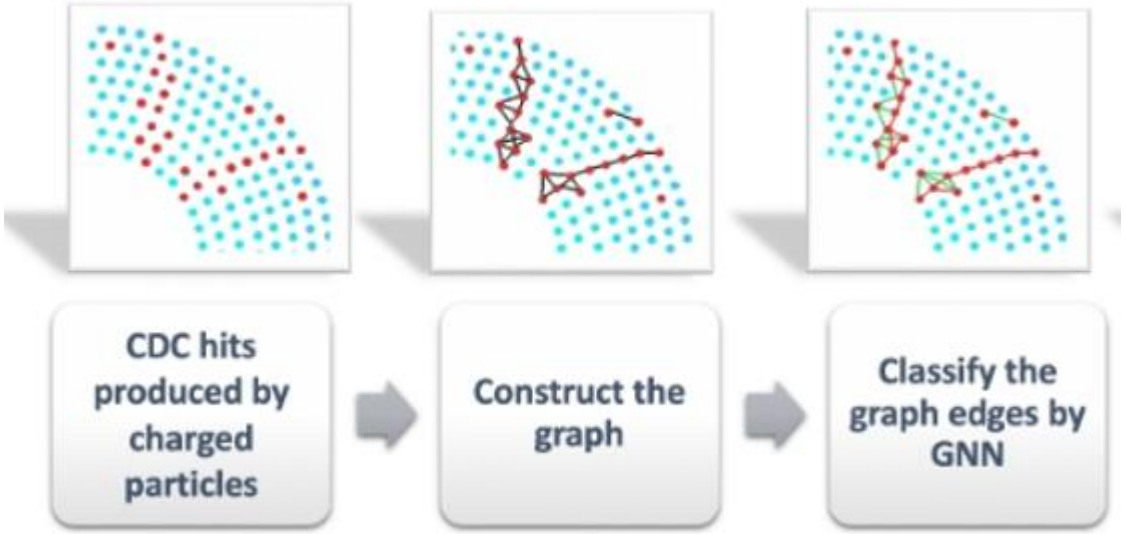
Edge:

Connect nodes

Edge classifier:

Edge connects nodes → true

Edge connects background → false



comparison