ParticleNet for Jet Tagging in Particle Physics on FPGA

Yutao Zhang，Yaodong Cheng，Yu Gao

Computing center, IHEP, CAS

2023-12-05

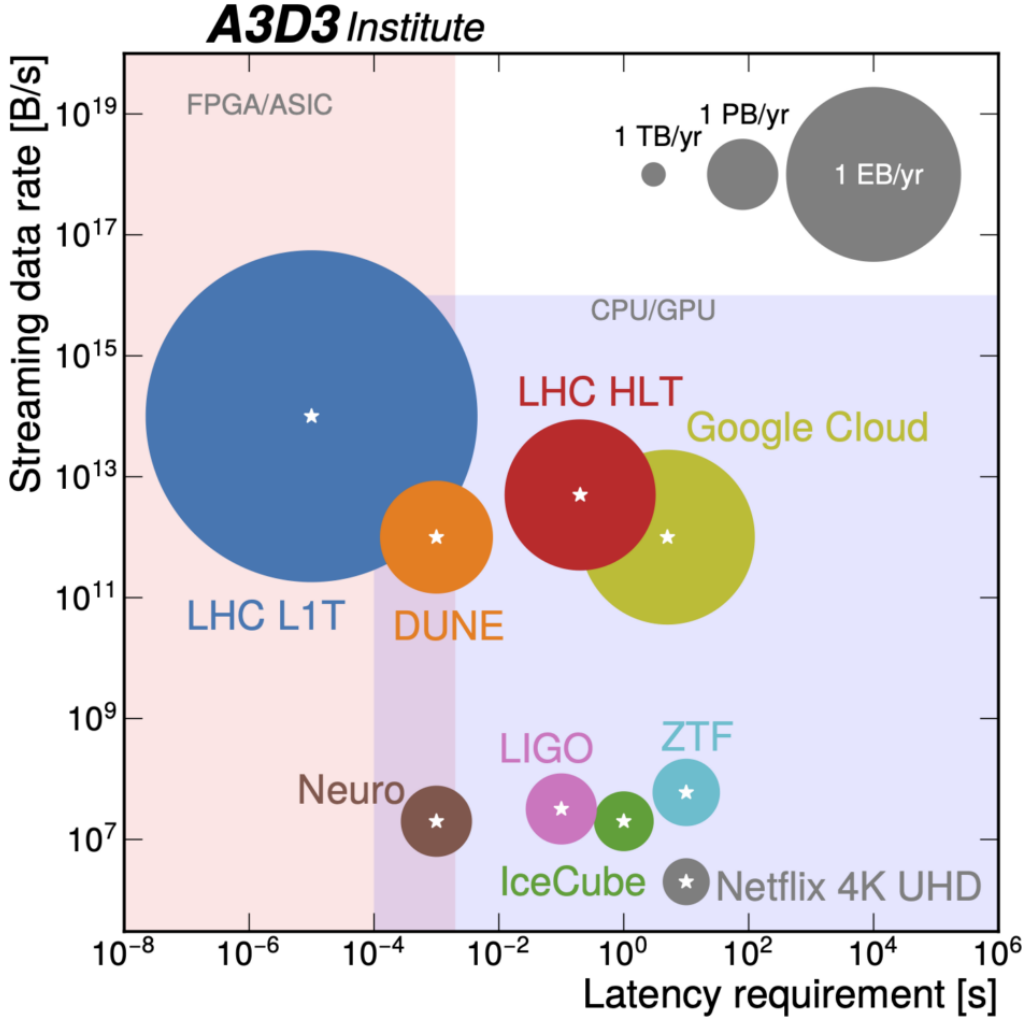
中国科学院高能物理研究所
Institute of High Energy Physics, Chinese Academy of Sciences

Outline

- Background and motivation
- Brief introduction to FPGA AI programming
- ParticleNet implementation on FPGA
- Summary

AI Algorithms Acceleration

- AI has emerged as a solution to efficiently analyze these massive data sets
- GPUs and FPGAs allow AI algorithms to be greatly accelerated
- The combination of AI and these processors is leading to a revolution in the way we analyze data, minimizing the time needed to perform the most advanced of analyses
- A3D3: Accelerated AI Algorithms for Data-Driven Discovery
- high energy physics, multi-messenger astrophysics, and systems neuroscience



A3D3 institue

AI and FPGA application in Particle physics

✓ Requirements

Low Latency

- Strictly limited by collisions occurring every 25ns

Low resource usage

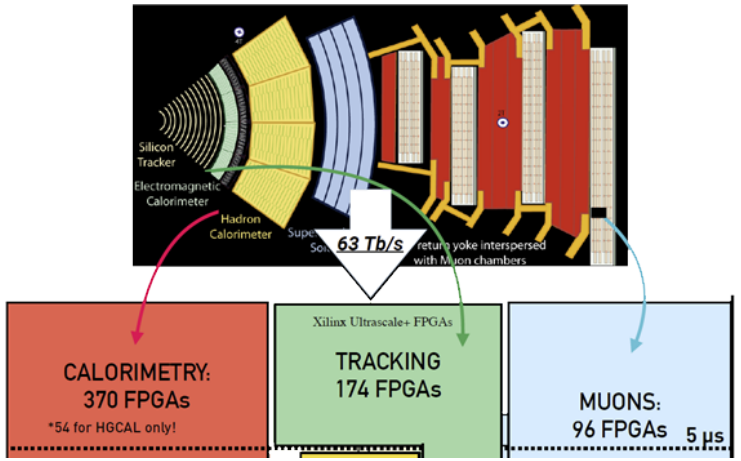
- Several algorithms in parallel on single device

High throughput

- System processing ~5% of total internet traffice

✓ AI and FPGA has widely applied in trigger and data processing

- Trigger, ml for data compression
- Simulation, ml for data generation
- Data reconstruction and analysis



✓ Why are FPGAs used?

Low Latency

- Resource parallelism and pipeline parallelism! High bandwidth

Latency deterministic

CPU/GPU processing randomness, FPGA repeatable and predictable latency

Power efficient

- FPGAs up to ~10x more power efficient than GPU

How are FPGAs programmed?

Hardware Description Languages (HDLs)

HDLs are programming languages which describe electronic circuits

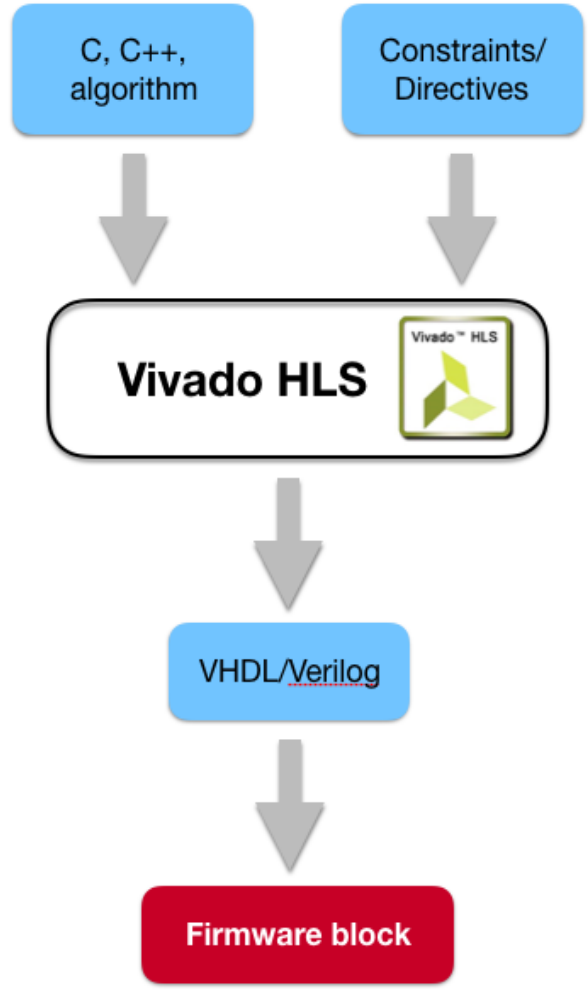
High Level Synthesis (HLS)

Compile from C/C++ to VHDL

Pre-processor directives and constraints used to optimize the design

Drastic decrease in firmware development time!

Currently we mainly use Xilinx Vivado HLS [*]



[*] https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf

AI Programming on FPGA

TensorFlow ONNX PYTORCH QKERAS

Model
(quantized/Pruned)

hls4ml

1. Convert model into internal representation

2. Write HLS project targeting specified backend

3. Run emulation

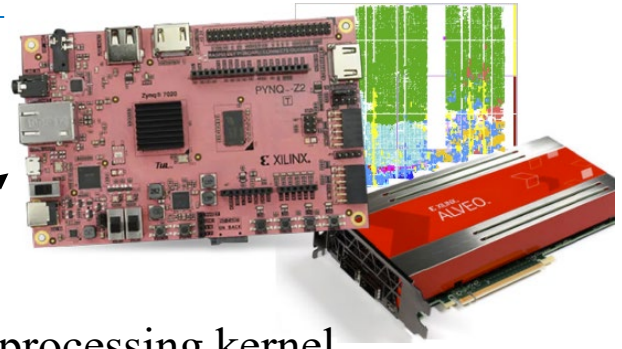
4. Run synthesis

Vivado/Vitis best supported
Intel Quartus
Intel One API
Mentor Catapult HLS
available soon

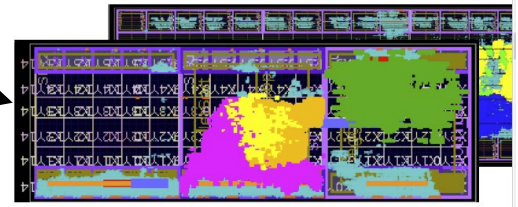
Vivado™ HLS

Mentor A Siemens Business

Intel HLS COMPILER



Co-processing kernel
(Xilinx accelerators and/SoCs)



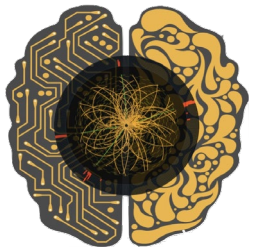
FPGA custom designs
(eg trigger algorithm)

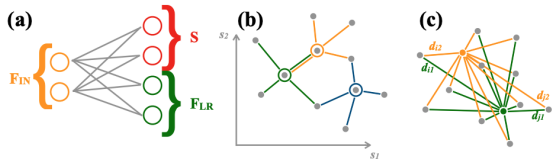
ASIC/NPU

hls4ml

- **Hls4ml**: high level synthesis for machine learning
- **hls4ml** is a package for translating neural networks to FPGA firmware for inference with extremely low latency on FPGAs
  - ✓ https://github.com/hls-fpga-machine-learning/hls4ml
  - ✓ https://fastmachinelearning.org/hls4ml/
  - ✓ pip install hls4ml
- **hls4ml** origins: triggering at (HL-)LHC
  - Extreme collision frequency of 40 MHz → extreme data rates O(100 TB/s)
  - Most collision "events" don't produce interesting physics
  - "Triggering" = filter events to reduce data rates to manageable levels
- **hls4ml** community is very active now
- Tutorial

https://github.com/fastmachinelearning/hls4ml-tutorial





Efficient NN design for FPGAs

FPGAs provide huge flexibility

Performance depends on how well you take advantage of this

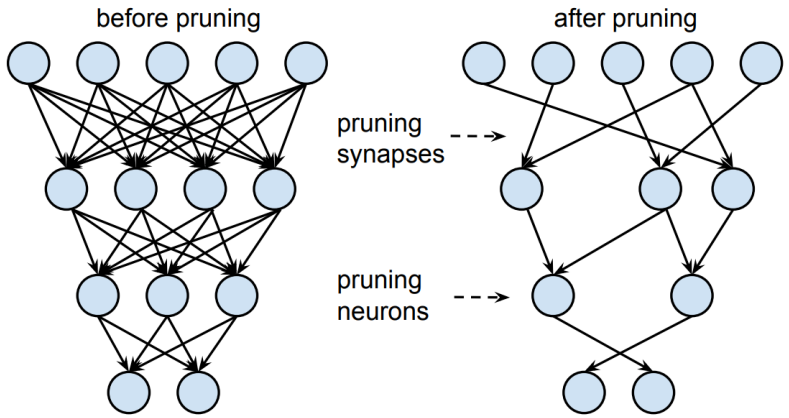
Constraints:

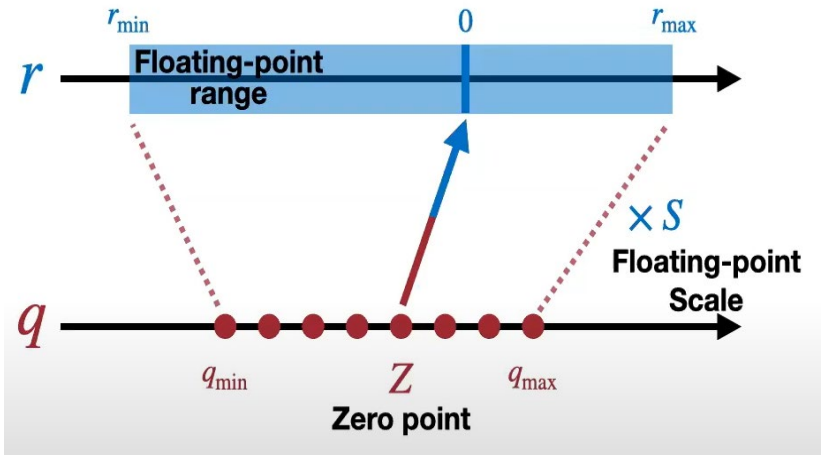
Input bandwidth

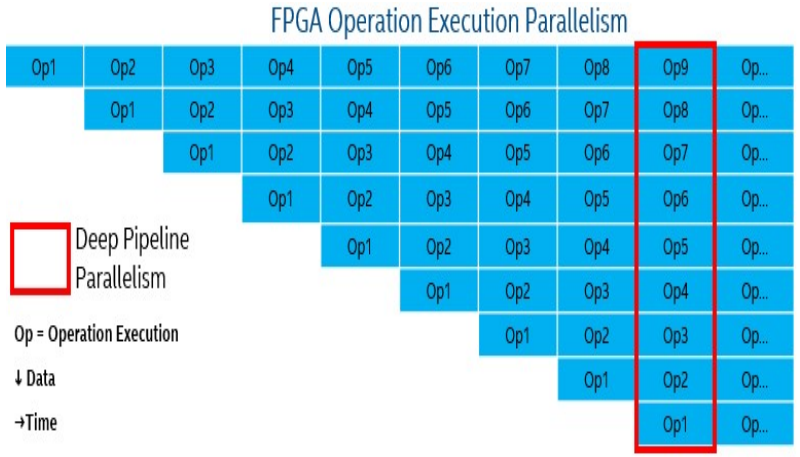
FPGA resources

Latency

Main methods to optimize the FPGA project







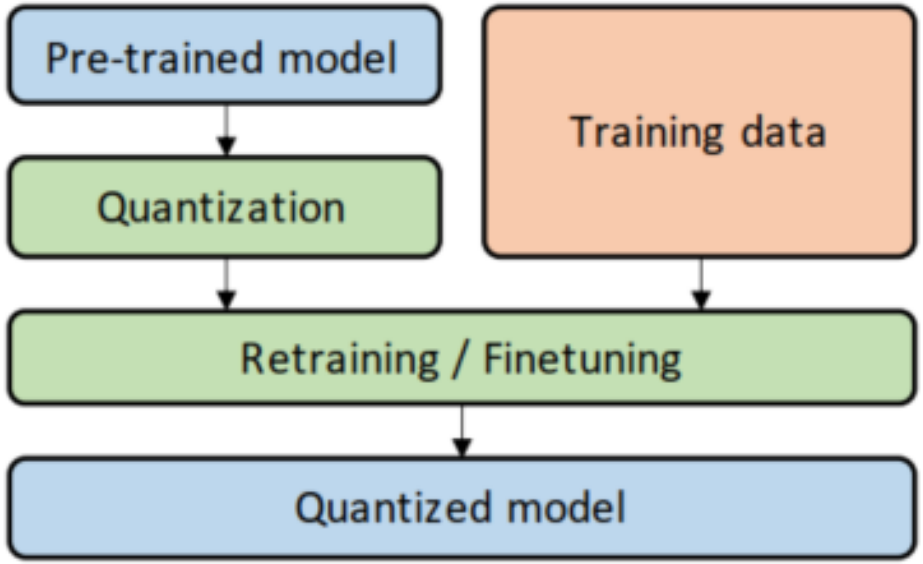
Pruning: reduce number of neurons

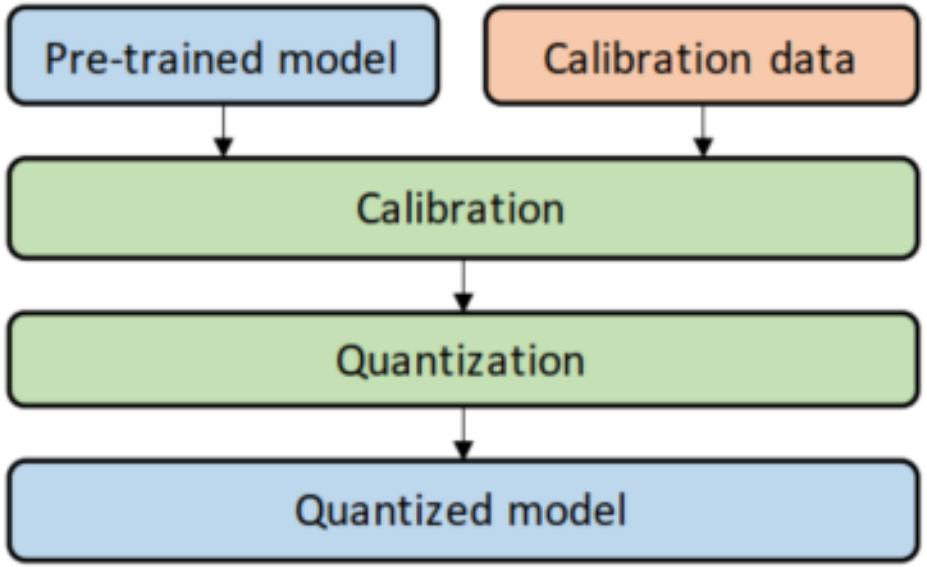
Quantization: reduce the precision of the calculations (inputs, weights, biases)

Parallelization : tune how much to parallelize to make the inference faster

Quantization – QAT vs. PTQ

- Quantization is a common technique used to reduce model size, though it can sometimes result in reduced accuracy

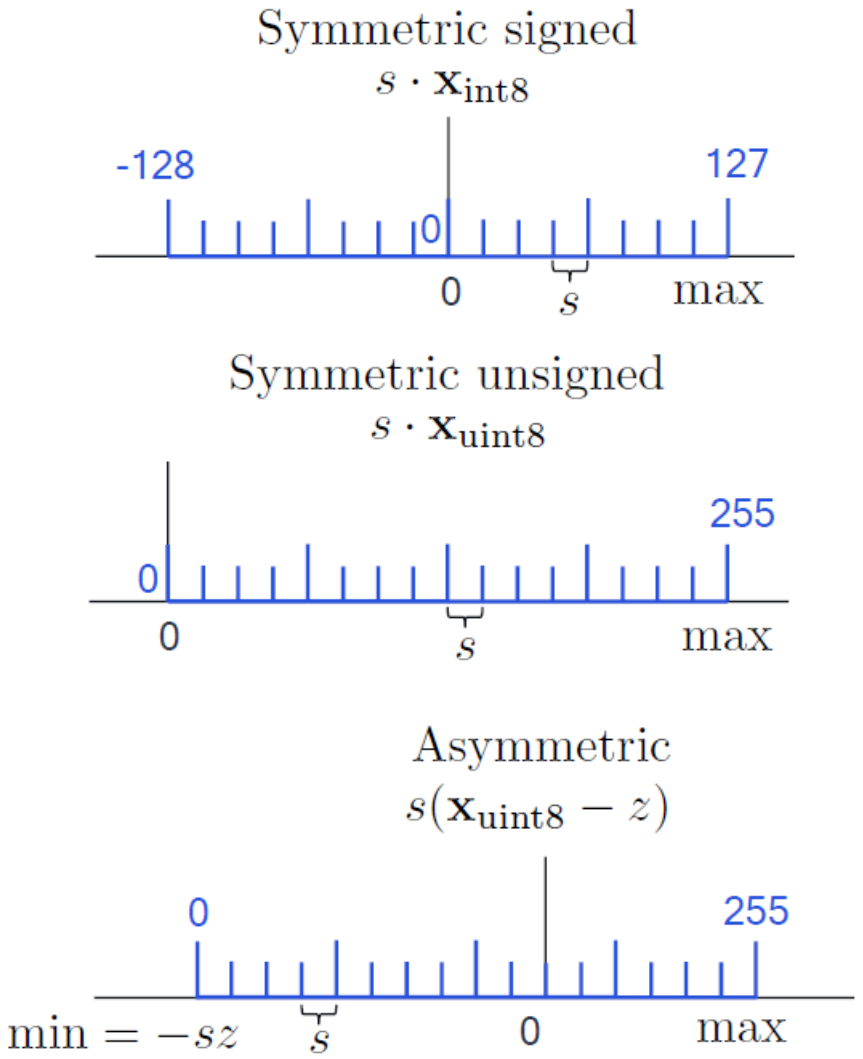




QAT: Quantization Aware Training
The network is further trained for few epochs in a process called Fine-Tuning/Retraining.
Without/Less sacrificing accuracy

PTQ: Post-Training Quantization
The model's weights and activations are quantized from high precision to low precision, such as from FP32 to INT8
Does not consider for the loss of accuracy

Quantization – Symmetric vs. Asymmetric



Symmetric :

Quantize: $Q = clamp(round(\frac{R_{float}}{S}))$

Dequantize: $R_{float} = S * Q$

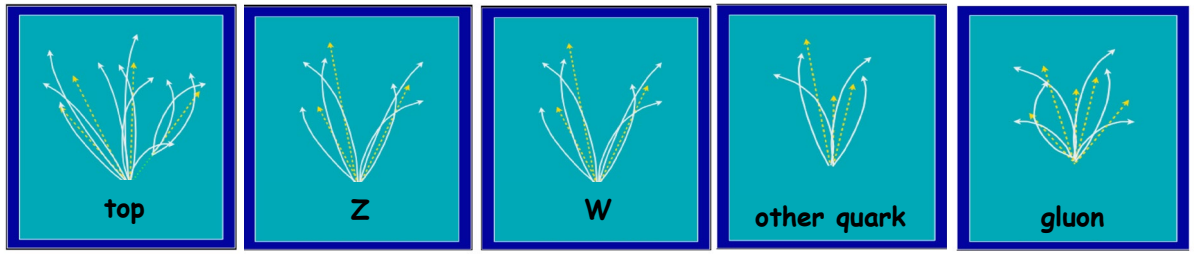
Asymmetric:

Quantize: $Q = clamp(round(\frac{R_{float}}{S}) - Z)$

Dequantize: $R_{float} = S * (Q + Z)$

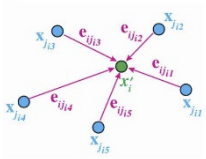
Physical case: Jet tagging

- Jet: collimated spray of hadrons initiated by energetic quarks or gluons, and they are ubiquitous at a hadron collider
- Jet tagging: identifying the hard scattering particle that initiates the jet, examples:
  - heavy flavor tagging (bottom/charm)
  - heavy resonance tagging (top/W/Z/Higgs)
  - quark/gluon discrimination
  - exotic jet tagging (displaced, 4-prong, …)

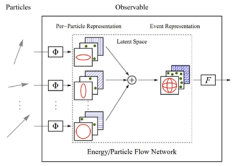


t→bW→bqq	**Z→qq**	**W→qq**	**q/g background**
3-prong jet	2-prong jet	2-prong jet	no substructure and/or mass ~ 0

- The rise of machine learning (ML) has brought lots of new progresses to jet tagging

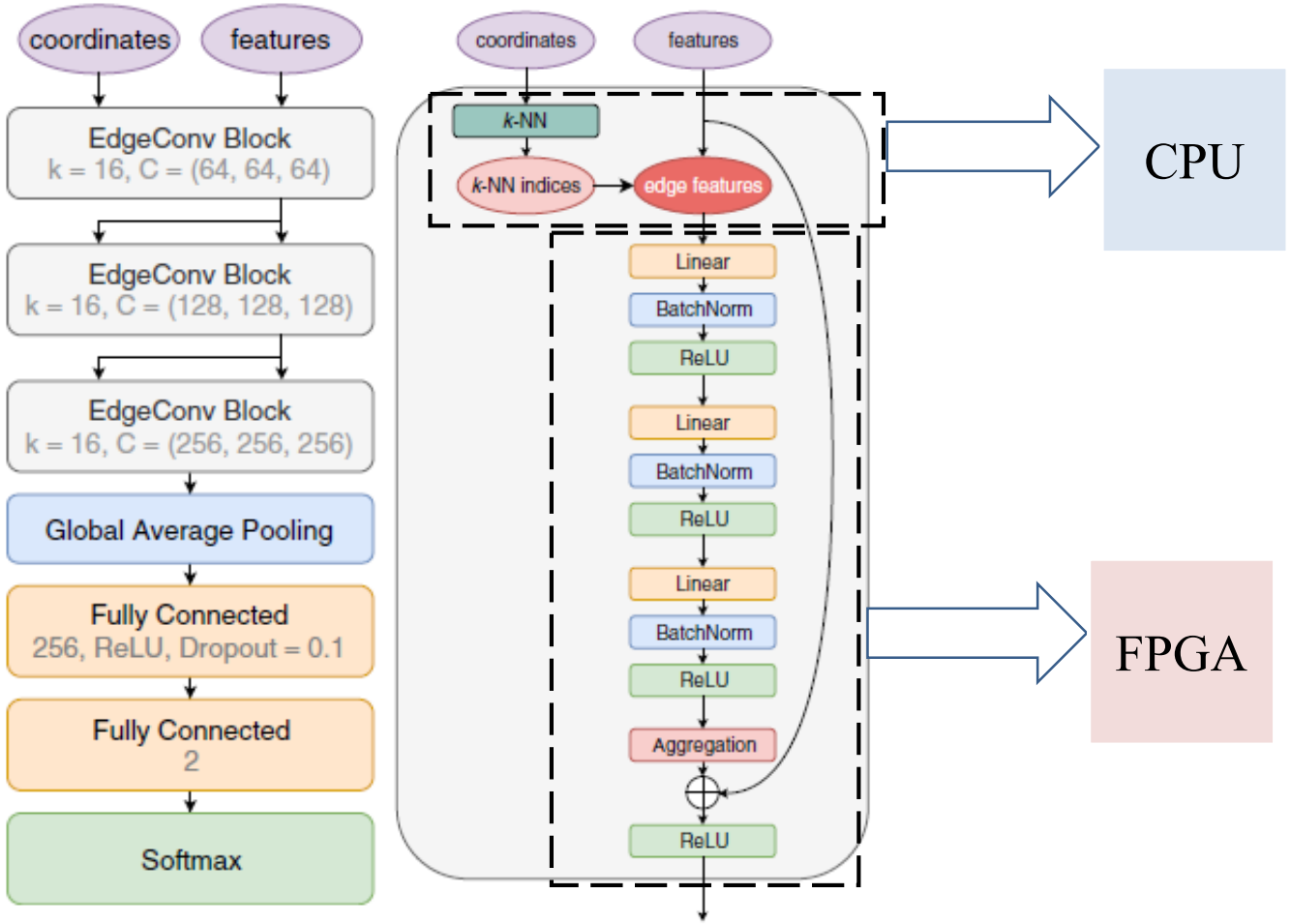


ParticleNet: jet tagging via particle clouds [arXiv:1902.08570]



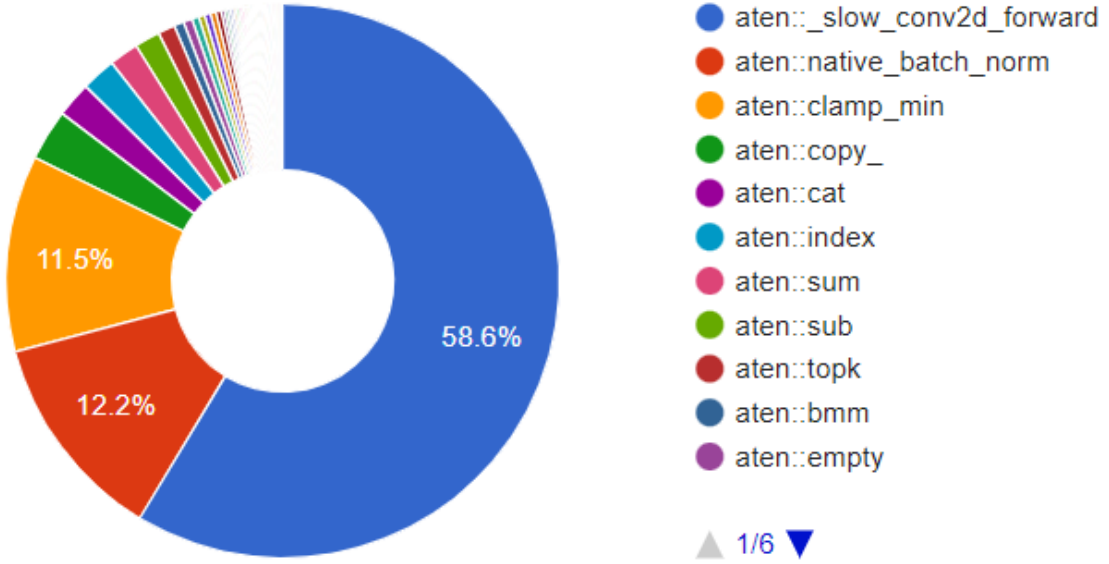
PFN: Particle Flow Network based on the Deep Sets [arXiv:1810.05165]

Workflow of ParticleNet



CPU

FPGA



Accounts for ~70% of the total time

✓ Using heterogenous solution with CPU and FPGA

Quantization of ParticleNet

- The quantization policy combined with Symmetric, Post-Training and Per tensor
- Symmetric: power-of-Two Quantization, $scale = 2^x$

	Parameters	ACC	Type
ParticleNet	**993KB**	**~93.9%**	**float32**
ParticleNet-quantization	**289KB**	**~93.3%**	**Int8**

$$ACC = (TP + TN)/(TP + TN + FP + FN)$$

Quantization implementation

Convert floating point calculation to fixed point calculation

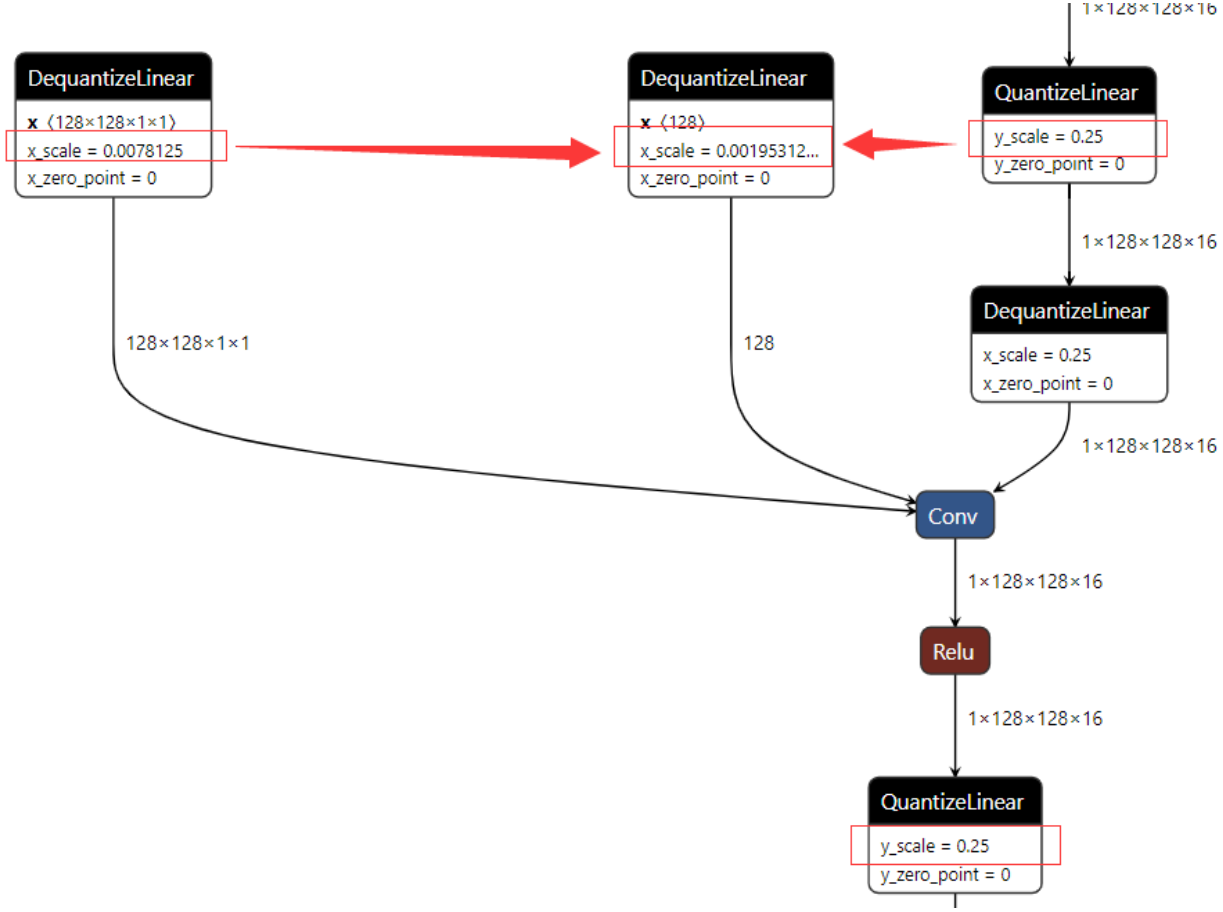
$$A_{n+1} = A_n \otimes W_n + B_n$$

$$Aq_{n+1}S_{a_{n+1}} = Aq_nS_{a_n} \otimes Wq_nS_{w_n} + Bq_nS_{b_n}$$

$$Aq_{n+1}S_{a_{n+1}} = (Aq_n \otimes Wq_n)S_{a_n}S_{w_n} + Bq_nS_b$$

$$\because S_{b_n} = S_{a_n}S_{w_n}$$

$$\therefore Aq_{n+1} = (Aq_n \otimes Wq_n + Bq_n)\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}}$$



Quantization implementation (II)

Convert floating point calculation to fixed point calculation

$$Aq_{n+1} = (Aq_n \otimes Wq_n + Bq_n)\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}}$$

Any 2^x operation can be further simplified in hardware logic as:

$S = 2^x$, **therefore:**

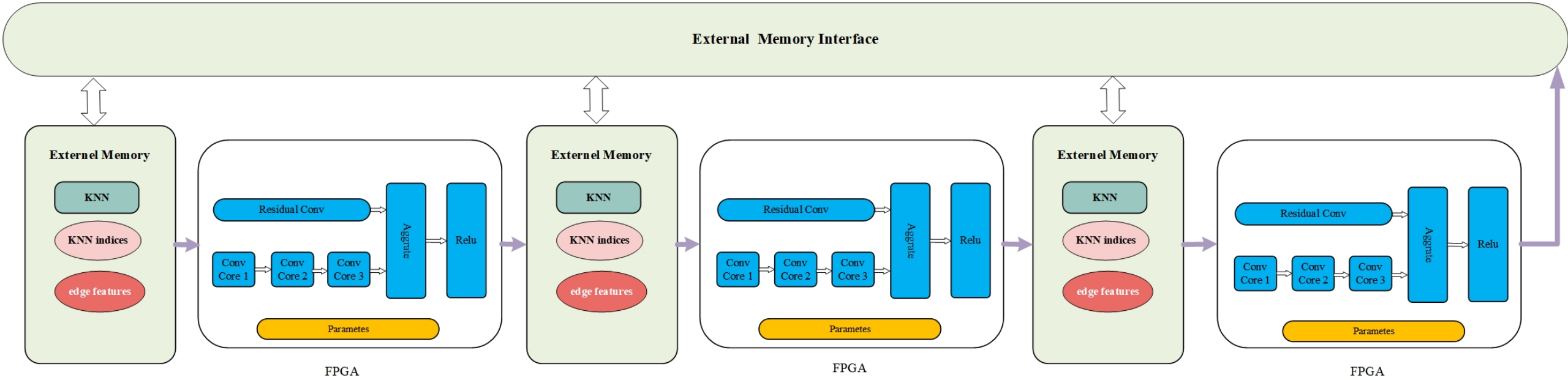
$$\begin{cases} R \times 2^x = R << |x|, x \geq 0 \\ R \times 2^x = R >> |x|, x < 0 \end{cases}$$

$$\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}} = 2^{a_n + w_n - a_{n+1}},$$

$$Aq_{n+1} = (Aq_n \otimes Wq_n + Bq_n)2^{a_n + w_n - a_{n+1}}$$

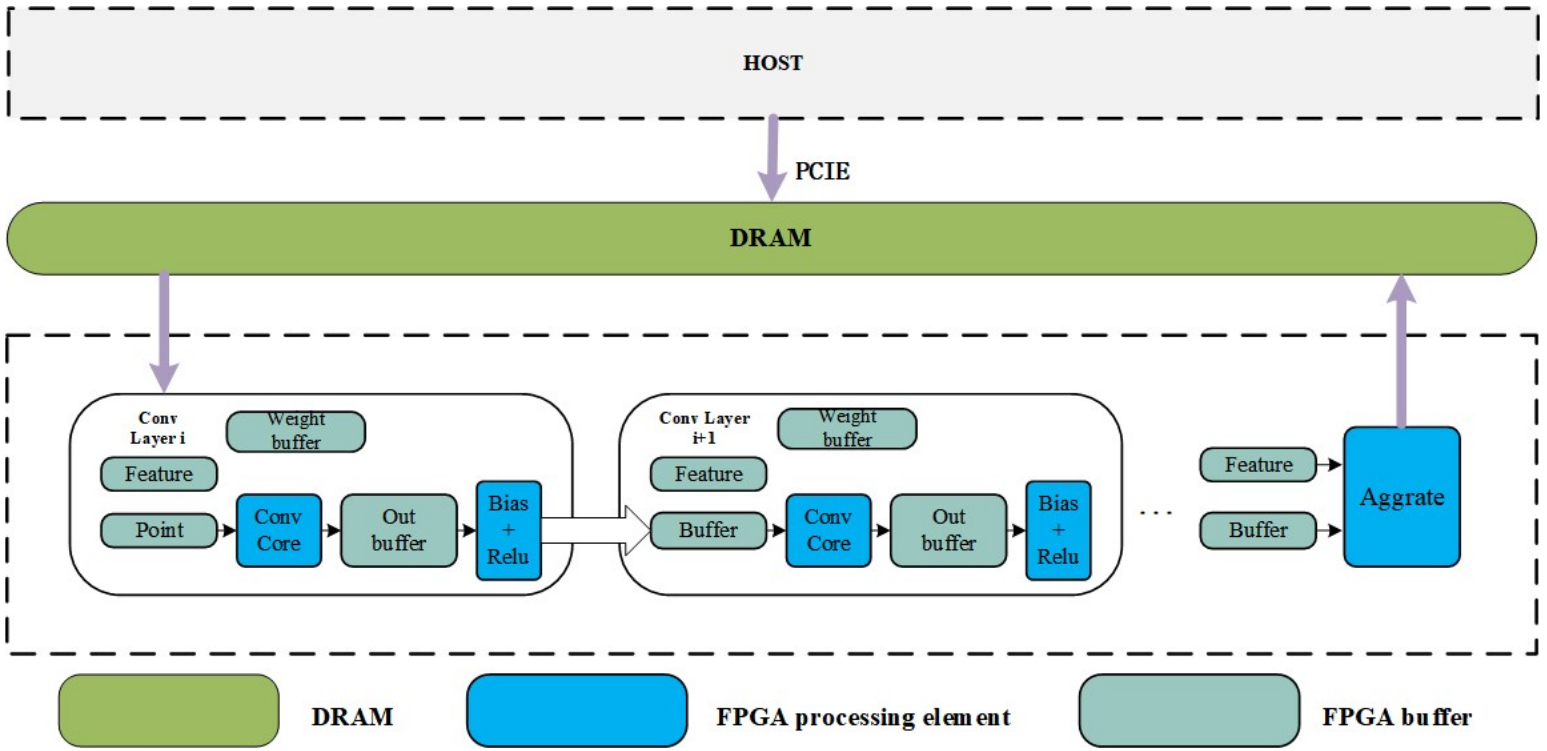
$$\log_2 \frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}} = a_n + w_n - a_{n+1}$$

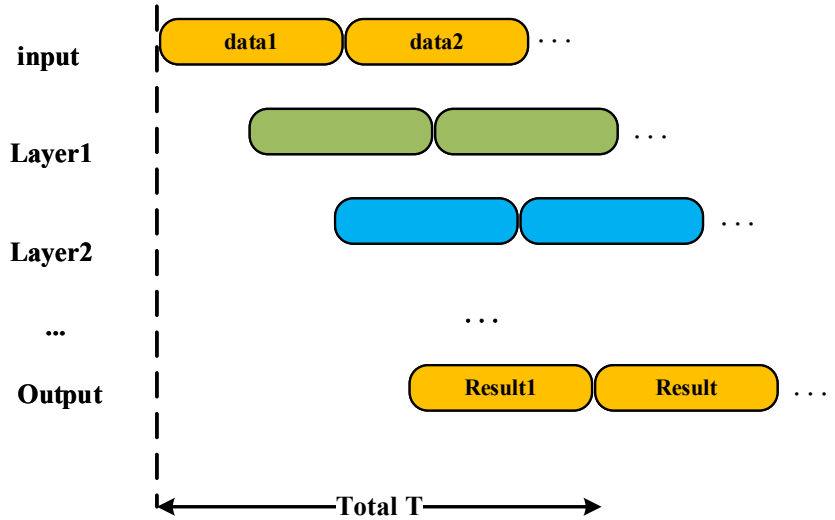
System Architecture



Overall System Architecture for Software-Hardware Co-design

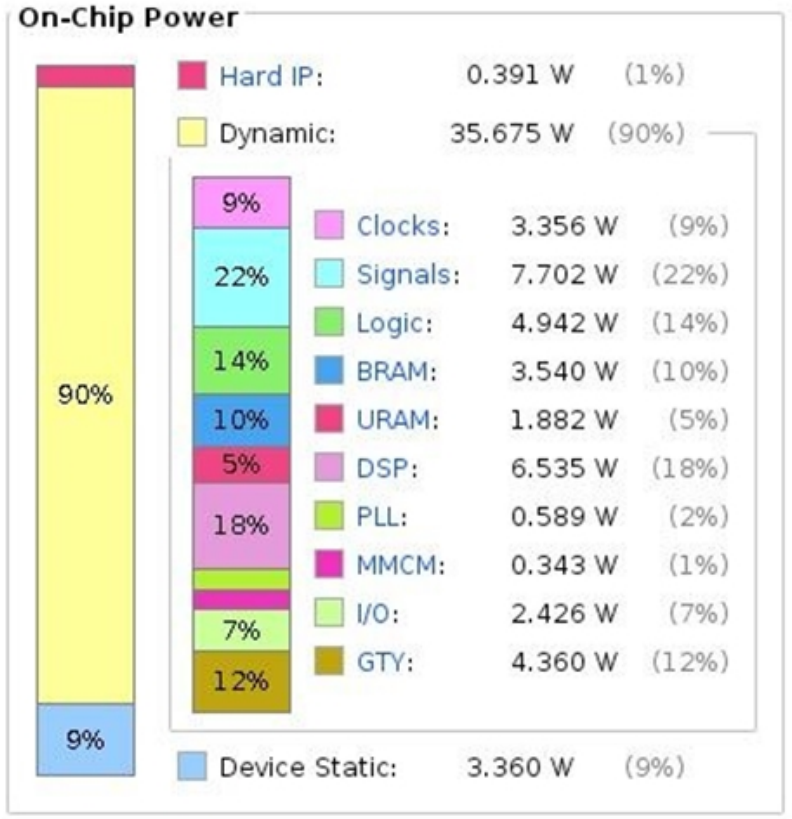
Pipeline parallelism

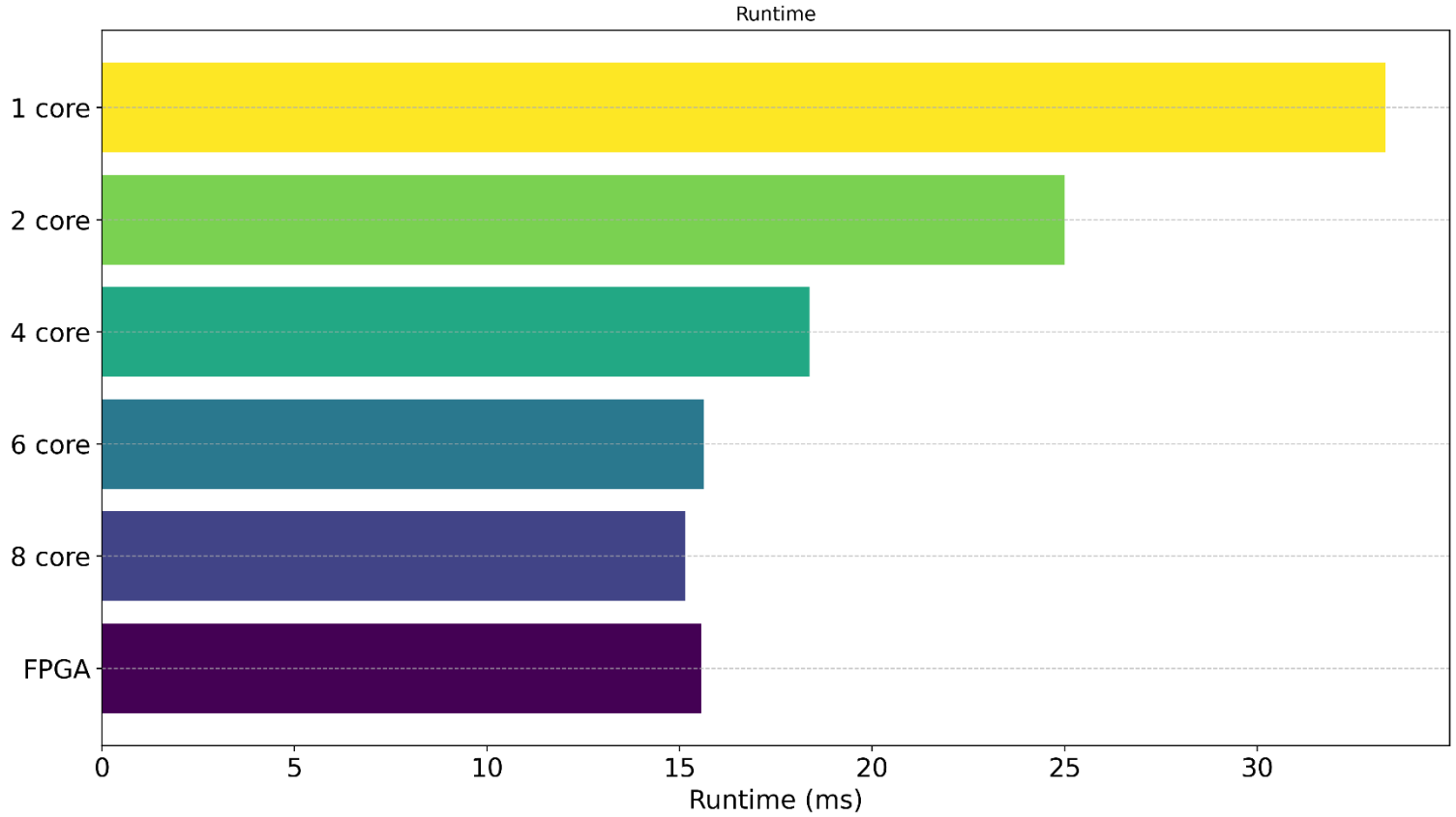




- ✓ A customized architecture to make full use of the resources of the hardware
- ✓ Each convolution layer is calculated in parallel without the need to write intermediate results back to external memory

Result

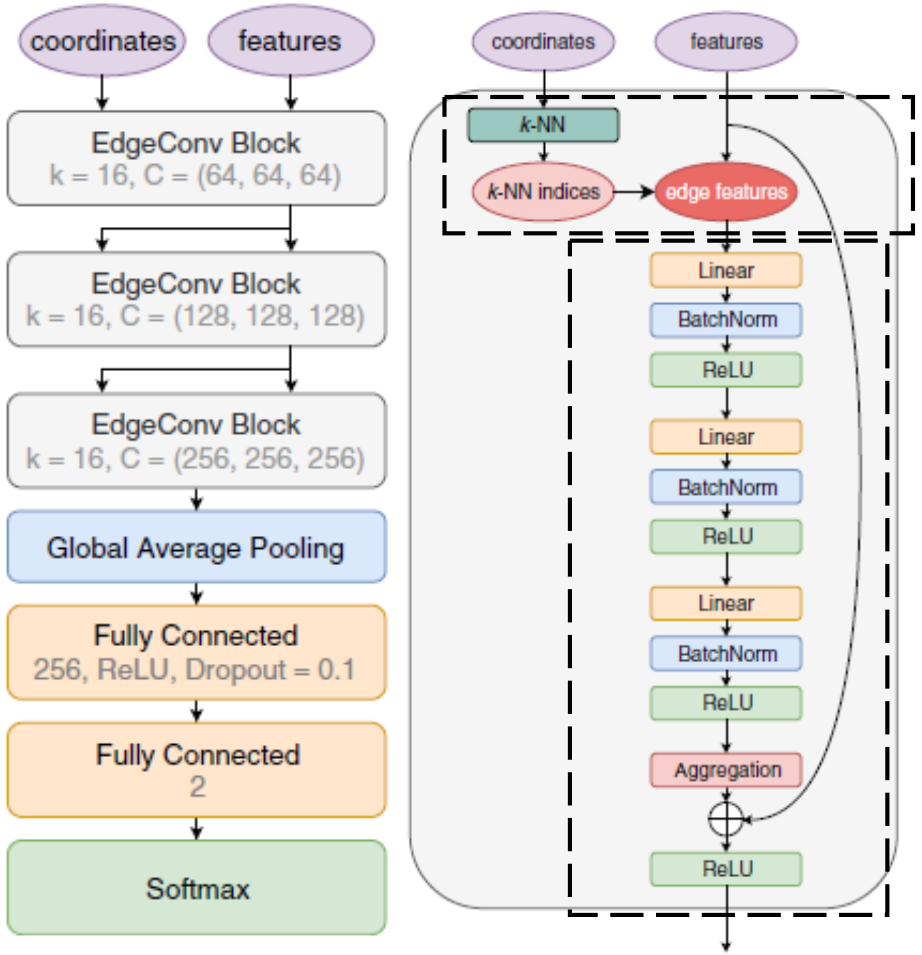




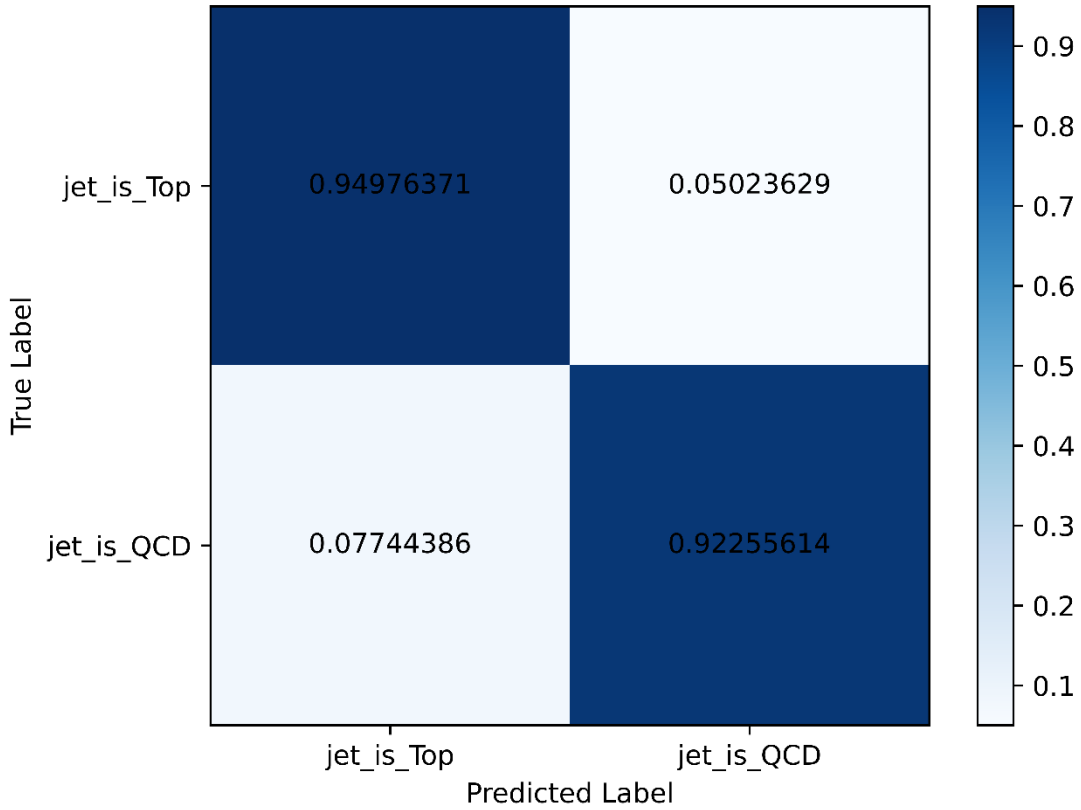
FPGA: Xilinx U200

CPU:Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz

Try

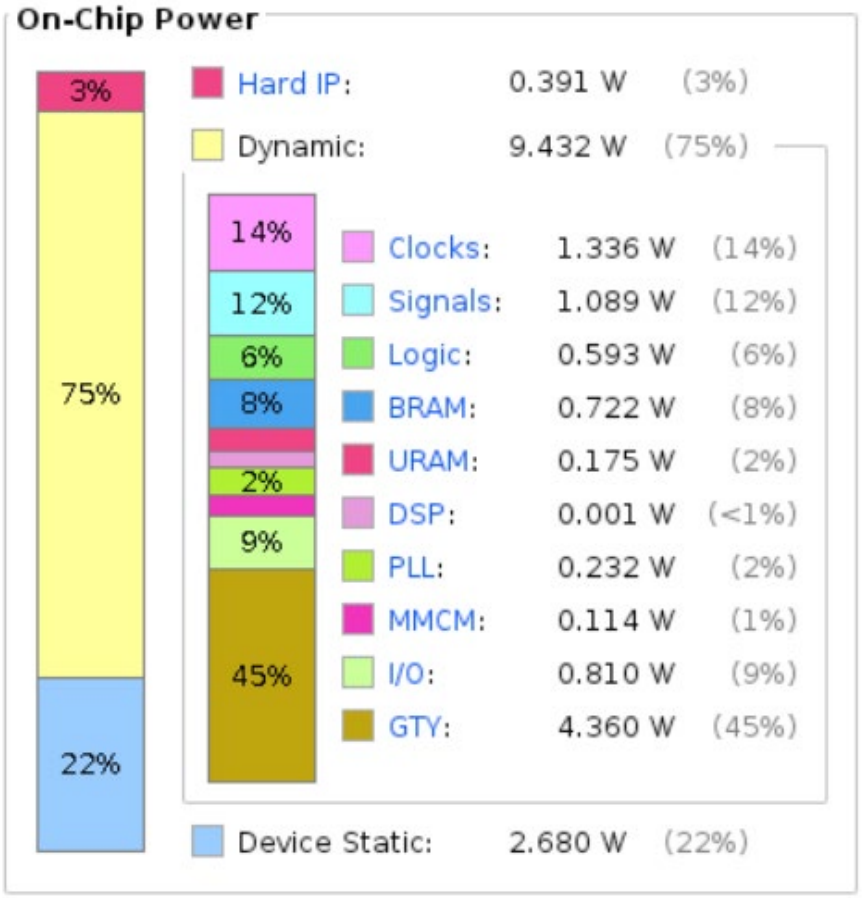


The first EdgeConv have the k-NN,edge features, other EdgeConv not have. Call “Static ParticleNet”

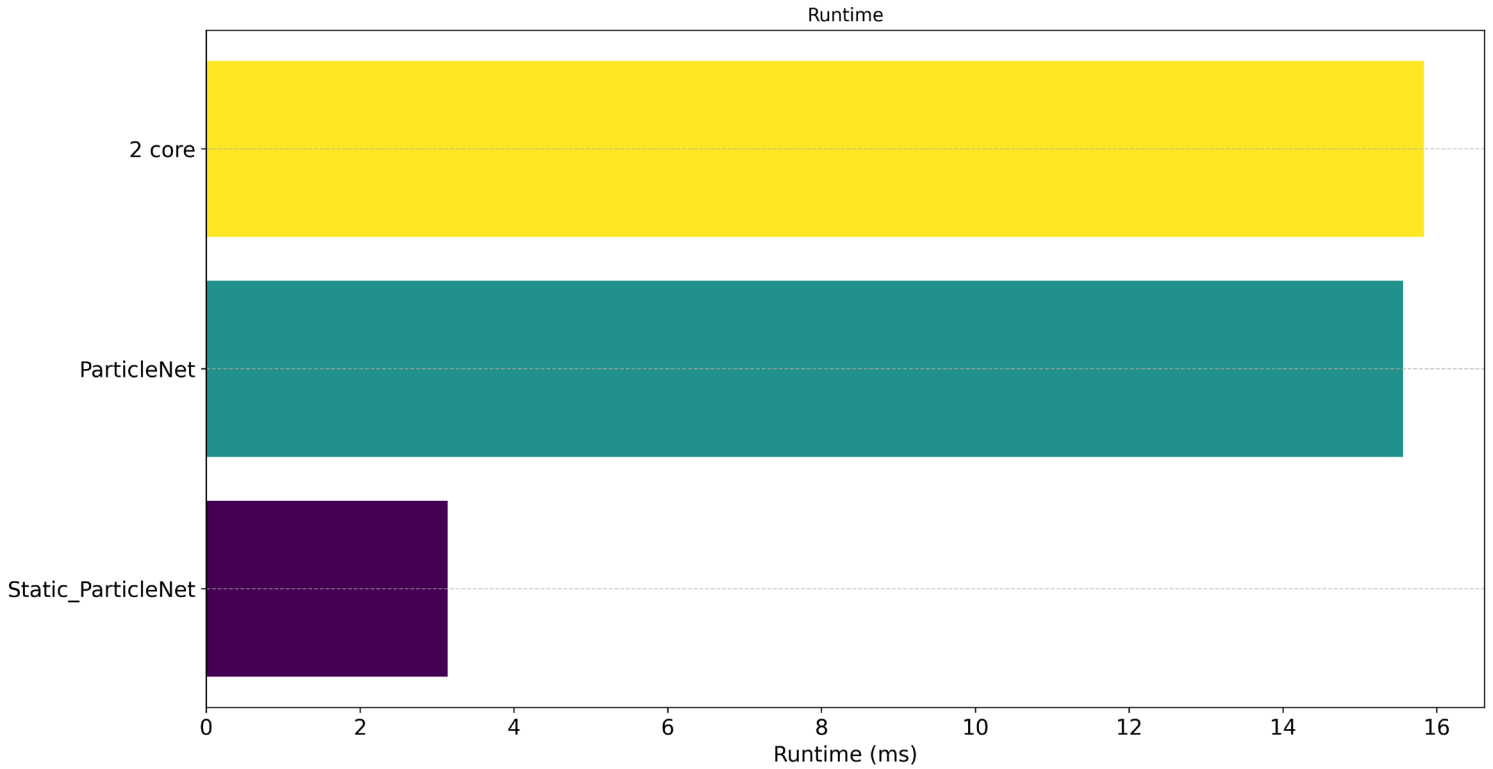


Top tagging datasets

Result for Static ParticleNet



Power only 9.432w



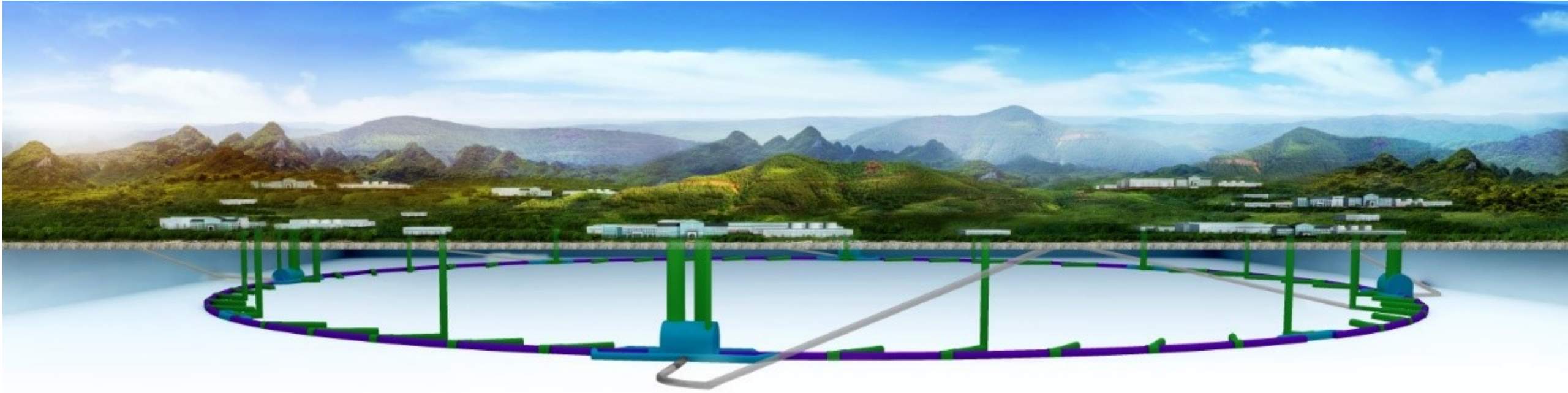
latency only 3.13ms

Summary

- FPGA as a coprocessor can accelerate the AI inference dramatically with low latency and power consumption
- Some projects such as hls4ml greatly simplifies the difficulty of programming
- We have tried to implement particleNet on FPGA

Acknowledgement

- The majority of the work was completed by our team members, including Yutao Zhang, Yaodong Cheng , Yu Gao and so on. I thank them for their outstanding work.
- Some of the materials in this presentation are sourced from projects like fastml and hls4ml, or Internet, and I would like to express my gratitude. If there are any copyright concerns, please notify me.



Thank you for your attention

zhangyutao@ihep.ac.cn