Co-Design for Ultra-Small ML on FPGAs with an Open-Source Toolchain and AI Agentic Workflow

Speaker: Qibin LIU (SLAC)

IEEE Real-Time 2026, Elba

Collaboration work with Larry Ruckman and Julia Gonski (SLAC), and Chang Sun (Caltech)

SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University

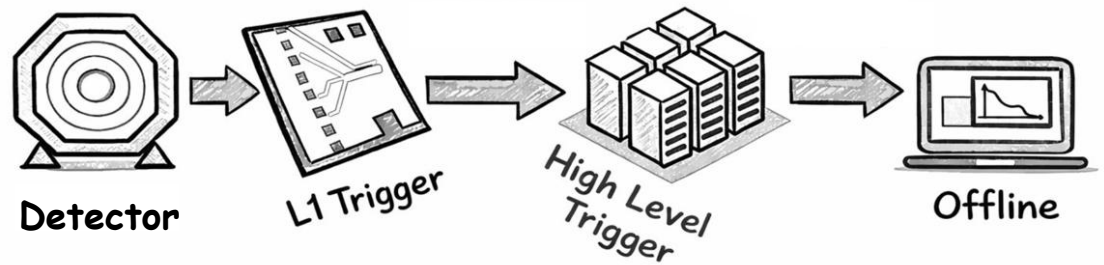
U.S. DEPARTMENT of ENERGY

Overview

- Modern detectors push readout systems to extreme data rates
- ML-based real-time algorithms enable data reduction at the source
- Customized hardware eFPGAs enable on-detector real-time ML
- Challenge: hardware-aware co-design
- Solution: open-source toolchain and agentic AI-assisted design
- Case study:
  - Design of ultra-small ML for real-time cluster counting algorithm on drift chamber
- Conclusion and Outlook
  - End-to-end agentic co-design

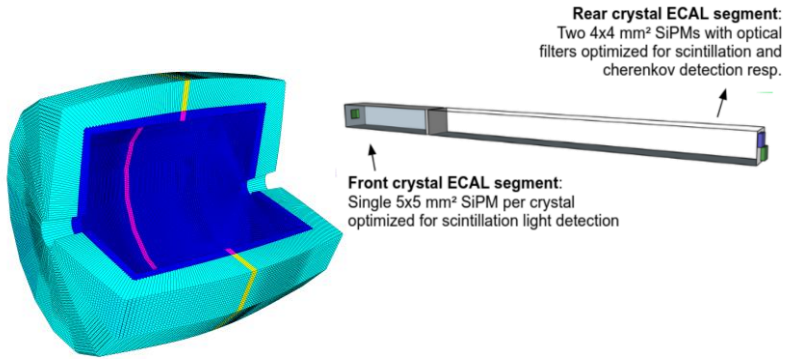
Readout Challenge For Modern Detector

Current: typical data chain for collider experiment at scale (e.g. LHC-ATLAS)

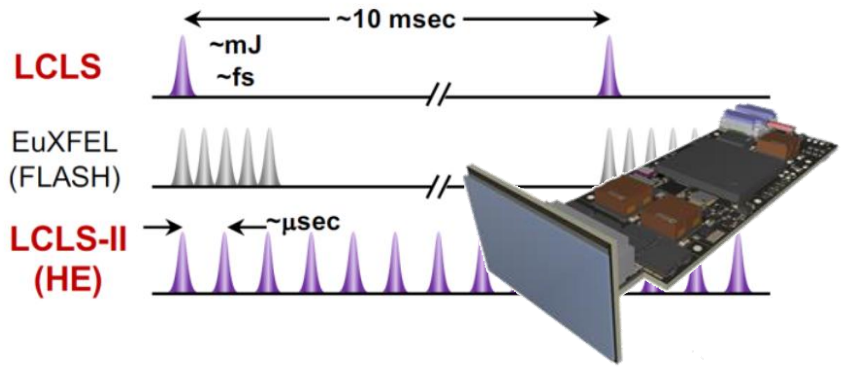


	Detector	L1 Trigger	High Level Trigger	Offline
Event Rate	40 MHz	100 KHz	3 KHz	Hz-KHz
Data Rate	O(100) TB/s	O(100) GB/s	O(10) GB/s	MB~GB/s

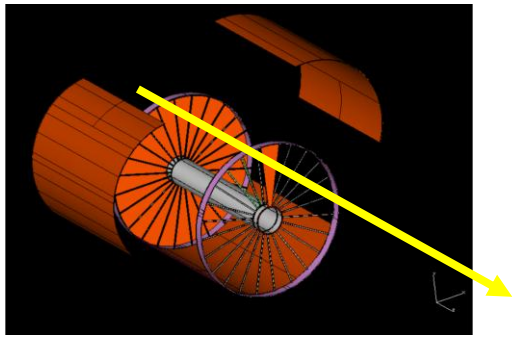
Future: challenge due to high granularity, high event rates, and strict material budget



Dual-readout calorimeter @ IDEA FCC *2502.21223*
High granularity up to **1.6M** readout channels
Waveform readout, sampling rate up to **GSa/s**



ePix/SparkPix@SLAC *Larry's talk*
Readout match the PW mode
High event rate up to **1MHz**



Drift chamber @ IDEA FCC *2502.21223*
High transparency – low materials
0.050 X_0 forward, **75% from electronics**

Gonski, J., Gupta, A., Jia, H., Kim, H., Rota, L., Ruckman, L., ... & Herbst, R. (2024). Embedded FPGA developments in 130 nm and 28 nm CMOS for machine learning in particle detector readout. Journal of Instrumentation, 19(08), P08023.
Yilmaz, D., Wu, L., Gonski, J., Rankin, D., & Herwig, C. (2025). Edge Machine Learning for Cluster Counting in Next-Generation Drift Chambers. arXiv preprint arXiv:2511.10540.
Wu, L., Liu, Q., Lucchini, M. T., Gonski, J., Campajola, M., & Moneta, S. (2026). Machine Learning Enables Real-Time Waveform Decomposition for Dual-Readout Calorimetry. arXiv preprint arXiv:2604.26090.

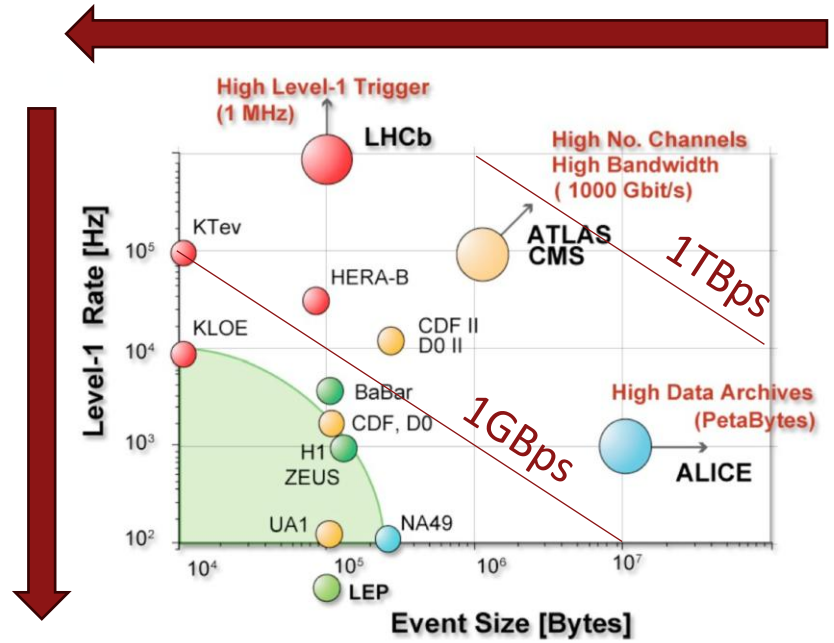
Data Reduction At Source: Real-Time ML

Max reduction when data processed BEFORE transmission

- Reduce event rate: trigger and streaming filter
- Reduce event size: data compression
- Real-time, limited resources → highly customized algorithm

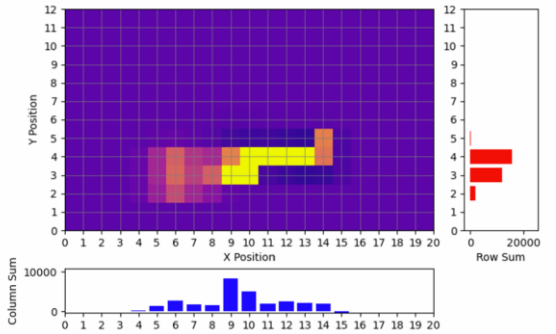
Machine Learning (ML) provides general solution

- Reusable: architecture/resources usage mainly rely on I/O dimensions
- Learn from data: smart information extraction

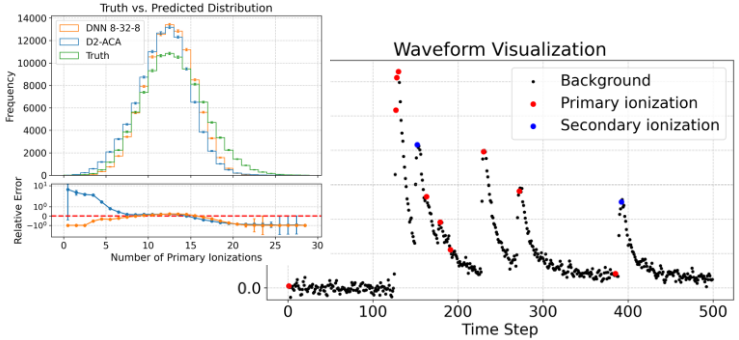


Applications of real-time ML algorithm for future HEP detector

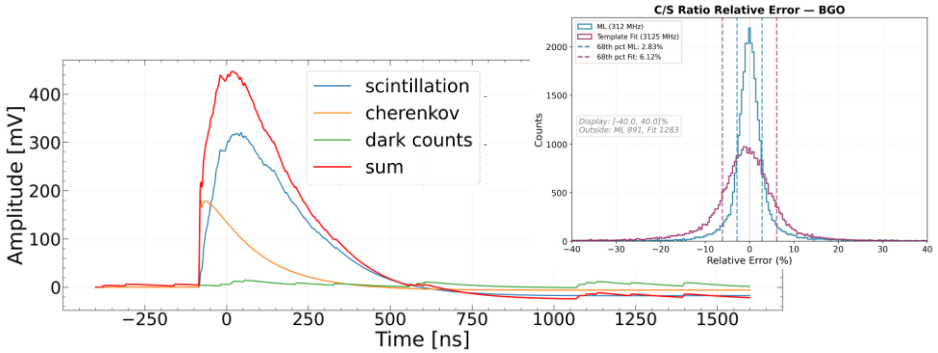
Smart-Pixel Signal Disc. with BDT



Drift Chamber dN/dX PID with NN



Dual-readout Calo. C/S separation with NN



Device for On-Detector Applications: eFPGA

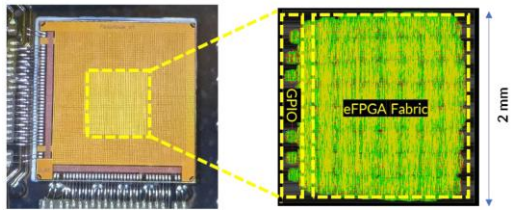
On-detector applications impose special requirements:

- Harsh environments → radiation tolerance, extreme temperatures, ...
- Tight resource and footprint constraints → low material budget and low power consumption
- Configurability and programmability → enable in-situ upgrades and flexible deployment

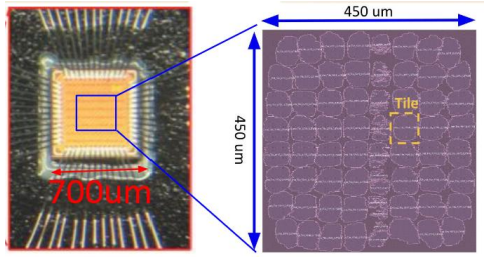
Embedded FPGA (eFPGA) provides a unique and powerful solution:

- Gate-level flexibility integrated into on-detector chips
- Enable customized algorithms optimized for available resources
- A promising platform for real-time on-detector ML

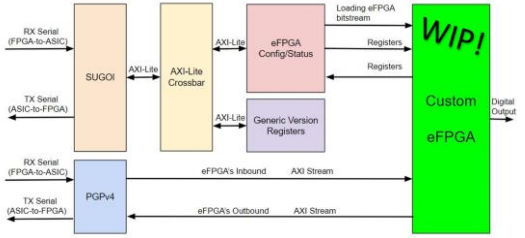
eFPGA R&D @ SLAC



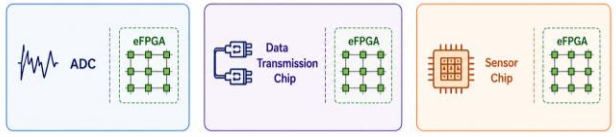
eFPGA prototype v1
130nm CMOS 5mmx5mm



eFPGA prototype v2
28nm CMOS 1mmx1mm



eFPGA prototype v3
...



eFPGA next generation
Real **e(**mbedded)**FPGA**

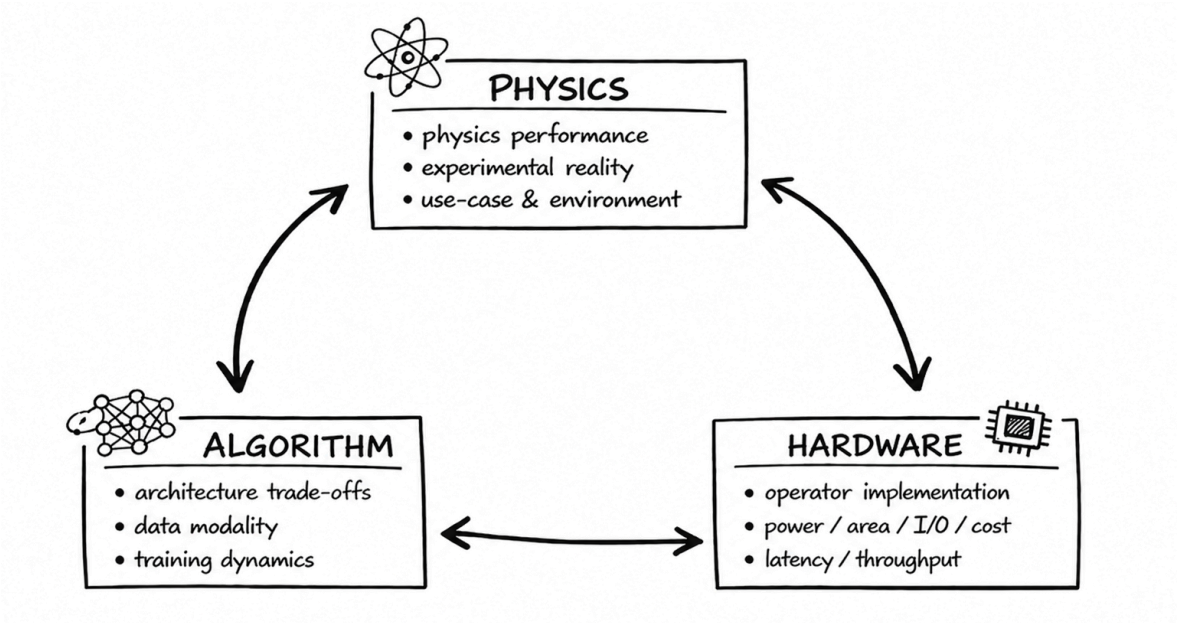
...

Challenge: Hardware-aware Co-design

Challenges in hardware-aware co-design for customized devices

- Missing out-of-the-box solution: ML-related applications often rely on **vendor-locked HLS tool**
- **Difficult fast prototyping**: iteration between cross-domain researchers and EE professionals becomes slow
- Time-consuming optimization: optimal performance must be explored **case by case, back and forth**

How can we close the loop and accelerate hardware–algorithm co-design?



Solution: Open-Source Toolchain and Agentic Workflow

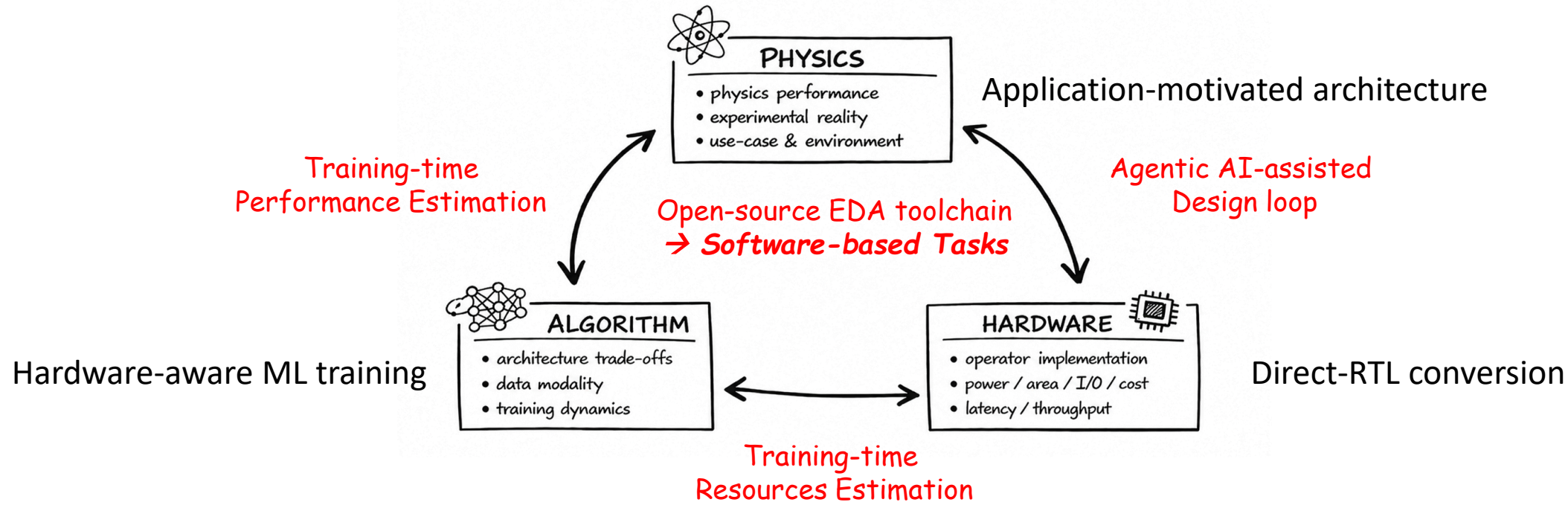
Open-source toolchains provide a transparent end-to-end co-design workflow for real-time application:

- From algorithm design, evaluation to RTL generation, validation and implementation
- Feedback of performance and HW utilization estimation at training-time

Agentic AI closes the full loop and fills the remaining gaps toward automated co-design

- Open-source EDA transform RTL design into software-based tasks
- LLM embedded EDA/EECS knowledge helps on automatic debugging, scripting and interfacing

Open-source EDA toolchain and agentic AI-assisted workflow



Does AI really understand EDA?

Large Language Models (LLMs) are trained on massive amounts of text data

- Manual, script, RTL syntax, design spec, **open-source EDA codes**: EDA domain knowledge included
- Observed by studies and benchmarks [Zang2025]: AI approaches EDA problems through a human-like way
- Example: debugging of full spec AXI-S wrapper using agentic AI framework (local open-source) [Liu2026]

Firmware Debug — Task Description and Result

Context

FPGA Verilog project under `sim/`. A streaming wrapper (`stream_wrapper.v`) chains a conv kernel and dense layer through a shift register. Verilator C++ binder, Python verify script, and golden data in `dataset/`.

The design in `sim/src/stream_wrapper.v` is pre-written. Set up a Verilator toolchain, build, and run the verify script. If anything fails, fix only `stream_wrapper.v` — do not touch other files.

Use the `common_env` skill to set up a shared env with the needed toolchain. Write `env_activate.sh` so future tasks can reuse it.

Todo

1. Set up toolchain via `common_env` skill, write `env_activate.sh`.
2. Build and verify. Fix `stream_wrapper.v` if needed.
3. Both verify modes must pass.

Expect

- `nopause.log` and `paused.log` both contain `PASSED: All`
- `notes.md` exists with a summary
- Only `sim/src/stream_wrapper.v` modified under `sim/` (if needed)

Output — RTL codes fixes

`-- deleted` `++ added` context **[Sn]**/**[Rn]** bug tags

```
localparam integer KERNEL_OUTPUT_BIT = 80;  [S1]
localparam integer KERNEL_OUTPUT_BIT = 90;
localparam integer II                = 19;  [R1]
localparam integer II                = 20;
shift_reg <= {shift_reg[SHIFT_REG_WIDTH:KERNEL_OUTPUT_BIT], kernel_out};  [S2] [R3]
shift_reg <= {kernel_out, shift_reg[SHIFT_REG_WIDTH-1:KERNEL_OUTPUT_BIT]};
if (cnt == 5'd21)  [R2]   out_valid <= (inp_valid && cnt == 5'd21);  [R2]
if (cnt == 5'd19)         out_valid <= (inp_valid && cnt == 5'd19);
```

Most bugs fixed in early stage: syntax error, width mismatch, AXIS signal spec, loop boundary so on

Flow control and logic error fixed with several iterations with simulation scripts

Firmware Debug — Iteration History

Work model	gemma4	**Review model**	gemma4	**Skills**	common_env
Total loops	4 (3 exhausted, 1 pass)	**Total iters**	118	**Wall time**	∼14 min

Loop 1 — 30 iters EXHAUSTED

1–7	Env setup	Auto-detect and reuse shared environment (warm start, 23 s)
8–9	Compile	Verilator catches `WIDTHEXPAND` on kernel output
10–11	Analyze	Read `.v` source files
12	Fix	Full rewrite: fixed S1 + S2 + R1 + R2 (all 4 params, 26 s)
13	Compile	New warning: `PROCASSINIT` (inline initializers)
14	Fix	Rewrite with `initial` blocks
15–30	Debug	Compile OK, verify fails. Debugging loop, hit iteration limit

Loop 2 — 30 iters EXHAUSTED

1–30 Debug Warm start + continued debugging loop (210 s). Hit iteration limit

Loop 3 — 30 iters EXHAUSTED

1–30 Debug Warm start + debugging (300 s). Reverted R1 fix in wrong direction (confused with R3)

Loop 4 — 28 iters PASS

1–28 Fix + verify Warm start. Found real R3 (shift direction). Both verify modes → **PASS** (190 s)

Case Study

Design of Ultra-Small ML for Drift Chamber on-detector Readout

Readout for Drift Chamber Detector in HEP-ex

Tracking detector with precise spatial resolution and very low material budget

Mature, cost-effective technology:

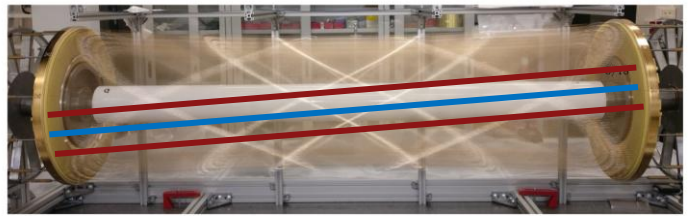
- Particle ionization in gas, followed by drift, amplification, and signal collection under an EM field

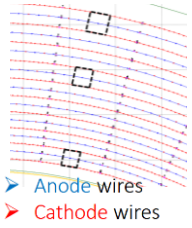
Readout design: precise timing measurement combined with pulse-shape analysis

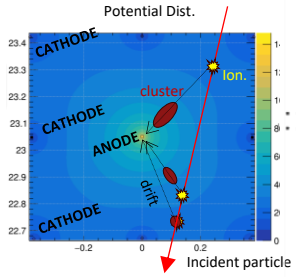
- **Leading-edge timing**: measures drift time and reconstructs the **incident particle position**
- **Pulse shape and peak structure**: encode ionization information, enabling **particle-identification (P-ID)**

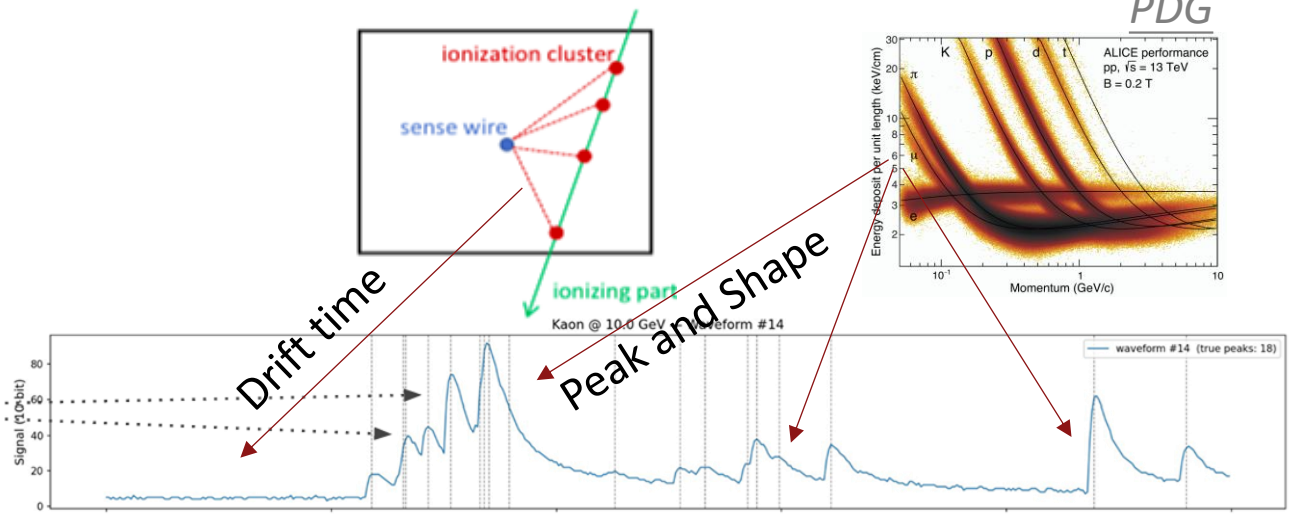
Full waveform readout from all channels generates enormous data rates

- **Real-time waveform analysis is motivated**









PDG

Marco Chiappini

Caputo, Claudio, et al. "Particle identification with the cluster counting technique for the IDEA drift chamber." Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 1048 (2023): 167969.
Tian, Zhe-Fei, et al. "Cluster counting algorithm for the CEPC drift chamber using LSTM and DGCNN: Z.-F. Tian et al." Nuclear Science and Techniques 36.7 (2025): 113.
Yilmaz, D., Wu, L., Gonski, J., Rankin, D., & Herwig, C. (2025). Edge Machine Learning for Cluster Counting in Next-Generation Drift Chambers. arXiv preprint arXiv:2511.10540. .

ML-based Real-Time Cluster Counting Algorithm

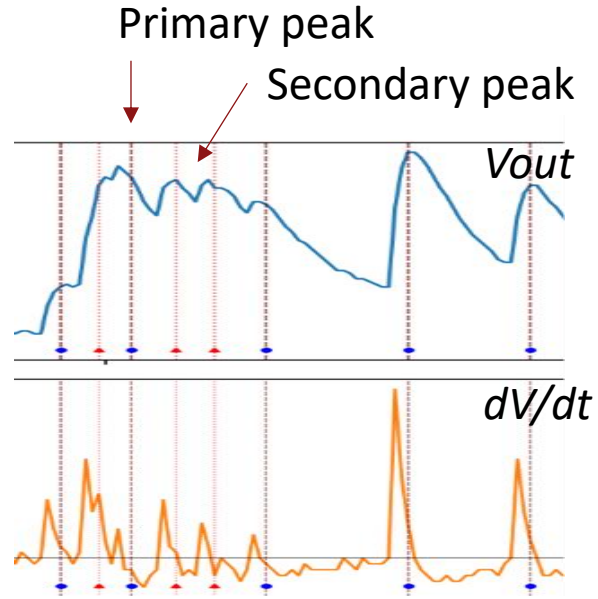
Principles of P-ID: ionization power ~ charge or number of clusters (peaks) created

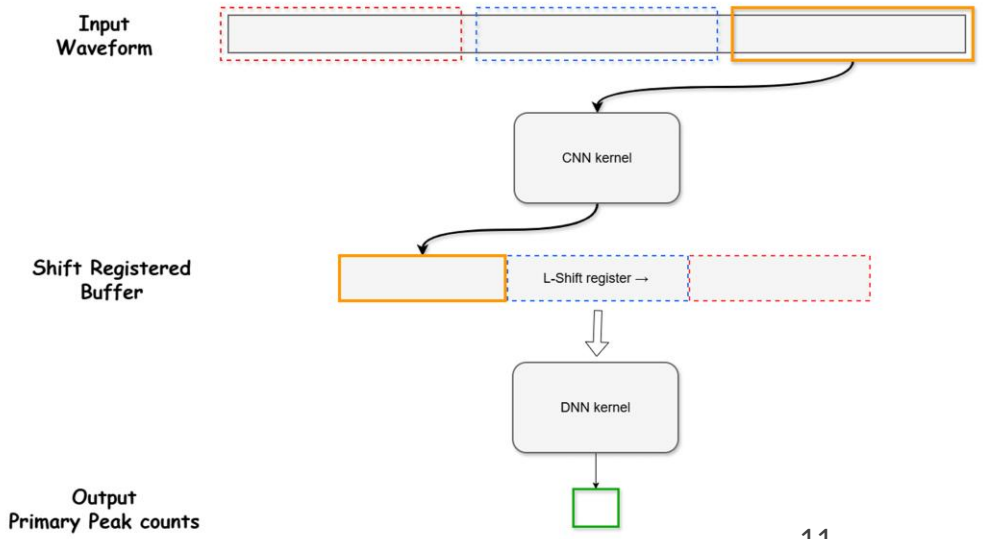
- Derivative based method gives large uncertainty: secondary peaks
- ML method [Tian2025] captures the non-linear peak-peak correlation
- Online ML method [Yil2025] runs small-ML on FPGA (~30K LUT+DSPs)

Ultra-small ML models (<10K LUTs) required for on-detector application

- Time translation: streaming CNN (filter)
- Peak-peak correlation: DNN (non-linearity)
- Hardware-friendly buffer: shift register

General ML model for waveform: learn any 1-feature from data





Sun, C., Que, Z., Aarrestad, T., Loncar, V., Ngadiuba, J., Luk, W., & Spiropulu, M. HGQ: High granularity quantization for real-time neural networks on FPGAs. In Proceedings of the 2026 ACM/SIGDA International Symposium on FPGA (pp. 79-91).
Sun, C., Que, Z., Zadeh, B., Liu, Q., Alvarez, K., Luk, W., & Spiropulu, M. (2026). HGQ-LUT: Fast LUT-aware training and efficient architectures for DNN inference. In Proceedings of the 2026 IEEE 34th International Symposium on FCCM.

Hardware-aware ML training

High Granularity Quantization (HGQ) [Chang2026] tool enables hardware-aware training and optimization of ML

- Differentiable Quantization: **quantization become trainable during training**
- Direct-RTL backend and EBOPs/LUTs utilization estimation: **hardware utilization become accessible during training**
- Efficient PID and pareto front scan: **training-time hardware-aware optimization**

Algo: Differentiable Quantization

1: Let f be the trainable parameter for quantization bits.

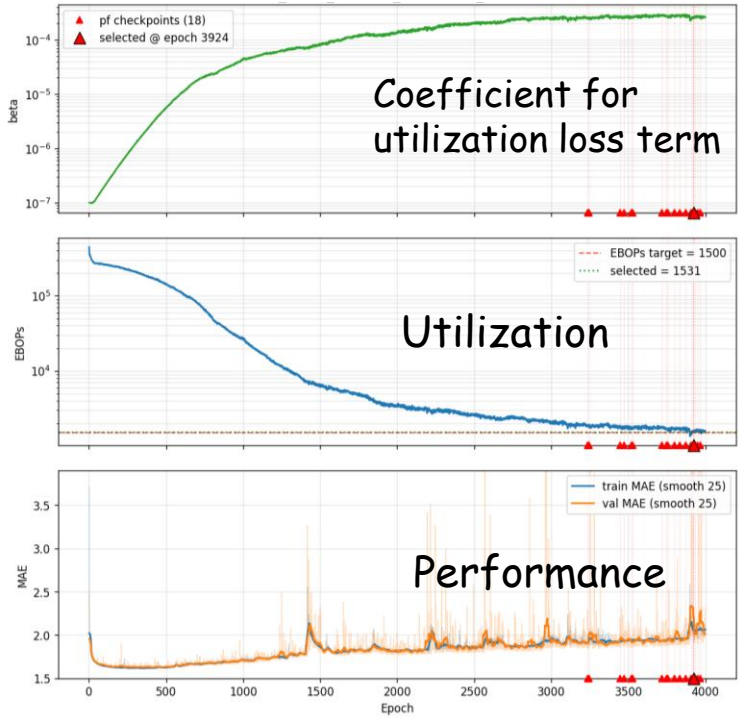
$$\begin{aligned}\delta_f &\equiv x - f^q(x) \\ &= x - \left[x \cdot 2^f\right] \cdot 2^{-f}\end{aligned}$$

2: The gradient with respect to f is given by

$$\frac{\partial \mathcal{L}}{\partial f} = \frac{\partial \mathcal{L}}{\partial \delta_f} \cdot \frac{\partial \delta_f}{\partial f}$$

3: The first term is obtained from back-propagation.
4: The second term is approximated a surrogate gradient,

$$\frac{\partial \delta_f}{\partial f} \leftarrow -\log 2 \cdot \delta_f$$

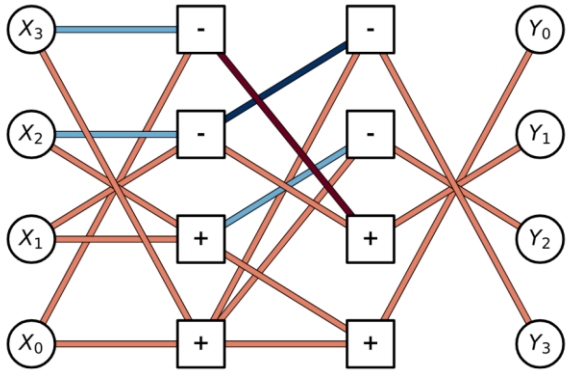


Sun, C., Que, Z., Loncar, V., Luk, W., & Spiropulu, M. (2026). da4ml: Distributed arithmetic for real-time neural networks on fpgas. ACM Transactions on Reconfigurable Technology and Systems, 19(1), 1-27.
Liu, Q., & Gonski, J. (2026). SciFi: A Safe, Lightweight, User-Friendly, and Fully Autonomous Agentic AI Workflow for Scientific Applications. arXiv preprint arXiv:2604.13180.

RTL implementation: Open-source tool and Agentic AI

Direct-RTL conversion of ML kernels using the DA4ML [Chang2026]

- HLS-free, bit-exact mapping of NN to RTL
- LUT-only and combinational implementation (optional pipeline staging)
- Fully open-source tool including RTL simulation (via Verilator)



System integration using agentic AI-assisted method

- Including interface integration, AXI-S handling and ML kernels scheduling
- Open-source agentic framework [Liu2026] utilized
- Commercial solution widely available (Claude Code, so on)
- Specification (RTL skeleton) and validation (loop)
- Generated codes passing validation

Output (agent)

```
...
localparam
  KERNEL_OUTPUT_BIT = 90;
kernel_wrapper u_kernel(...);
 localparam SHIFT_REG_WIDTH
   = 20 * KERNEL_OUTPUT_BIT;
reg [W-1:0] shift_reg;
 always @(posedge clk)
   if (inp_valid) begin
     shift_reg[count*90 +:  90]
       <= kernel_out;
     if (count==19) begin
       out_valid<=1; count<=0;
     end else begin
       count<=count+1;
       out_valid<=0;
   end end
dense_wrapper u_dense(...);
 assign model_out={18'b0,dense_out};
...
```

Reference (human design)

```
...
localparam
  KERNEL_OUTPUT_BIT = 90;
 localparam II = 20;
 localparam SHIFT_REG_WIDTH
   = II * KERNEL_OUTPUT_BIT;
reg [W-1:0] shift_reg;
 always @(posedge clk)
   if (inp_valid)
     shift_reg <= {kernel_out,
       shift_reg[W-1:90]};
 reg [4:0] cnt = 0;
 always @(posedge clk)
   if (inp_valid)
     cnt <= (cnt==19) ?  0 :  cnt+1;
 always @(posedge clk)
   out_valid <=
     (inp_valid && cnt==19);
 assign model_out =
   {{18{1'b0}}, dense_out};
...
```

Algorithm Evaluation

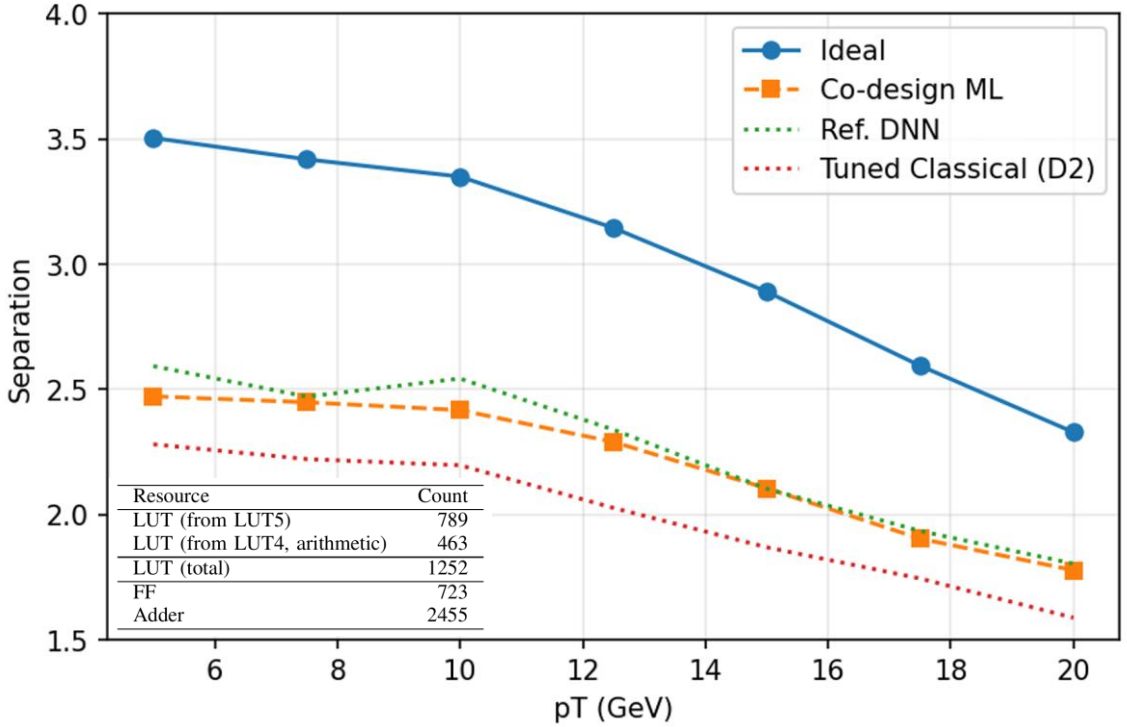
Kaon/pion separation metric as performance benchmark

- Precise cluster counting → close to ideal separation

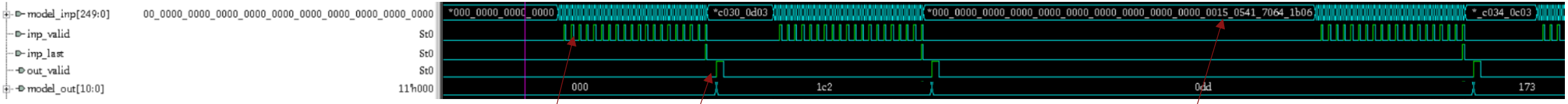
Ultra-small ML achieved with ~1K LUTs

- Better performance than derivative-based (D2) peak-finding
- Comparable performance as the reference ML model (HLS)
- **20x smaller in the utilization** and no DSP inference

RTL validated in the system with AXI-S interface (simulation)



Resource	Count
LUT (from LUT5)	789
LUT (from LUT4, arithmetic)	463
LUT (total)	1252
FF	723
Adder	2455



Input Streaming
25 Sa/cycle

Output Burst
20 cycle/event

Emulation of Input Stall

Summary

Large granularity and high event rates make data movement a key bottleneck for modern HEP detector readout.

On-detector real-time processing reduces complex data directly at the source.

Ultra-small on-detector devices such as eFPGAs provide fully customized, low-power solutions.

Open-source toolchains and agentic workflows boost co-design for specialized algorithms on customized devices.

- Ultra-small ML designed for clustering/counting algorithms in drift chamber detectors
- **Reference design for general waveform ML under limited-resource constraints**

Looking forward:

- **Agentic EDA is becoming increasingly practical**, with rapid progress from both vendors and open-source communities
- *Purely AI-generated EDA design at the next RT conference?*

Support Info

Work supported by the U.S. Department of Energy under contract number DE-AC02-76SF00515

Design for AI,
AI for Design

Thanks!