Intelligence on Chip

-- On-chip FastML for FCC applications

Speaker: Qibin LIU (SLAC)

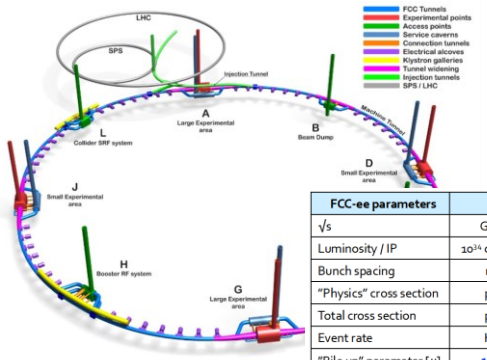
US-FCC Workshop Sep. 10, 2026

Overview

- Modern detectors push readout systems to extreme data rates
- ML-based real-time algorithms enable data reduction at the source
- **eFPGA provides a promising on-detector platform for real-time ML**
  - High-Level Overview of eFPGA Design and Implementation
- Application for FCC-ee detectors:
  - **Cluster Counting** for Drift Chamber on-detector Readout
  - **Waveform decomposition** for Dual Readout Calorimeter
- Conclusion and Outlook:
  - **Intelligence on Chip**: Technology is ready, more applications to come
  - **AI × AI**: Boost co-design loop for on-chip **AI** with agentic **AI** workflow

Readout Challenge For Detector at FCC-ee

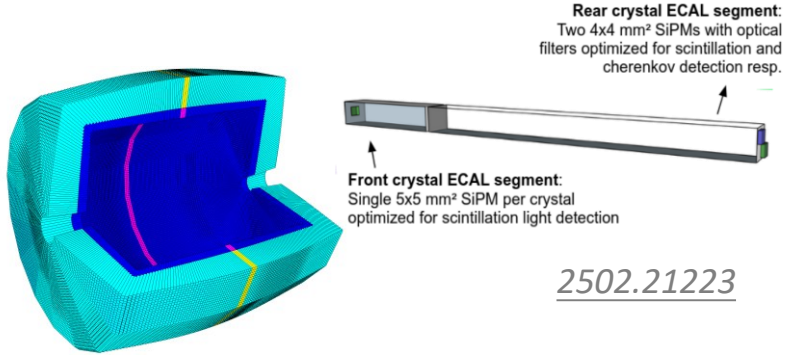
High event rates, high granularity, and strict material budget



Mogens Dam (NBI)

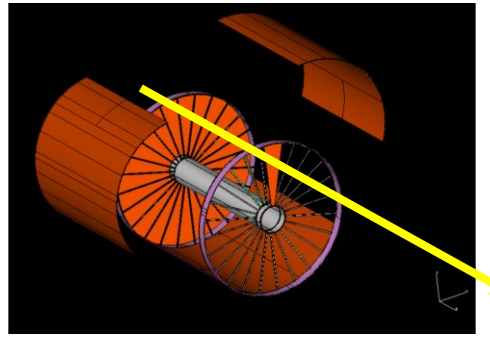
FCC-ee parameters		Z	W⁺W⁻	ZH	ttbar
√s	GeV	91.2	160	240	350-365
Luminosity / IP	10^{34} cm^{-2} s^{-1}	**140**	20	7.5	1.5
Bunch spacing	ns	25	160	680	5000
"Physics" cross section	pb	35,000	10	0.2	0.5
Total cross section	pb	70,000	30	10	8
Event rate	Hz	**100,000**	6	0.5	0.1
"Pile up" parameter [μ]	10^{-6}	2,500	1	1	1

FCC-ee luminosity up to **10^36 cm^-2 s^-1**
Physics events at **200 kHz**
Trigger-less/streaming readout under discussion



2502.21223

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



2502.21223

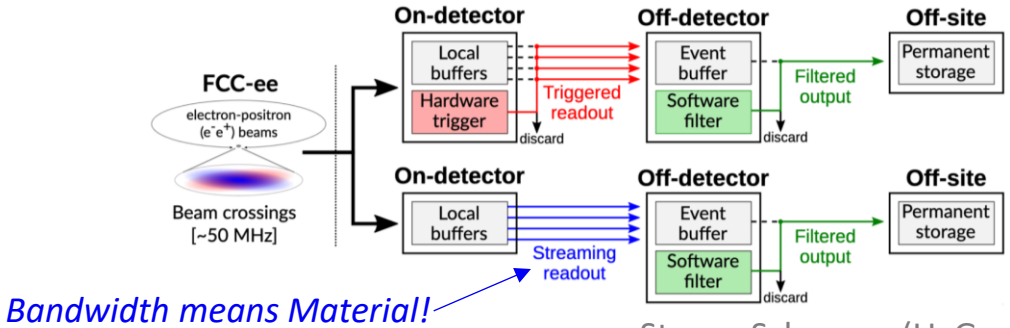
Forward direction

Drift chamber @ IDEA FCC
High transparency – low materials
5.0% X_0 forward, **75% from electronics**

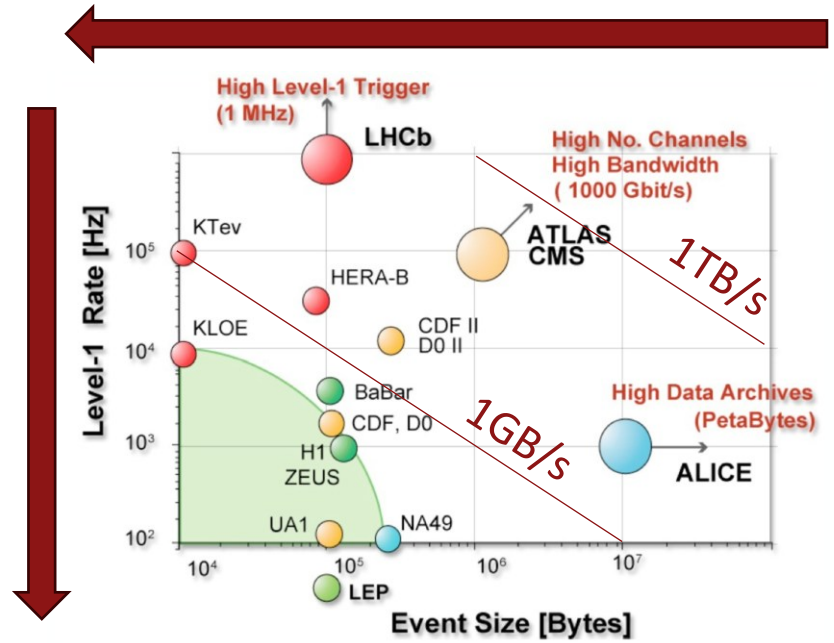
Data Reduction At Source: Real-Time ML

Max reduction when data processed BEFORE transmission

- Reduce event rate: trigger or streaming w/ filter
- Reduce event size: data compression for streaming w/o filter
- Real-time algo needed, limited resources → highly customized



Steven Schramm (U. Geneva)



Assume 200 KiB event size

	Raw input		Trigger →	Filter input		Filter →	Storage input	
FCC-ee	50 MHz	10 TB/s		0.5 MHz	0.1 TB/s		200 kHz	40 GB/s
	Streaming, with filter	50 MHz	10 TB/s				500 kHz	100 GB/s
				Streaming, no filter			50 MHz	10 TB/s

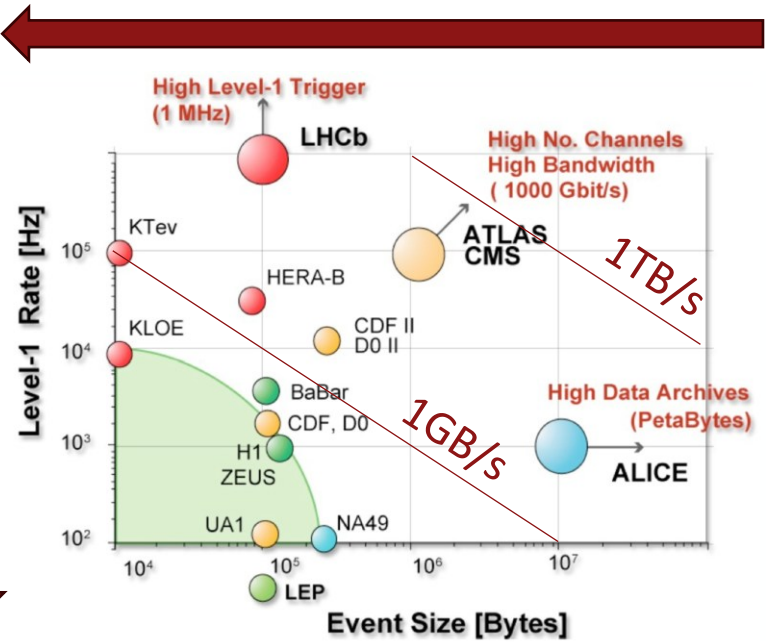
Steven Schramm (U. Geneva), Modified.

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 or streaming w/ filter
- Reduce event size: data compression for streaming w/o filter
- Real-time algo needed, limited resources → highly customized

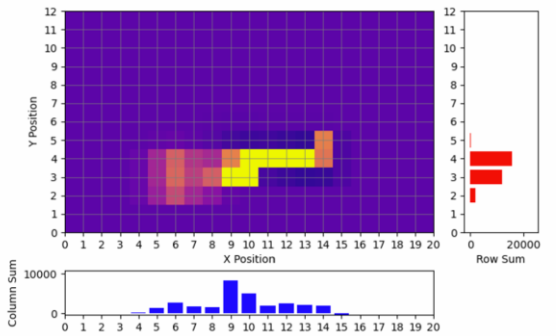
Machine Learning (ML) provides general solution
- Reusable architecture: swappable weights for different tasks
- General smart readout: learn any algorithm from training dataset



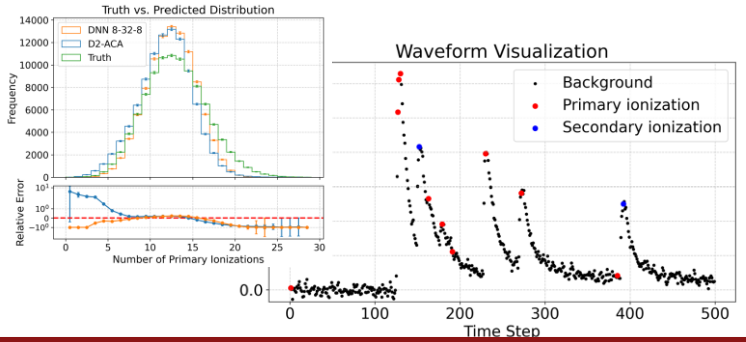
Applications of real-time ML algorithm for future HEP detector

Today's focus

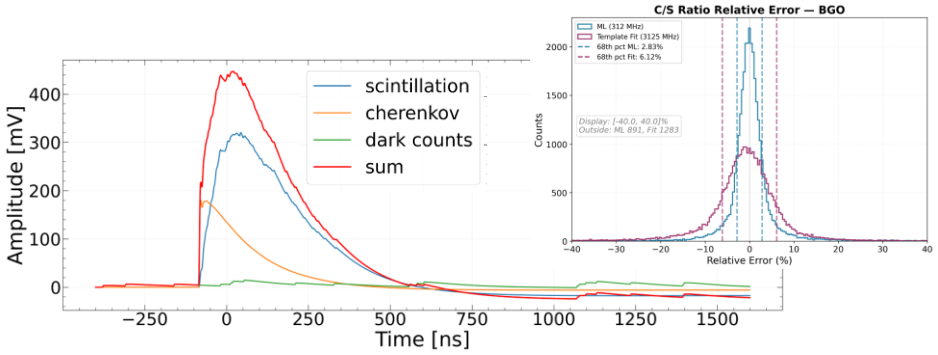
Smart-Pixel Signal Disc. with BDT



Drift Chamber dN/dX PID with NN



Dual-readout Calo. C/S separation with NN



Chip for On-Detector Real-Time ML: eFPGA

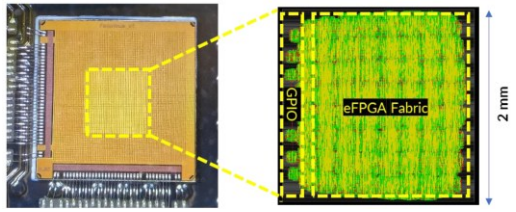
On-detector applications impose special requirements:

- 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

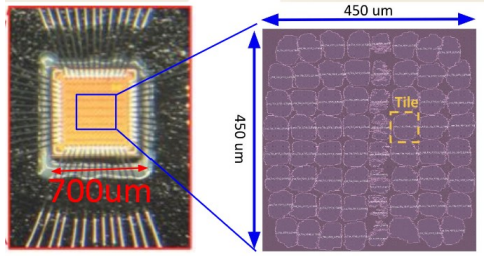
Embedded FPGA (eFPGA) provides a unique and powerful solution:

- Gate-level flexibility integrated into on-detector chips
- Enable customized algorithms optimized for available resources
- A promising platform for real-time on-detector ML
- Future development: embedded intelligence in any digital chip (*smart-X*)

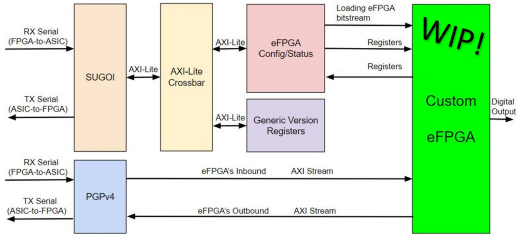
eFPGA R&D @ SLAC



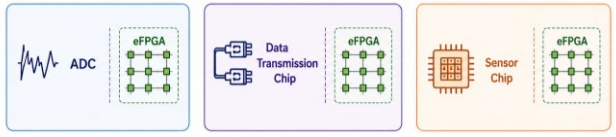
eFPGA prototype v1
130 nm CMOS 5 mm x 5 mm



eFPGA prototype v2
28 nm CMOS 1 mm x 1 mm



eFPGA prototype v3
(work towards tape-out)



Real **e**mbedded **FPGA**
(future goal)

...

CPAD-2024-eFPGA-Ruckman, ML4FE_2025_eFPGA_Ruckman

High-Level Overview of eFPGA Design and Implementation

The Overall Archtecture

Opensource solution!

proprietary and controlled workflow

ASIC RTL + Others

openFPGA

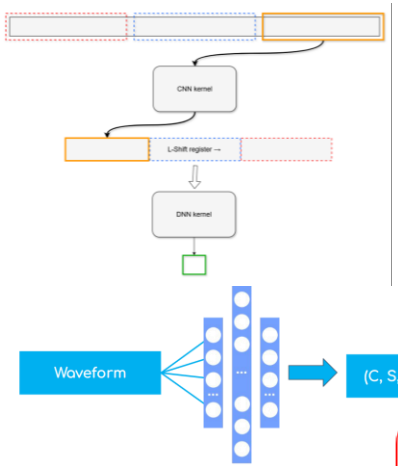
openFPGA Archtecture
(e.g. how many LUTs, how connect, I/O)

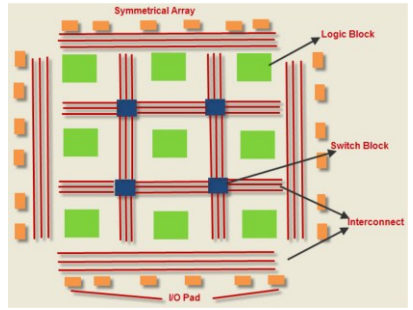
RTL compiler (VPR, Yosys)
for customized FPGA arch

Customized FPGA Chip (eFPGA)

ML/Application → Application RTL

Firmware



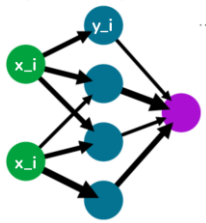


Look Up Table Implementation (P&R, Clocking, I/O)

Look Up Table Configuration

A simple neural network

input layer hidden layer output layer



$Y=\theta(WX+B)$

θ Activation: Element-wise function

W weight: Multiply-Add (dot product)

B bias: Addition

```
module mac2u(input  [1:0] W, X,
             input  [2:0] B,
             output [4:0] Y);
  wire w0=W[0], w1=W[1], x0=X[0], x1=X[1];
  wire t0=w0&x0, t1=w1&x0, t2=w0&x1, t3=w1&x1;
  wire p0=t0, p1=t1^t2, c1=t1&t2, p2=t3^c1, p3=t3&c1;
  assign S = {1'x0,p3,p2,p1,p0} + {2'x00,B};
endmodule
```

E.g. C = A & B

Input A	Input B	Output C
0	0	0
0	1	0
1	0	0
1	1	1

Application

Cluster Counting for Drift Chamber on-detector Readout

Collaboration work w/ Liangyu Wu and Deniz Yilmaz (Stanford), Larry Ruckman and Julia Gonski (SLAC), Chang Sun (Caltech)

More from more experts' view: workshop slides Wed, Thu

Readout for Drift Chamber Detector @ IDEA FCC-ee

Tracking detector with precise spatial resolution and very low material budget

Mature, cost-effective technology:

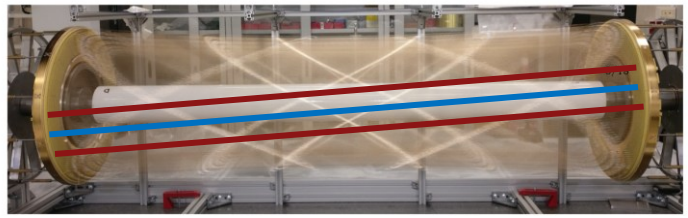
- Particle ionization in gas, followed by drift, amplification, and signal collection under an EM field

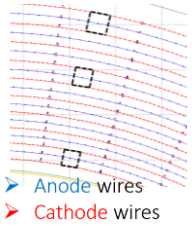
Readout design: precise timing measurement combined with pulse-shape analysis

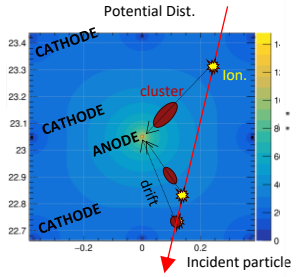
- **Leading-edge timing**: measures drift time and reconstructs the **incident particle position**
- **Pulse shape and peak structure**: encode ionization information, enabling **particle-identification (P-ID)**

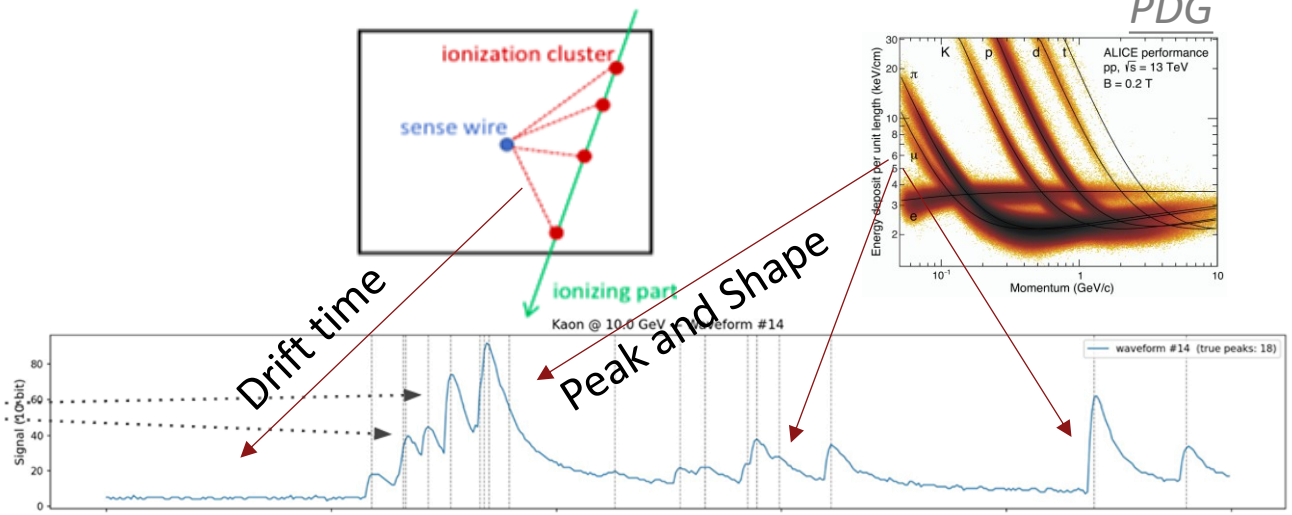
Full waveform readout from all channels generates enormous data rates

- **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 primary clusters (peaks)

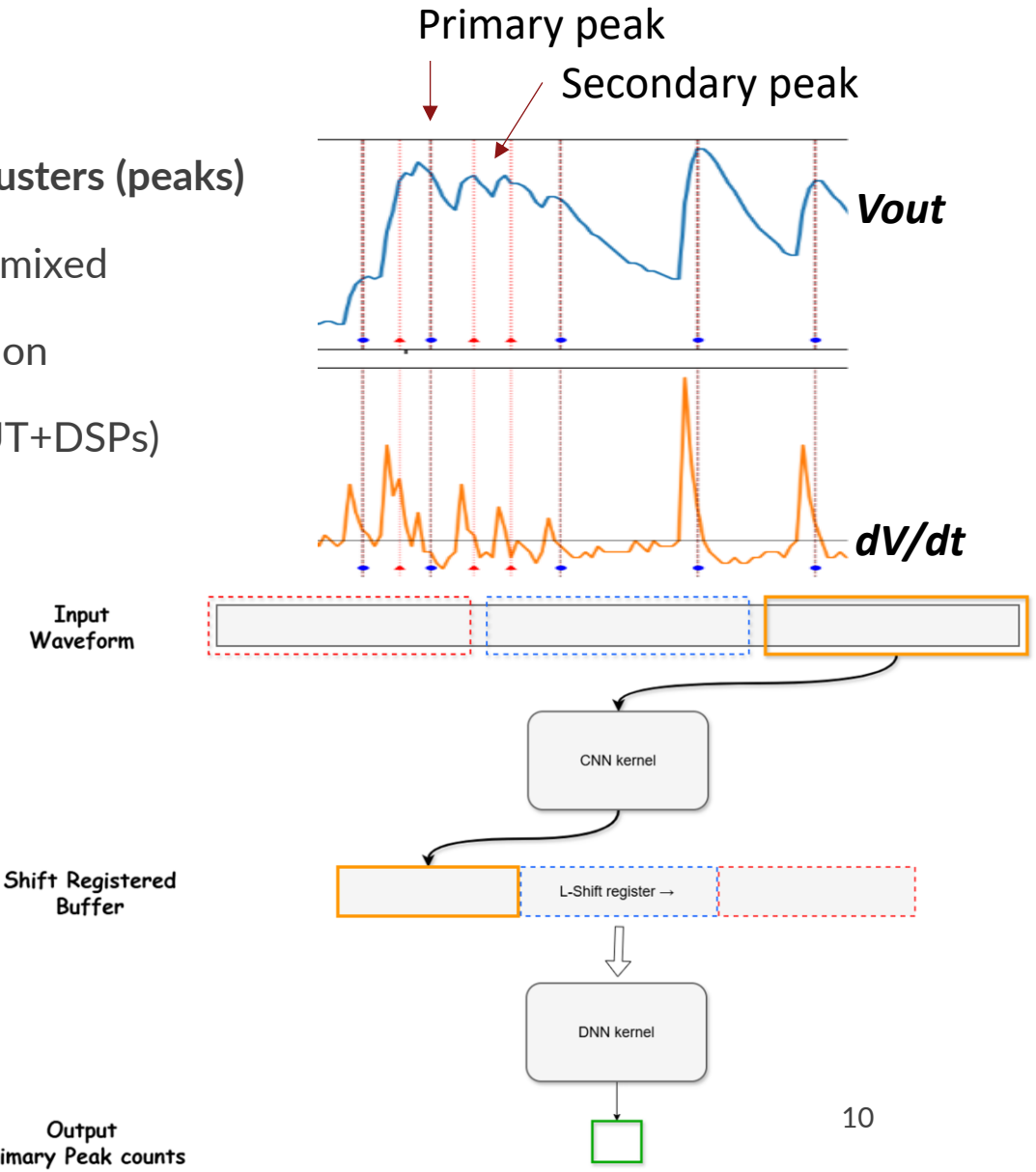
- Derivative based method gives large uncertainty: secondary peaks mixed
- ML method [Tian2025] captures the non-linear peak-peak correlation
- Online ML method [Yilmaz2025] runs small-ML on FPGA (~30K LUT+DSPs)

Ultra-small ML models (<10K LUTs) designed for on-detector chip

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

General architecture for waveform: learn any 1-feature from data

- Extendable to other application like de-noise, integral, timing.

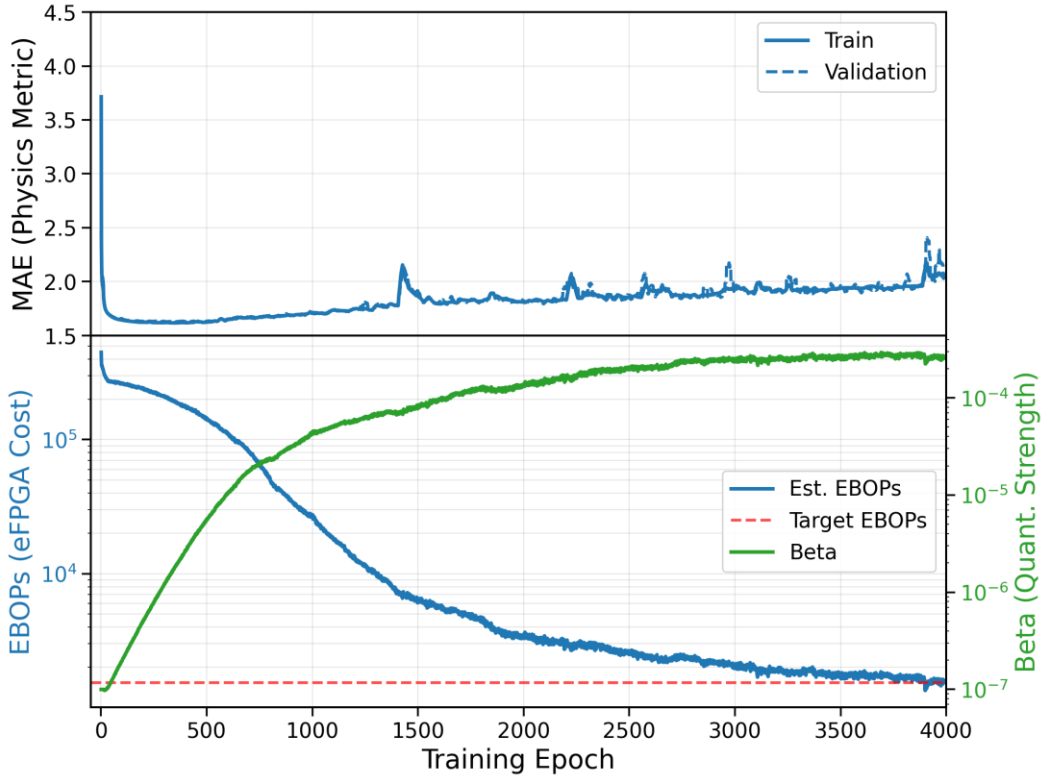


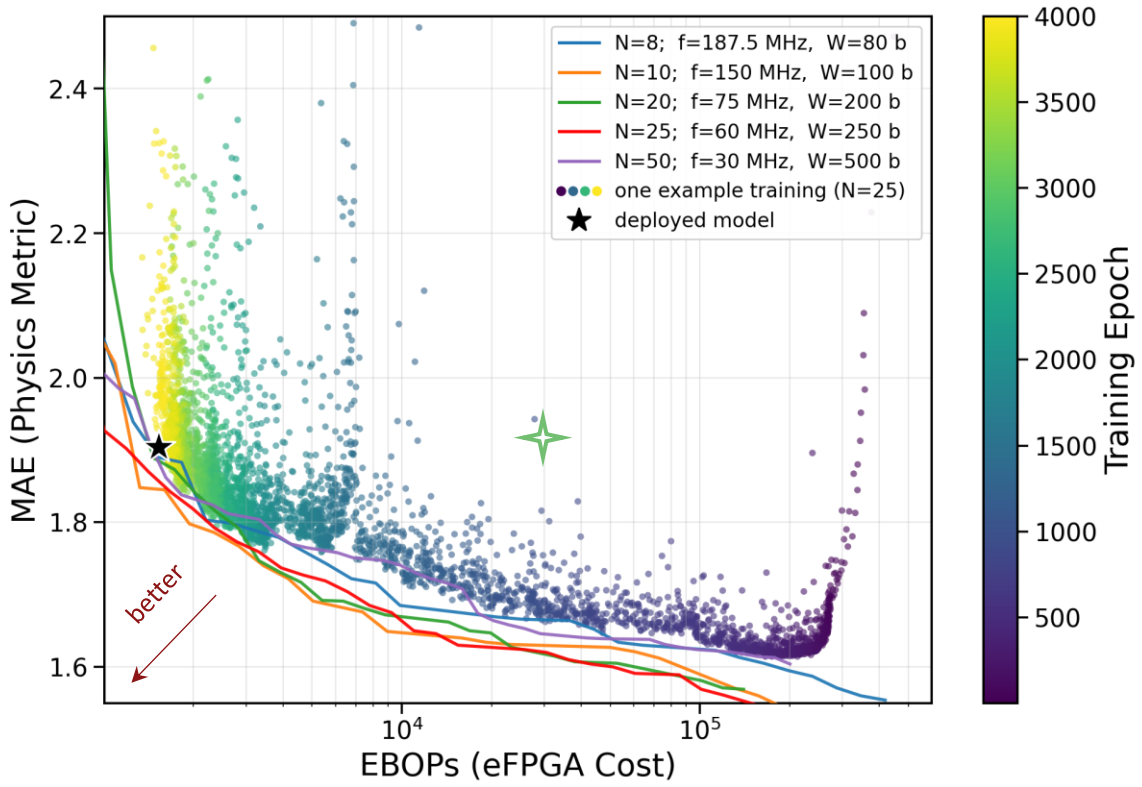
Sun, C., Que, Z., Aarrestad, T., Loncar, V., Ngadiuba, J., Luk, W., & Spiropulu, M. HGQ: High granularity quantization for real-time neural networks on FPGAs. In Proceedings of the 2026 ACM/SIGDA International Symposium on FPGA (pp. 79-91).
Sun, C., Que, Z., Zadeh, B., **Liu, Q.**, Alvarez, K., Luk, 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.
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.

Hardware-aware training and impl. for on-chip AI

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

- Dynamic and trainable quantization and EBOPs/LUTs estimation during training
- PID-training and pareto front scan: hardware-aware optimization for on-chip AI
- Direct-RTL (DA4ML) backend and agentic-based system integration: Efficient and customized implementation

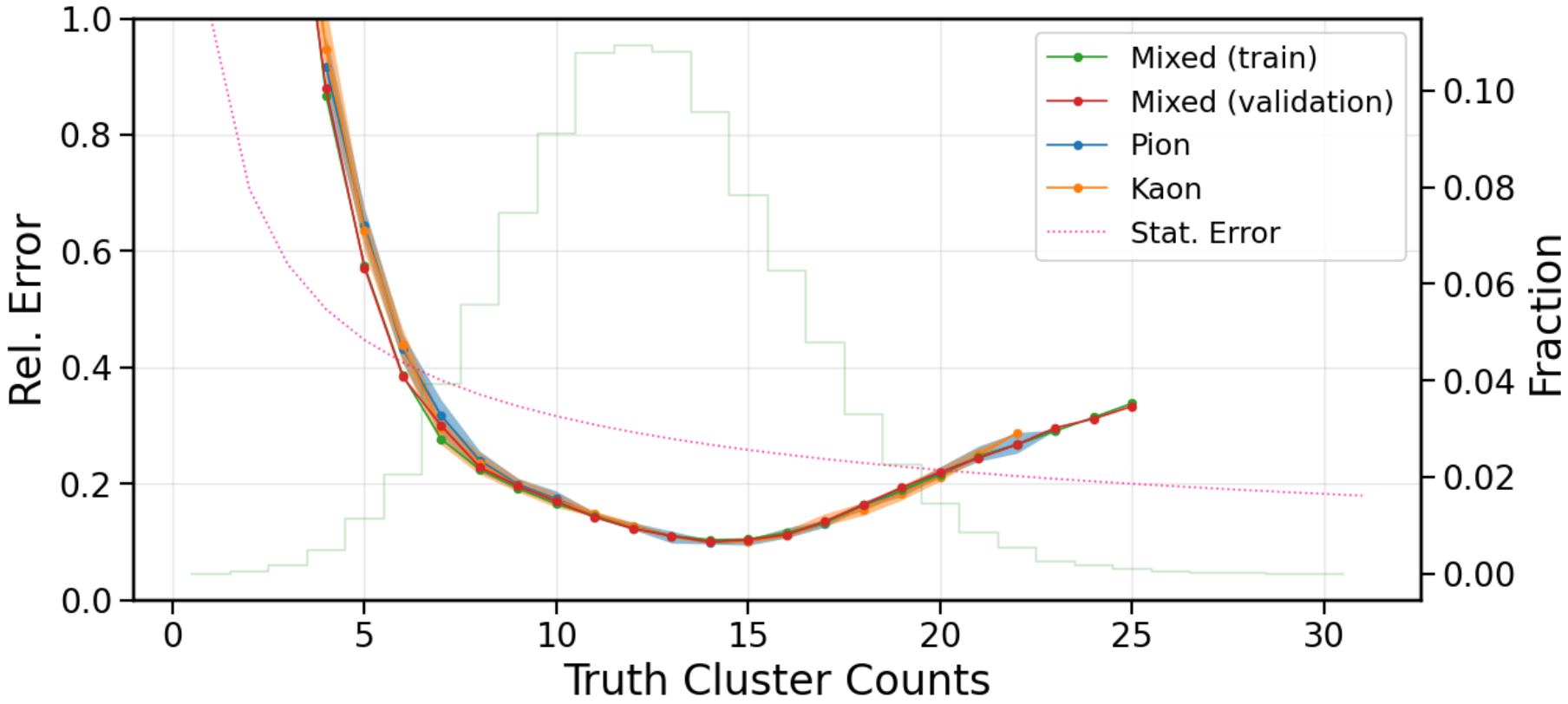




Physics Performance and Resources Utilization

Cluster counting evaluated on different particle (kaon/pion) and momentum across 5-20 GeV

- Relative error as low as 10%, reach the level of statical error
- No bias w.r.t. particle type and momentum: critical for particle ID application



Separation metric follows the same definition as workshop slides Wed

Physics Performance and **Resources Utilization**

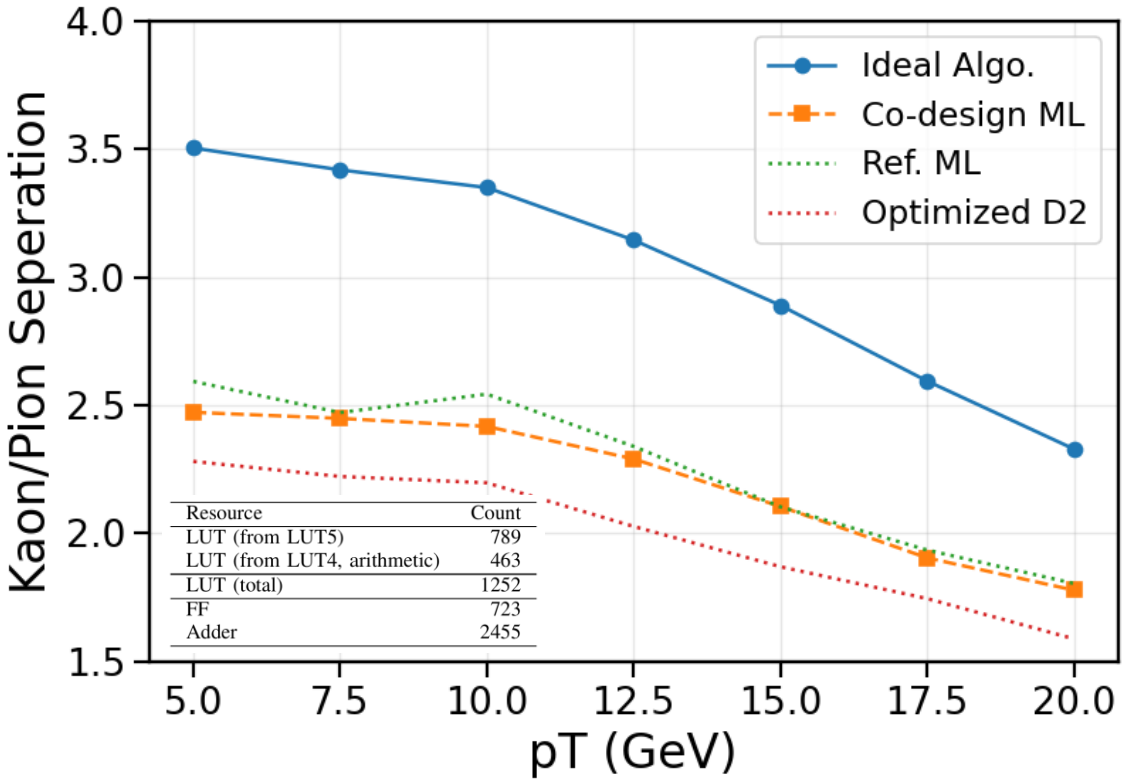
Kaon/pion separation evaluated as physics benchmark

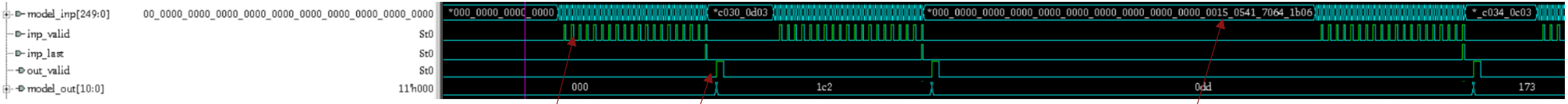
- Precise cluster counting → close to ideal separation

Ultra-small ML achieved with ~1K LUTs

- Better performance than derivative-based (D2) peak-finding
- Comparable performance to the reference ML model (HLS)
- **Tiny LUTs utilization** and no DSP (20x smaller than ref.)

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





Input Streaming
25 Sa/cycle

Output Burst
20 cycles/event

Emulation of Input Stall

Application

Waveform decomposition for Dual Readout Calorimeter

Collaboration work w/ Liangyu Wu (Stanford), Julia Gonski (SLAC),
and experts of DRO calorimeter: Marco Toliman Lucchini, Marcello Campajola and Stefano Moneta (INFN)

Dual-readout Calorimeter at FCC-ee

Different particles exhibit different electromagnetic-to-hadronic (EM/Had) shower fractions in the calorimeter

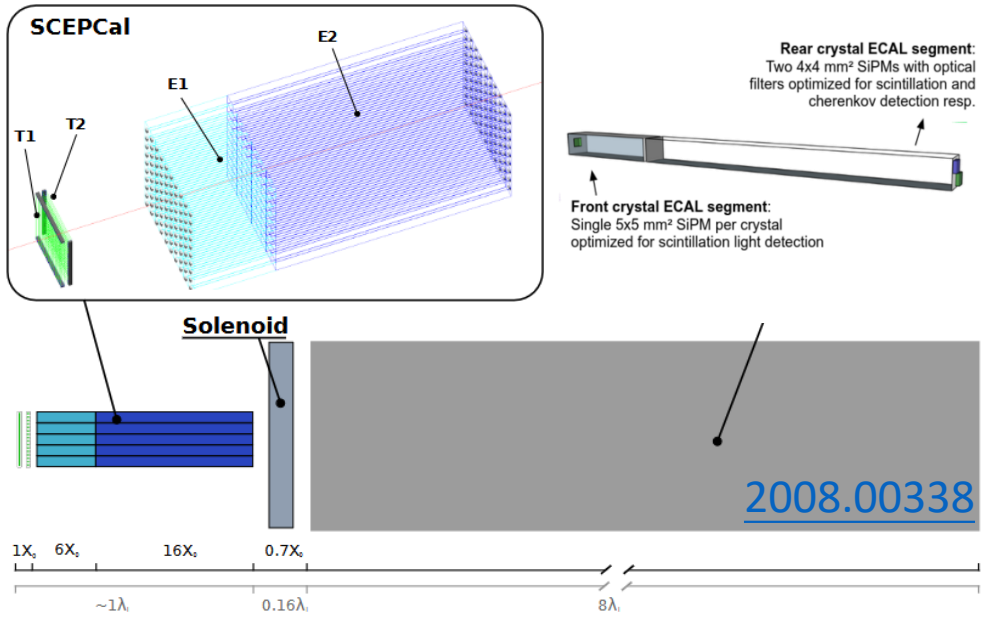
- Complicates energy calibration and significantly degrades the scale performance

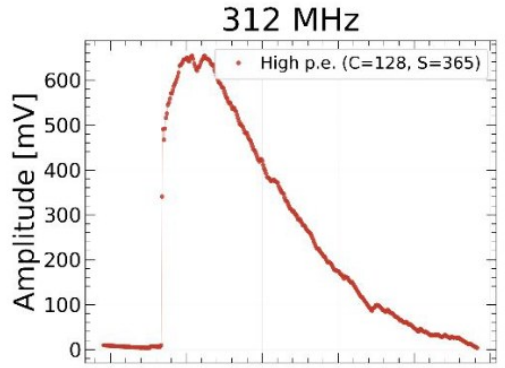
Dual-readout calorimeter measures both scintillation and Cherenkov signals

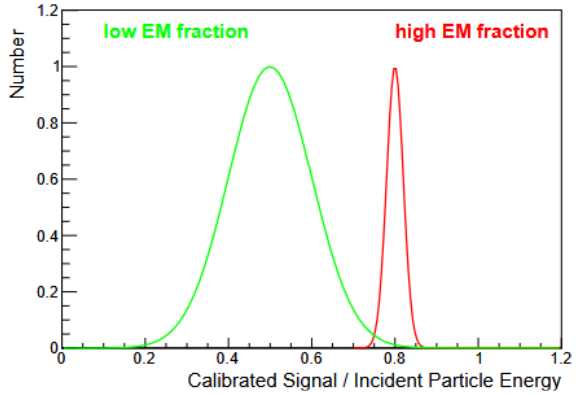
- **Cherenkov process is primarily sensitive to relativistic particles and provides an additional measurement**

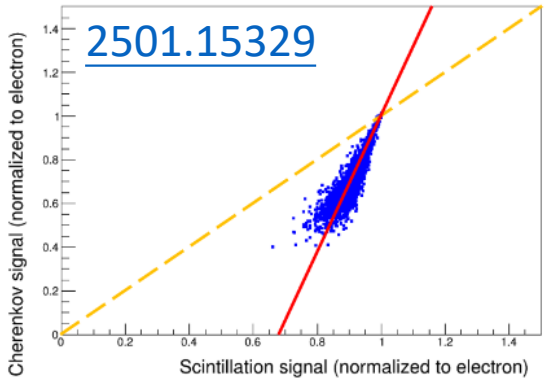
Combining and calibrating the two measurements, energy resolution can be improved

Challenge due to overlapping Cherenkov and scintillation: **waveform analysis necessary to decompose C/S**

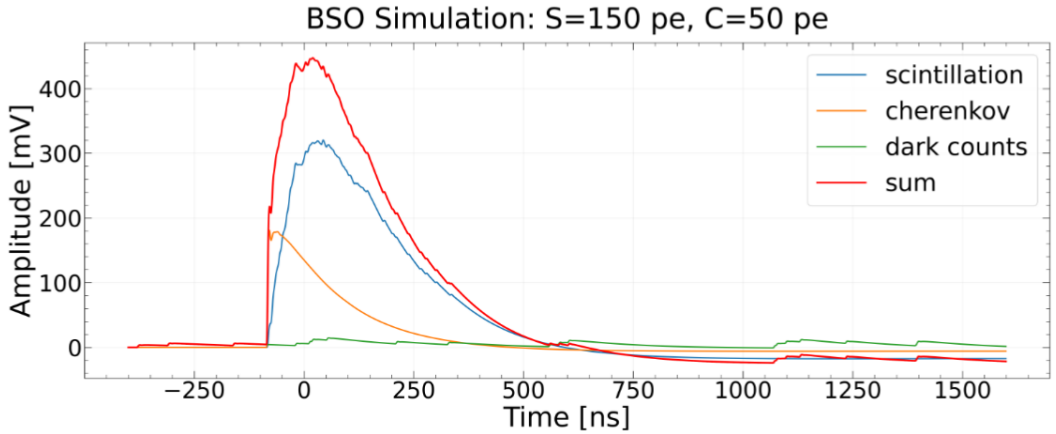






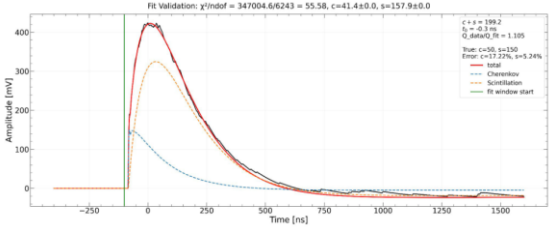


C/S Decomposition: Template Fitting v.s. ML-based



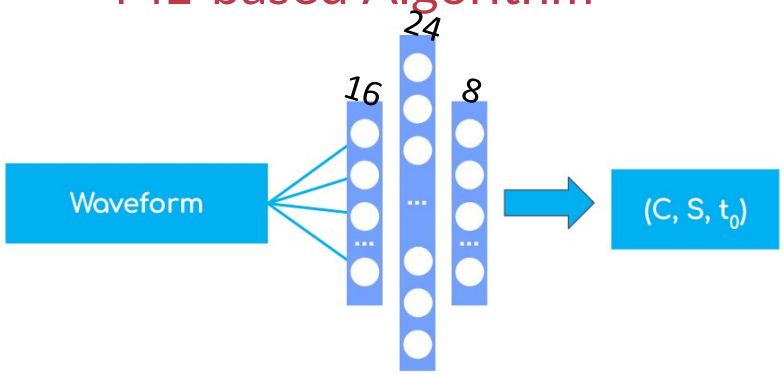
Template Fitting

$$\mathrm{WF(t)} = c \cdot C_{\mathrm{template}}(t - t_0) + s \cdot S_{\mathrm{template}}(t - t_0) + \mathrm{offset}$$



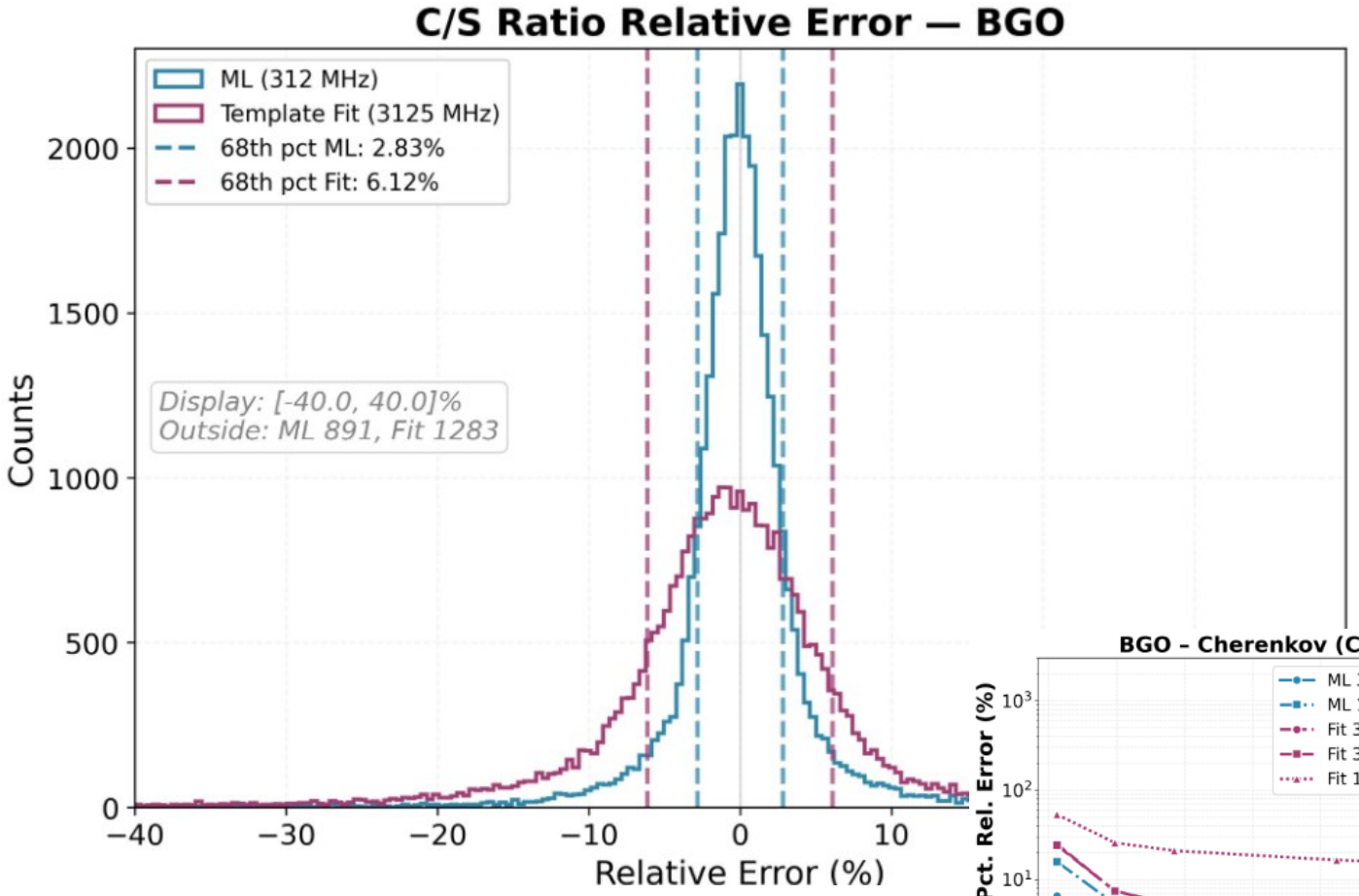
- Assuming linear scaling of the pulse
- Initial value and fitting range need tuning
- Fit the C and S component according to template
- Only possible offline due to the minimization computation

ML-based Algorithm



- One-pass regression with NN network
- Learn the parameters on simulation data
- Predict directly the number of C-photon and S-photon, as well as initial time information for more analysis
- Online eligible with FastML deployment workflow

Physics Performance and Resources Utilization



Training/Testing samples mixed 50% electron and 50% pion

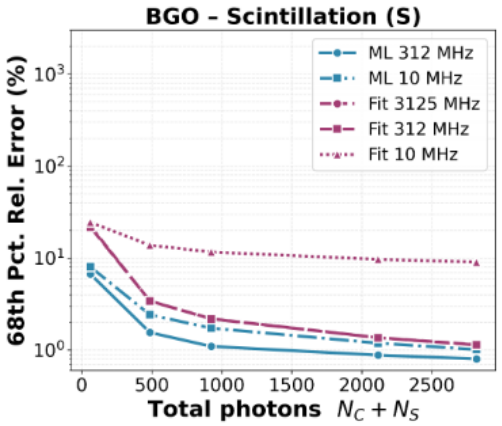
- Provide wide range of C/S ratio and reduce C/S correlation

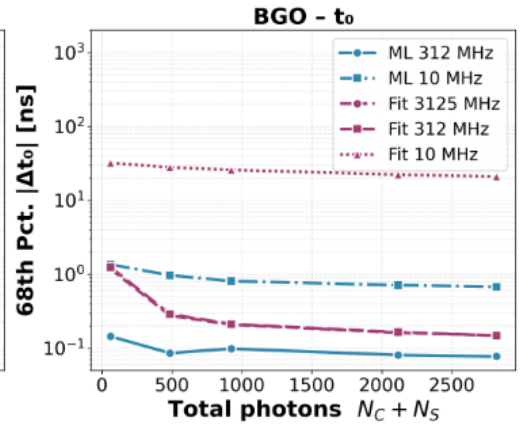
Good performance achieved with ML-based algorithm

- Central value correct and resolution highly improved

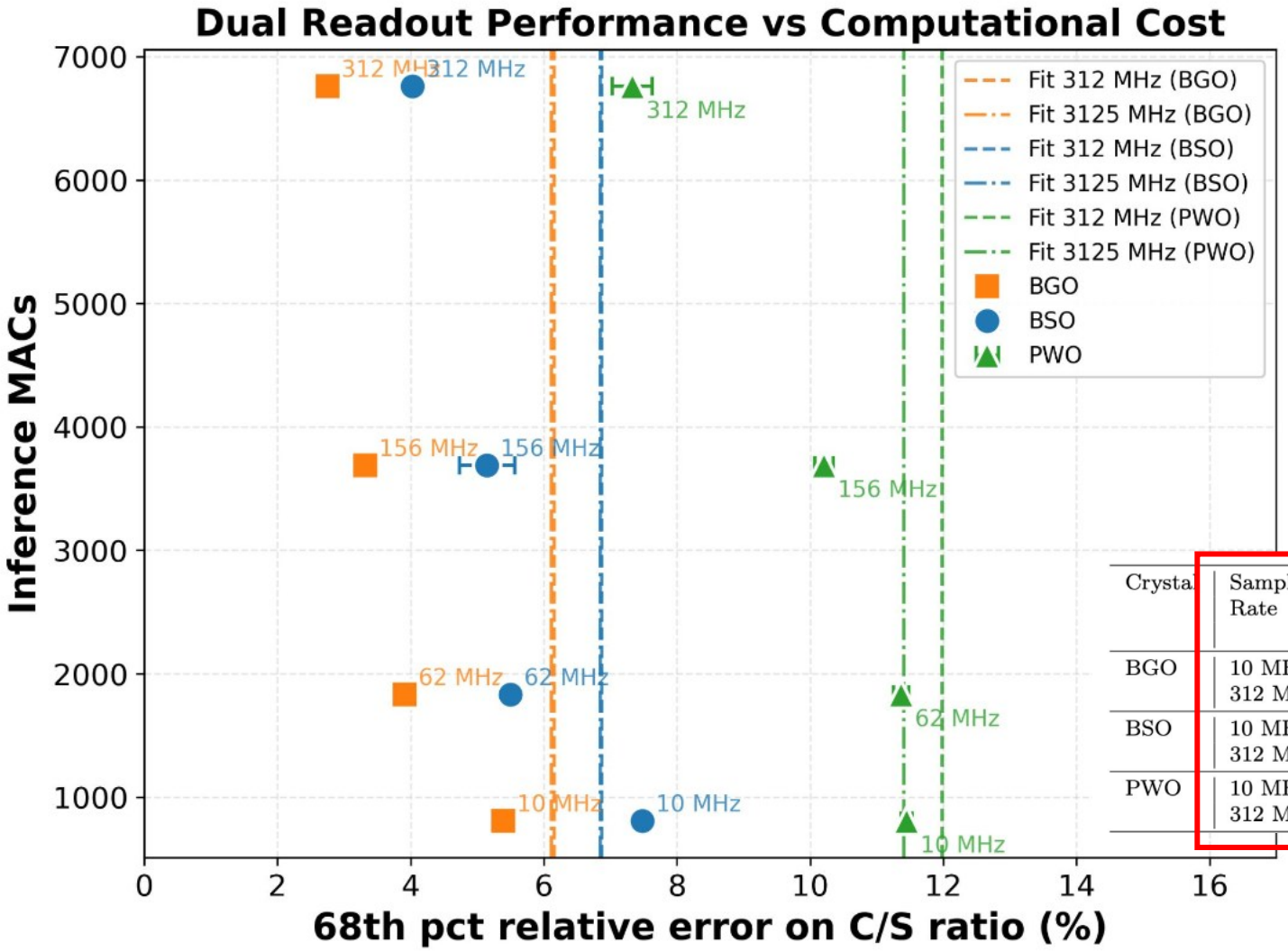
Evaluation across all the energy range

- 1, 5, 10 and 30 GeV incident E: ML-based method wins.
- The t0 prediction error reduced to sub-ns level





Physics Performance and **Resources Utilization**



ML-based method outperforms offline fitting

- Achieved with much slower sampling rate
- Low latency, low power and less requirement for ADC

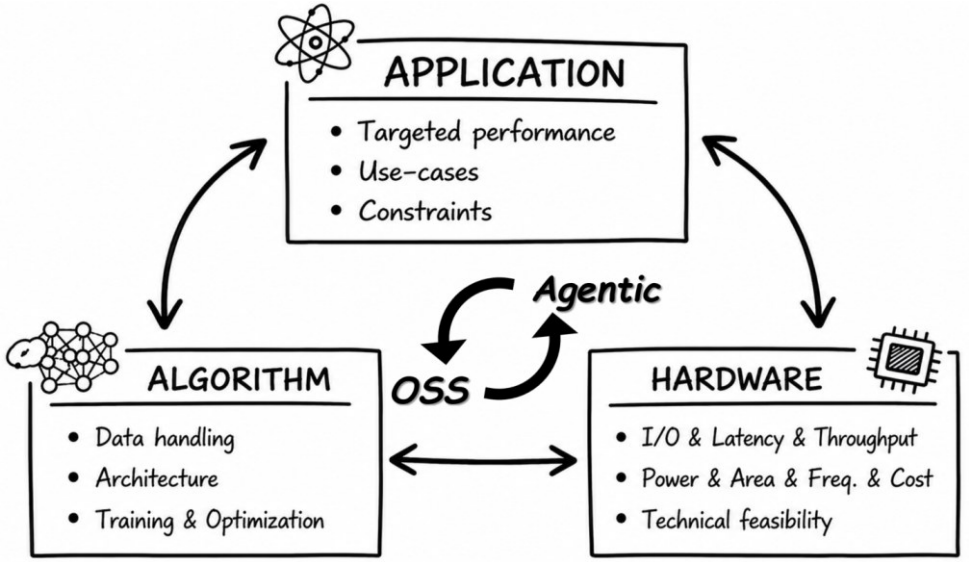
Small model enables the promising on-chip deployment

- 6% error at 30K LUTs and no DSP
- Purely combinational and simple for system integration

Crystal	Sampling Rate	Compression Strategy	Est. Freq [MHz]	LUT	FF	DSP	ML err68	Fit err68 @3GHz
BGO	10 MHz	40% sparsity, ⟨10, 1⟩ QAT	57.5	36,879	213	0	**5.84%**	6.12%
	312 MHz	30% sparsity, ⟨10, 2⟩ QAT	46.7	312,914	3,933	0	**3.08%**	6.12%
BSO	10 MHz	50% sparsity, ⟨10, 2⟩ QAT	57.7	30,471	213	0	8.14%	6.85%
	312 MHz	30% sparsity, ⟨10, 4⟩ QAT	46.6	328,953	3,933	0	**4.18%**	6.85%
PWO	10 MHz	50% sparsity, ⟨10, 2⟩ QAT	57.7	30,321	213	0	11.47%	11.41%
	312 MHz	30% sparsity, ⟨10, 4⟩ QAT	46.6	311,282	3,933	0	**7.40%**	11.41%

Outlook:

*Boost co-design loop for on-chip **AI** with agentic **AI** workflow*



In Production: Physics Algorithm → On-chip Application

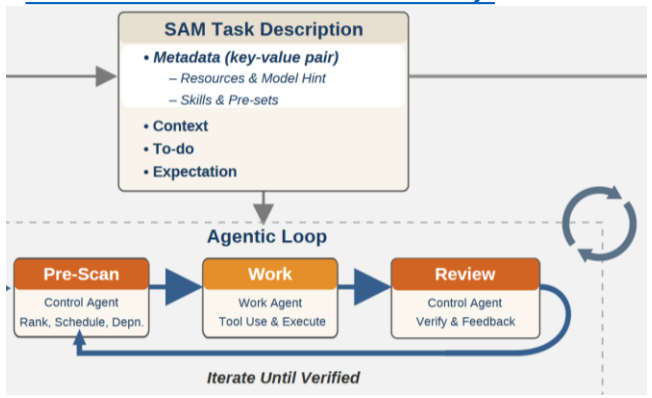
On-chip co-design motivates automated agentic workflows

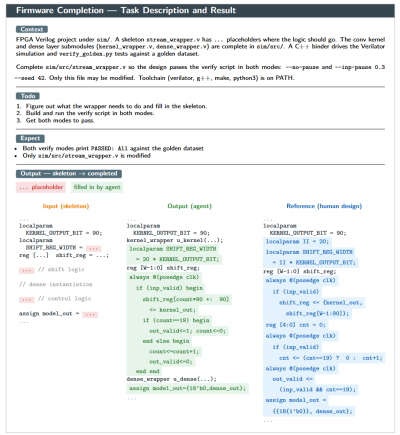
- Bash outputs, error logs, and commands are all "**language**"
- **Large Language Models (LLMs)** choose the most probable "**answer**", like human experts
- Close the loop: anything technically verifiable can be agentically solved

On-chip AI design reproduced with a fully agentic workflow

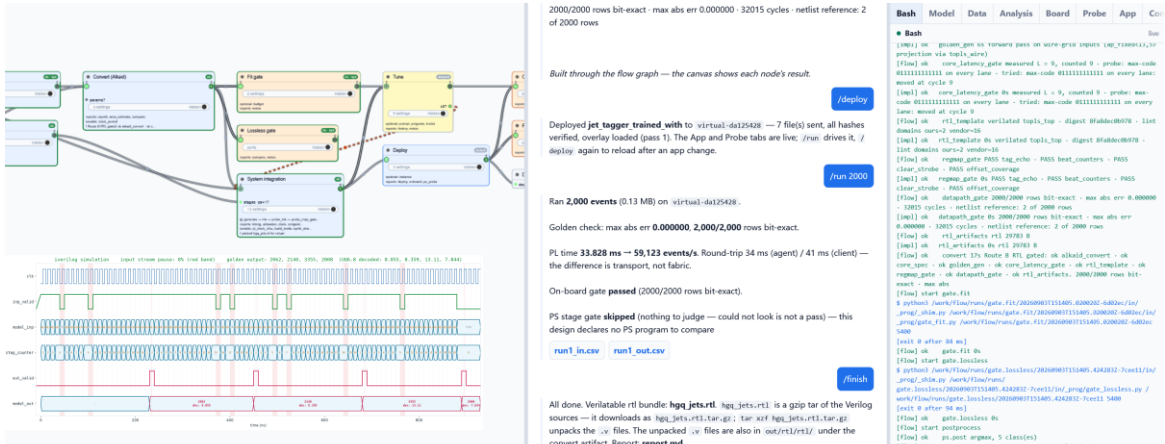
- Beyond feasibility — already in production, with a closed loop extending to real hardware (power cycle, I/O, …)
- Skills and data accumulate, accelerating future designs
- Towards **any-algorithm**, **any-hardware** deployment

SciFi & Feasibility study for LLM-generated RTL
IEEE RT2026: Case study





FastML demo: LLM-driven end-to-end design platform



In General: Physics Task → Reality Implementation

It is not limited to on-chip AI

Any technically intensive task — agentic workflows can and will do better

Detector and MDI optimization – **Agentic-based HPO**

Offline ML-algorithm (reconstruction, tagging, …) – **AI scientists**

Accelerator and beam control – **Agentic real-time control**

And more…

Summary

High 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 device, eFPGA, provides fully customized, **radiation-hard** and **small-size low-power** solutions.

ML-based applications developed target real-time and on-detector data processing:

- Ultra-small ML designed for clustering/counting algorithms in drift chamber detectors
- ML-based C/S waveform decomposition algorithms in dual-readout calorimeter

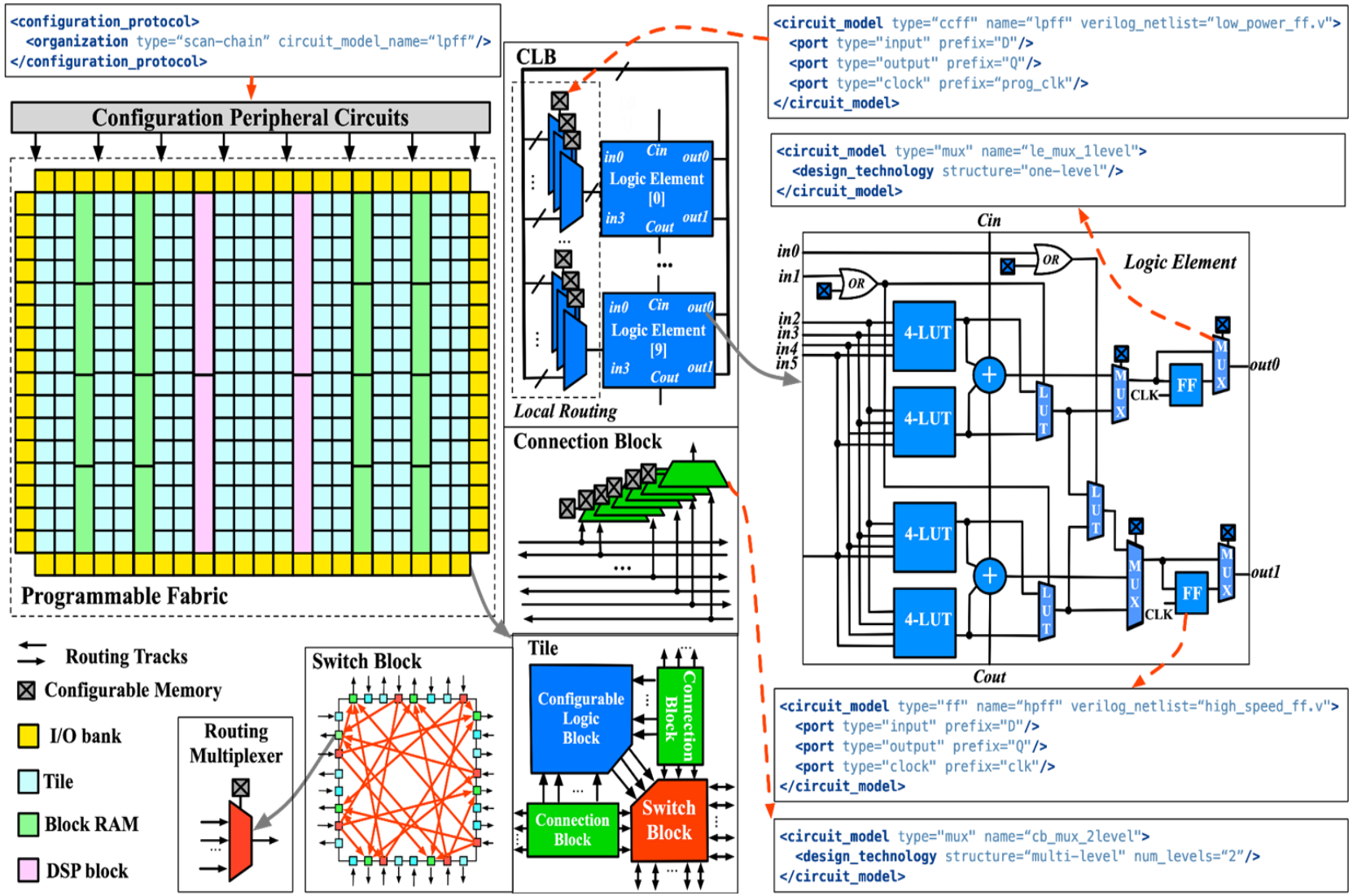
Looking forward:

- Technology for intelligence on chip is ready, stay tuned for more applications for FCC detector & TDAQ
- Agentic AI bridges the gap from design to implementation, bringing physics idea to reality faster
- New intelligence, new technology, new design → new possibility for new physics

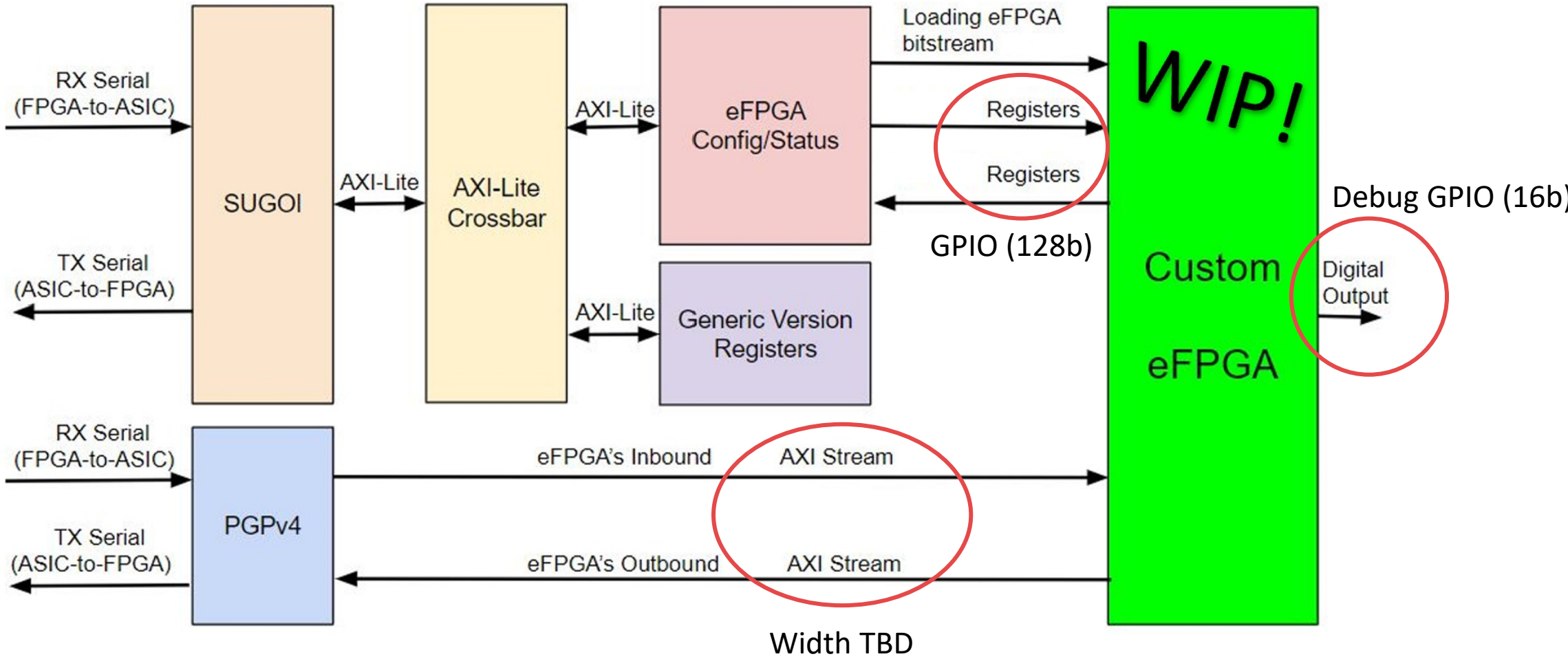
Thanks!

Low-Level Overview of eFPGA Design and Implementation

openFPGA



eFPGA: System Integration



ML@FPGA: growing HEP demand

Experimental HEP is fundamentally data-driven:

- **The more information extract from data, the higher the chance of discovering new physics**

Luminosity is basically decided beforehand: Improvements come from better use of the same data.

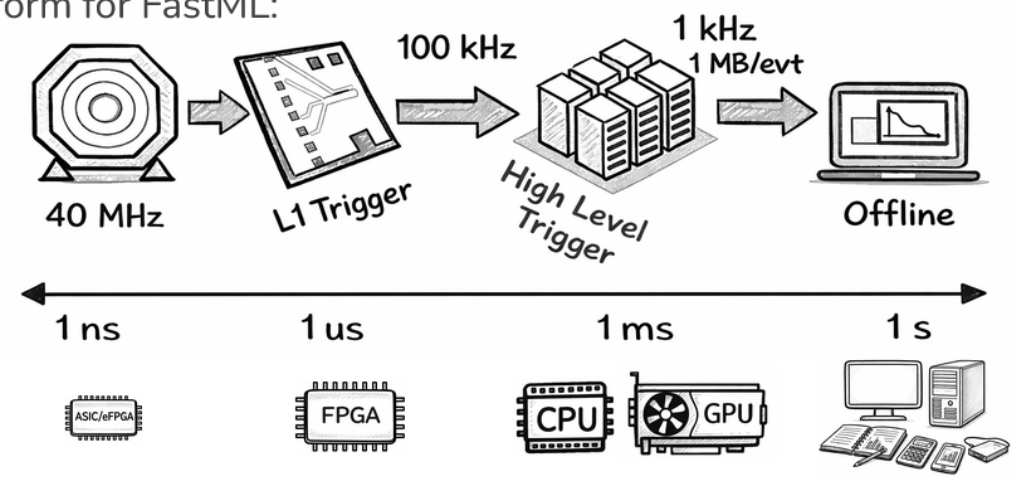
- Storage and bandwidth are limiting factors: can not save&load everytime
- Data analysis is moving earlier in the chain: pre-selection and pre-processing

FastML: Low-latency and small footprint method shipping with scalable performance

FPGA(Field-Programmable Gate Array) provides the best platform for FastML:

- Configurable, deterministic and relatively low cost
- Fine-grained parallel computation at any level
- Industrial and open source community support

Wide applications: tau ID, b-tagging, egamma ID, anomaly detection, LAr reconstruction, data compression, tracking, RF control, magnets control...



Basics of FPGA

C = A & B

Input A	Input B	Output C
0	0	0
0	1	0
1	0	0
1	1	1

Any N→1 bit operation can be implemented using a Look-Up Table (LUT): logic gate

N→M and **integer (binary) arithmetic** construct from LUTs using Boolean algebra: Y=F(X)

E.g. (A+B): Sum = (!A & B) | (A & !B), Carry = A & B

Clock and storage elements (flip-flops, memory, etc.) introduce the time: Y(t) = F(X(t), X(t-1))

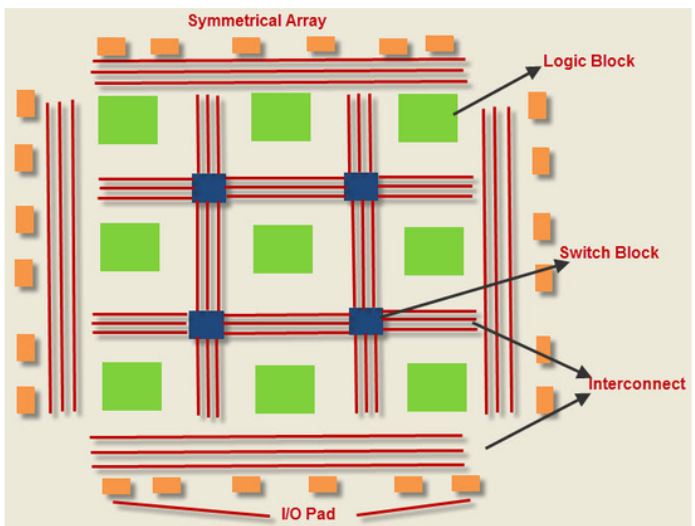
E.g. (A/2): A>>1 implemented as A[t,i] = A[t-1,i+1]

FPGA = LUTs + storage elements + configurable wires

Commonly used parts can also be provided as building “blocks”:

- DSP (scalar multiplication and accumulation)

- AI/ML engine (vector and tensor operation)



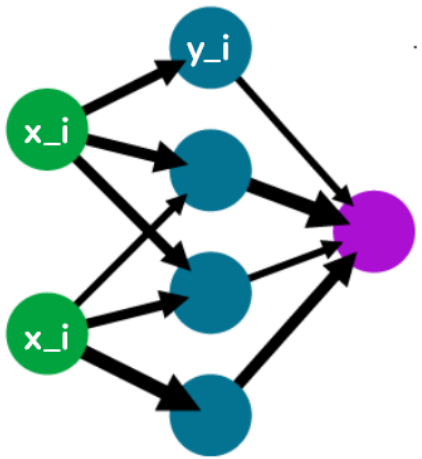
★: will be covered today

★: eFPGA related work in progress and will report soon!

How the ML actually runs on FPGA

A simple neural network

input layer hidden layer output layer

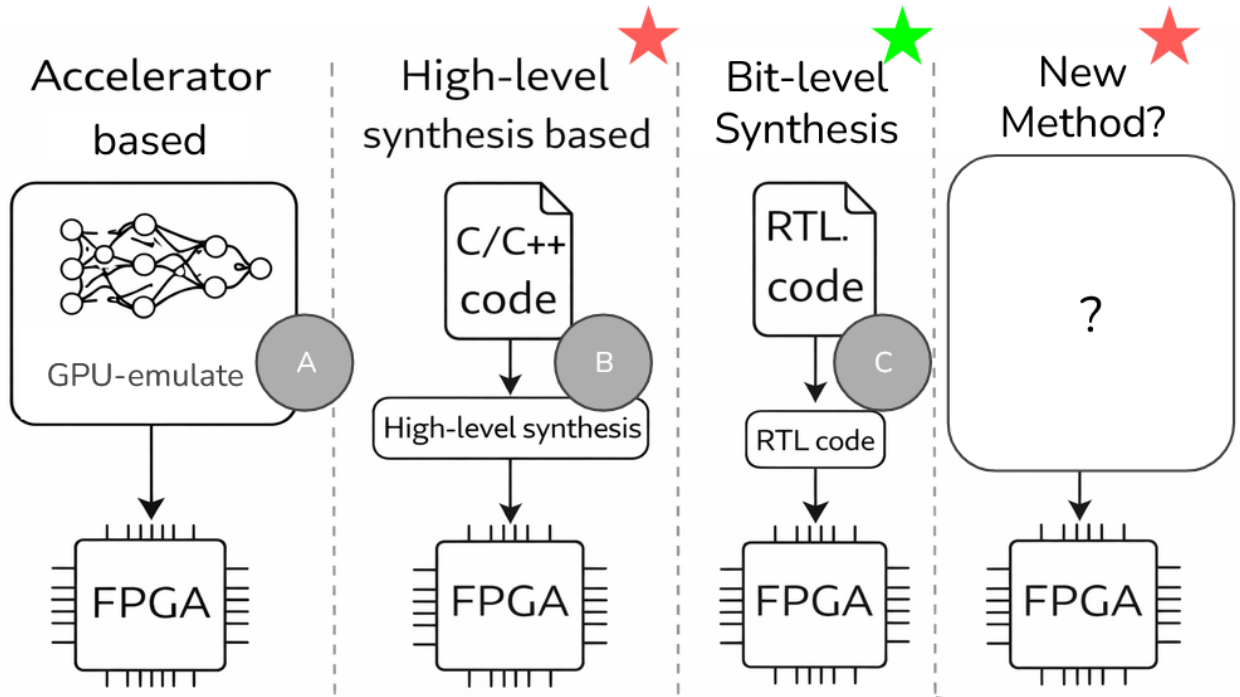


$Y=\theta(WX+B)$

θ Activation: Element-wise function

W weight: Multiply-Add (dot product)

B bias: Addition




```
mac_kernel(...){
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    y[idx] = w[idx] * x[idx] + b[idx]
}
// ...perform on N matrices
mac_kernel<<<blocks, threads>>>(X, W, B, Y, N);
```

Matrix/Tensor Op.

A $\vec{Y} = \theta(W \cdot \vec{X} + \vec{B})$

```
for (int i = 0; i < N; i++) {
    z += w[i] * x[i];
}
```

Scalar Op. + Loop

B $Y_i = \sum_j \theta(w_{i,j}X_i + b_i)$

```
module mac2u(input  [1:0] W, X,
             input  [2:0] B,
             output [4:0] Y);
  wire w0=W[0], w1=W[1], x0=X[0], x1=X[1];
  wire t0=w0&x0, t1=w1&x0, t2=w0&x1, t3=w1&x1;
  wire p0=t0, p1=t1^t2, c1=t1&t2, p2=t3^c1, p3=t3&c1;
  assign S = {1'x0,p3,p2,p1,p0} + {2'x00,B};
endmodule
```

C Bit Operation

ML@FPGA: challenging co-design task

Performance not come for free: limited space, power, and hardware choices

- We never care which GPU card runs the model, but for FPGA we should trace down to block
- Specialized models, customized inference pipelines, and dedicated hardware mapping Needed!

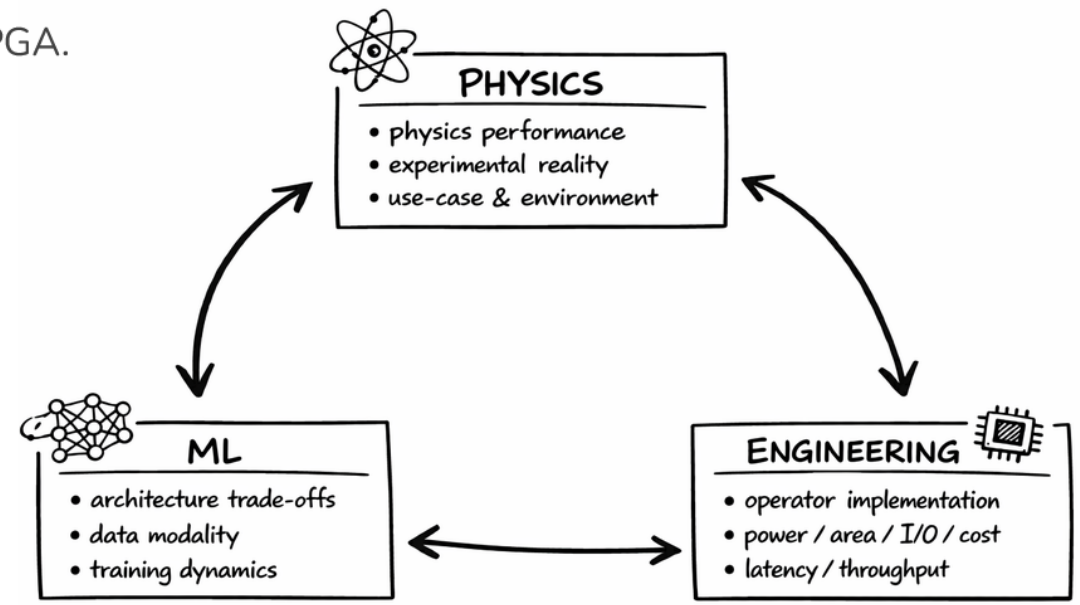
Co-design is essential to fully unlock the power of ML@FPGA.

- Physics → application and contraint
- ML → methodology and data stream
- Engineering → implementation and cost

More efforts needed but all the technology ready!

- Existing physics studies and use cases
- Mature open-source ML frameworks
- Industrial-grade hardware design tools

Co-design integrates technologies into a complete solution



Co-Design Loop
(iterate & converge)

Firmware Completion — Task Description and Result

Context

FPGA Verilog project under `sim/`. A skeleton `stream_wrapper.v` has `...` placeholders where the logic should go. The conv kernel and dense layer submodules (`kernel_wrapper.v`, `dense_wrapper.v`) are complete in `sim/src/`. A C++ binder drives the Verilator simulation and `verify_golden.py` tests against a golden dataset.

Complete `sim/src/stream_wrapper.v` so the design passes the verify script in both modes: `--no-pause` and `--inp-pause 0.3 --seed 42`. Only this file may be modified. Toolchain (verilator, g++, make, python3) is on PATH.

Todo

1. Figure out what the wrapper needs to do and fill in the skeleton.
2. Build and run the verify script in both modes.
3. Get both modes to pass.

Expect

- Both verify modes print `PASSED: All` against the golden dataset
- Only `sim/src/stream_wrapper.v` is modified

Output — skeleton → completed

`... placeholder` filled in by agent

Input (skeleton)

```
...
localparam
  KERNEL_OUTPUT_BIT = 90;
localparam
  SHIFT_REG_WIDTH = ...
reg [...]  shift_reg = ...;

... // shift logic

// dense instantiation

... // control logic

assign model_out = ...
...
```

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};
...
```

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.
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.

Hardware-aware ML training and RTL conversion

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

- Dynamic and trainable quantization and EBOPs/LUTs estimation during training
- Efficient PID and pareto front scan, balance hardware-aware optimization
- Direct-RTL(DA4ML) backend and agentic-based RTL wrapping, HLS-free and compact

Algo: Differentiable Quantization

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

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

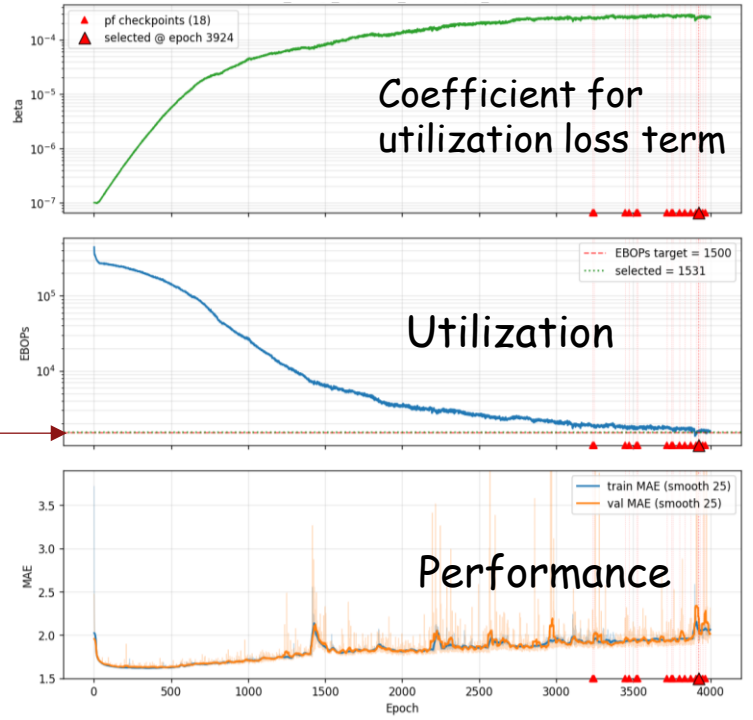
2: The gradient with respect to f is given by

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

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

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

Target resources

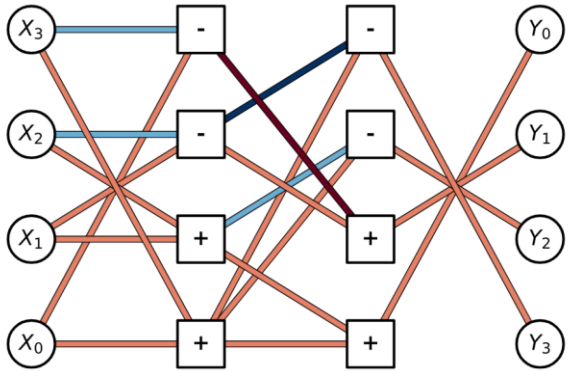


Sun, C., Que, Z., Loncar, V., Luk, W., & Spiropulu, M. (2026). da4ml: Distributed arithmetic for real-time neural networks on fpgas. ACM Transactions on Reconfigurable Technology and Systems, 19(1), 1-27.
Liu, Q., & Gonski, J. (2026). SciFi: A Safe, Lightweight, User-Friendly, and Fully Autonomous Agentic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

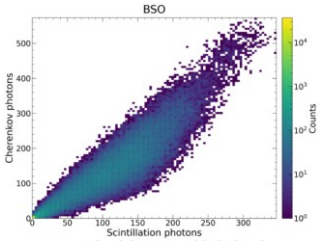
Output (agent)

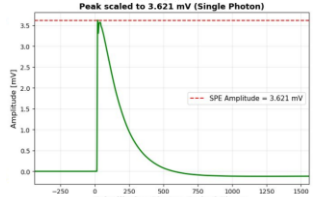
```
...
localparam
  KERNEL_OUTPUT_BIT = 90;
kernel_wrapper u_kernel(...);
 localparam SHIFT_REG_WIDTH
   = 20 * KERNEL_OUTPUT_BIT;
reg [W-1:0] shift_reg;
 always @(posedge clk)
   if (inp_valid) begin
     shift_reg[count*90 +:  90]
       <= kernel_out;
     if (count==19) begin
       out_valid<=1;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

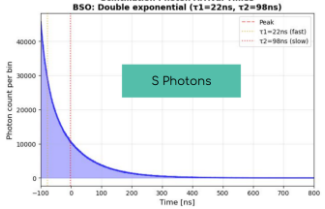
Reference (human design)

```
...
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};
...
```

Digitization/Waveform Simulation







E, v, Light yield

Poisson & efficiency

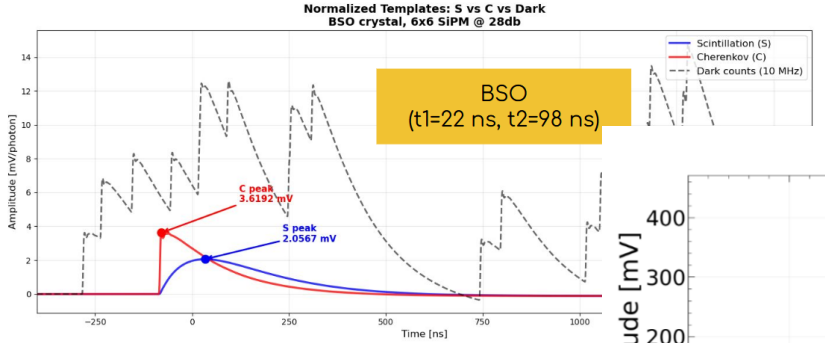
Genat4-based simulation → Truth S and C photon count → S and C photon count

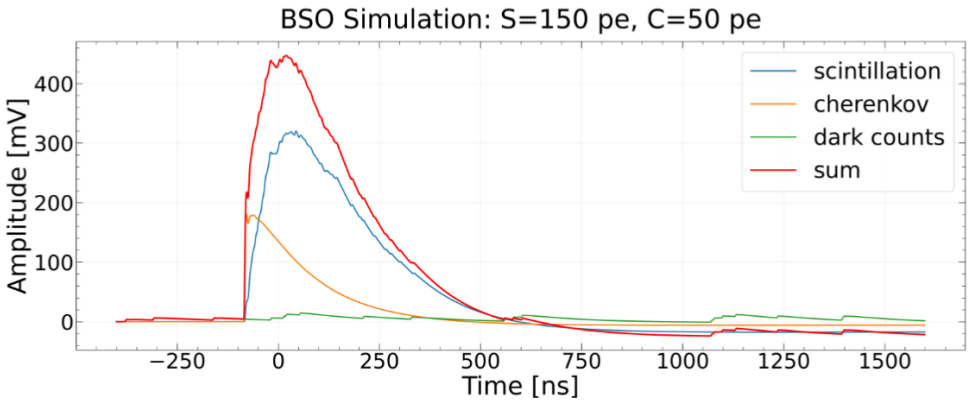
Single Photon Response (SPR)

Time profile acc. to xtal type

Sample S and C → Normalized S and C template

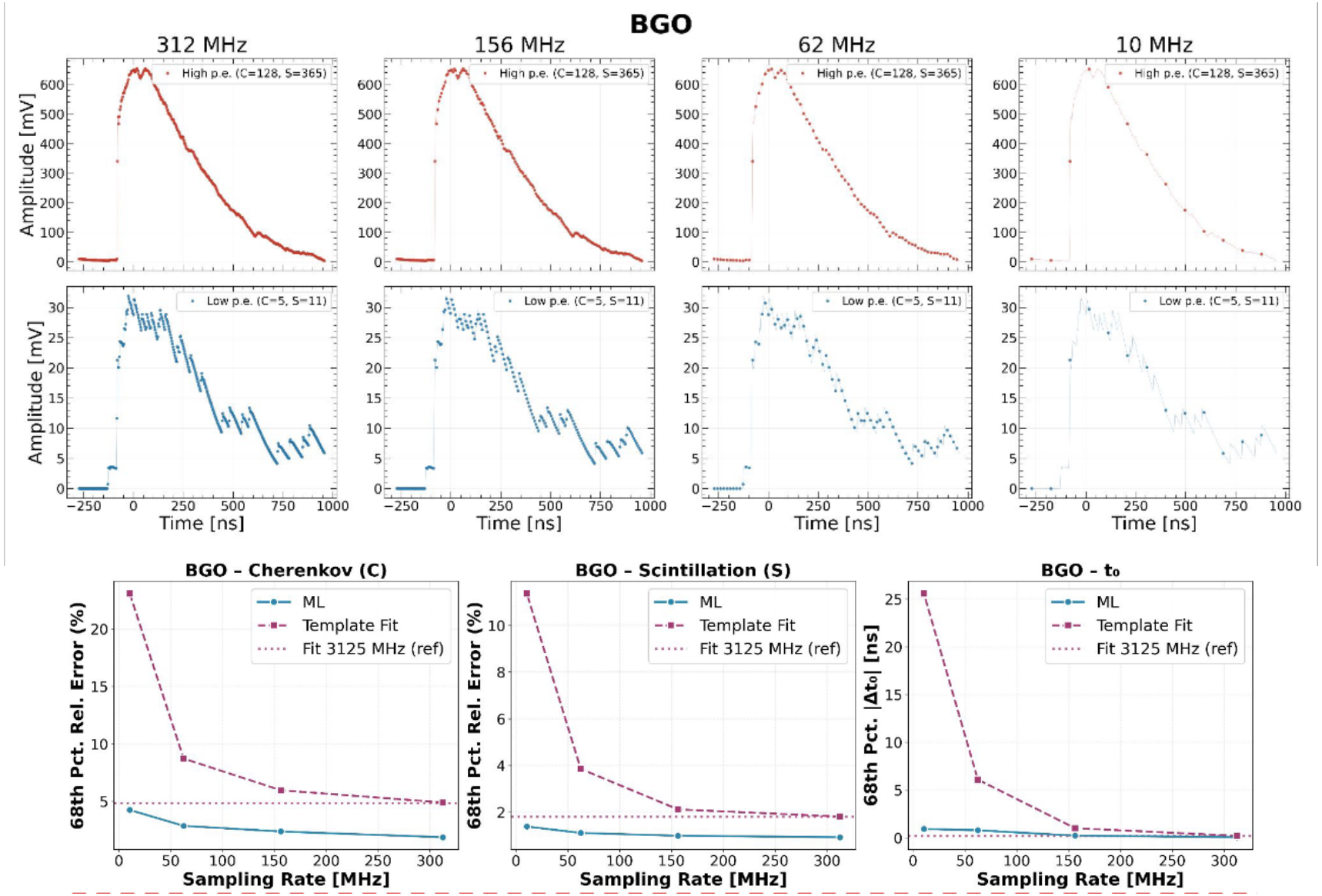
Sample Photon → Add dark count



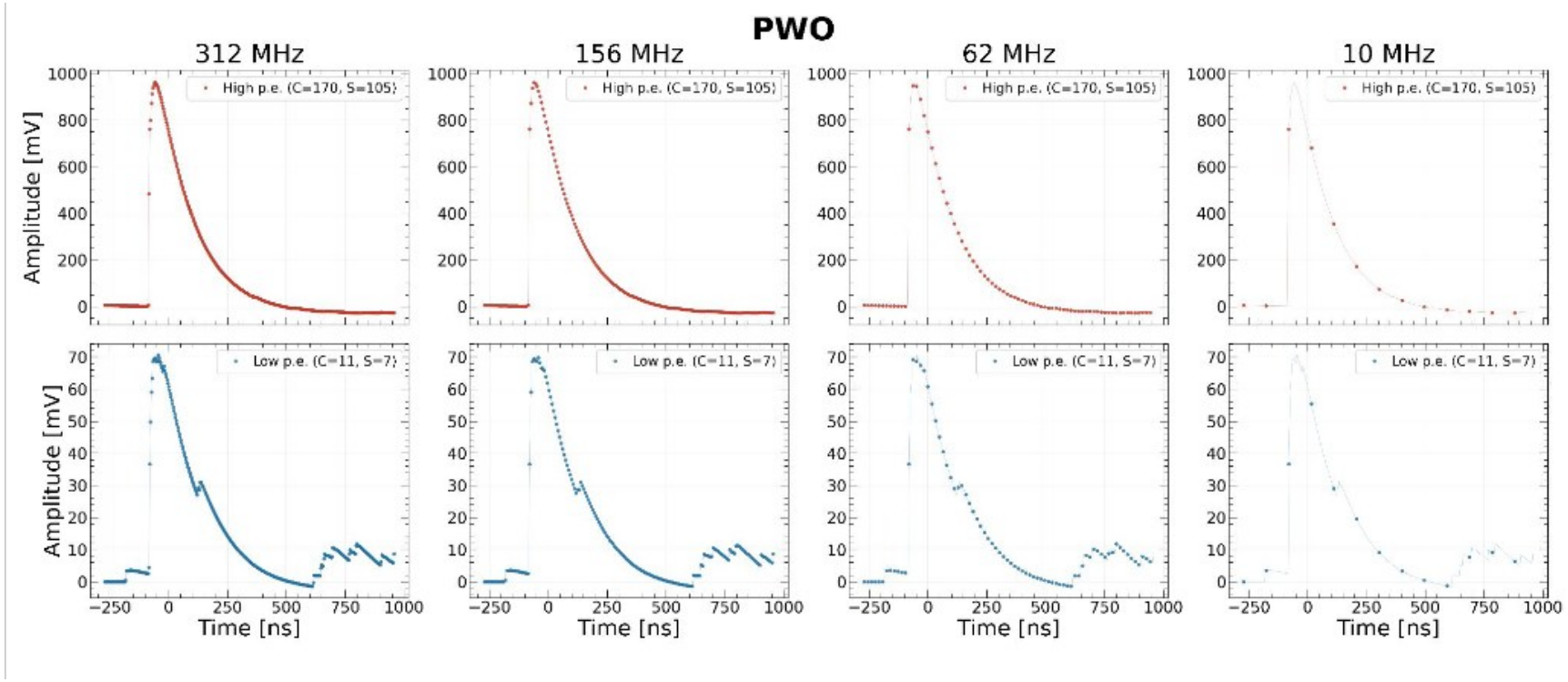


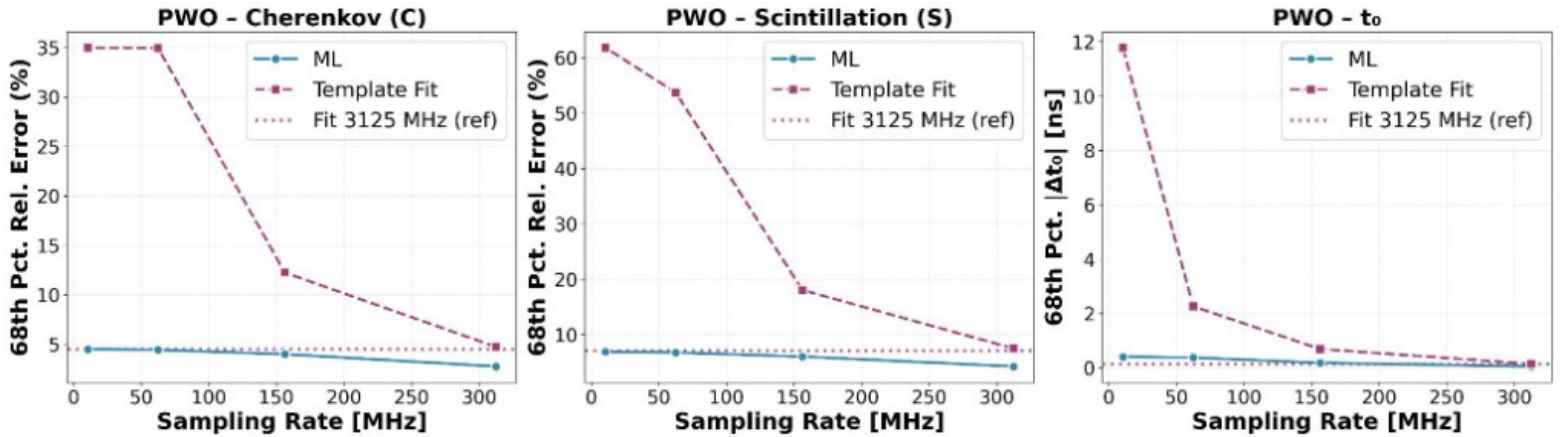
Property	BGO	BSO	PWO
Scintillation light yield (phe/GeV)	170	30	30
Cherenkov light yield (phe/GeV)	50	50	50
Decay model	Bi-exp	Bi-exp	Single-exp
Fast decay constant τ_1 (ns)	50	22	7.5
Slow decay constant τ_2 (ns)	320	98	—

Down-sampled Waveform



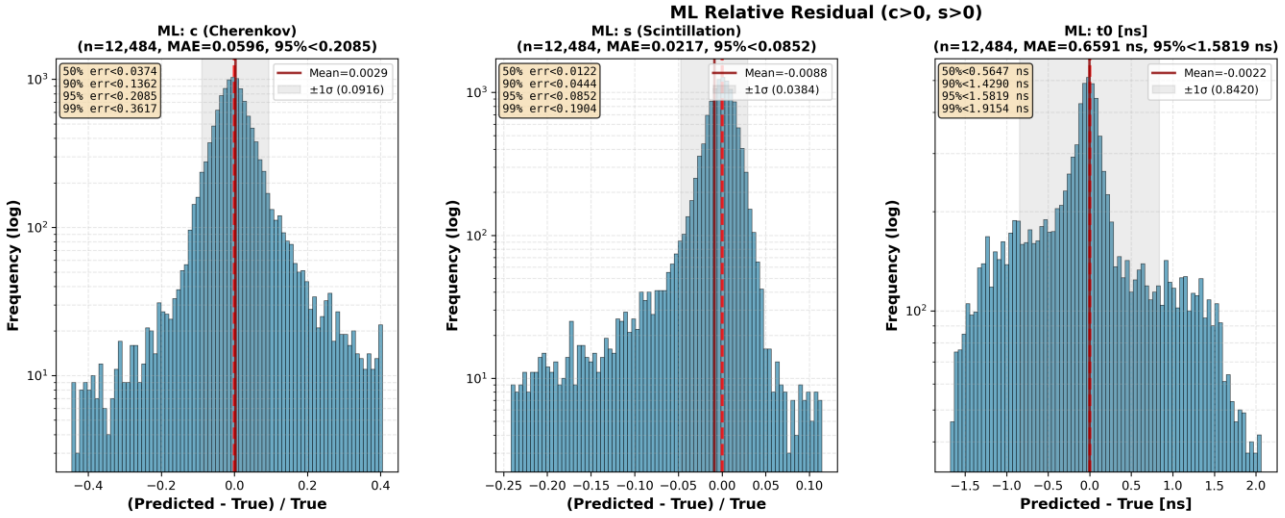
Down-sampled Waveform





Electron/Kaon separated evaluation

Kaon



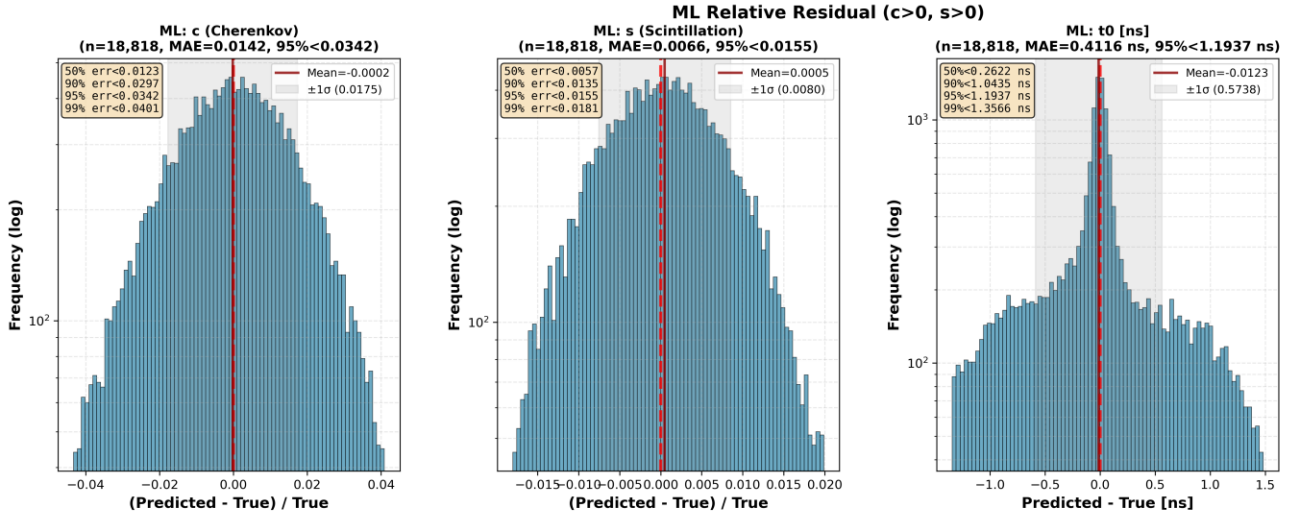

```
ML Statistics
==================================================
Filter: c>0, s>0
Events: 13,142/20,000

c (Cherenkov):
  Events:  12,484 (clip 658)
  Mean:    0.002884
  Std:     0.091633
  MAE:     0.059624
  50% err: 0.037365
  90% err: 0.136175
  95% err: 0.208521
  99% err: 0.361734

s (Scintillation):
  Events:  12,484 (clip 658)
  Mean:    -0.008797
  Std:     0.038372
  MAE:     0.021747
  50% err: 0.012167
  90% err: 0.044375
  95% err: 0.085177
  99% err: 0.190416

t0 [ns]:
  Events:  12,484 (clip 658)
  Mean:    -0.002170
  Std:     0.842015
  MAE:     0.659111
  50% err: 0.564746
  90% err: 1.428998
  95% err: 1.581906
  99% err: 1.915381
```

Electron




```
ML Statistics
==================================================
Filter: c>0, s>0
Events: 19,810/20,000

c (Cherenkov):
  Events:  18,818 (clip 992)
  Mean:    -0.000193
  Std:     0.017545
  MAE:     0.014238
  50% err: 0.012309
  90% err: 0.029664
  95% err: 0.034232
  99% err: 0.040109

s (Scintillation):
  Events:  18,818 (clip 992)
  Mean:    0.000495
  Std:     0.008040
  MAE:     0.006574
  50% err: 0.005738
  90% err: 0.013516
  95% err: 0.015508
  99% err: 0.018104

t0 [ns]:
  Events:  18,818 (clip 992)
  Mean:    -0.012294
  Std:     0.573823
  MAE:     0.411621
  50% err: 0.262182
  90% err: 1.043543
  95% err: 1.193723
  99% err: 1.356556
```