Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Implementation of Machine Learning algorithms on low power real time hardware devices for future space missions

A. Abba, V. Arosio, F. Caponio, S. Carsi, A. Giannini

Spoke: F. Gargano, F. Cuna, M. Bossa

Nuclear Instruments

Spoke 3 III Technical Workshop, Perugia 26-29 Maggio, 2025

Nuclear Instruments

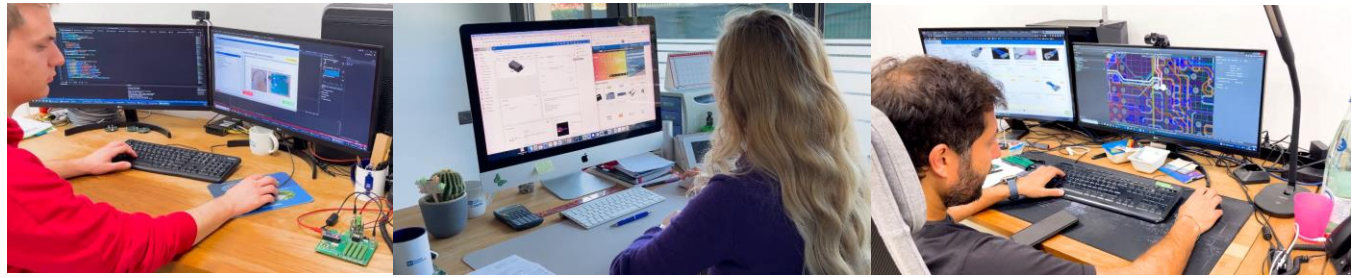
Who we are
- an outstanding Italian company
- Team of highly qualified young engineers, physicists, and computer scientists (22 people)

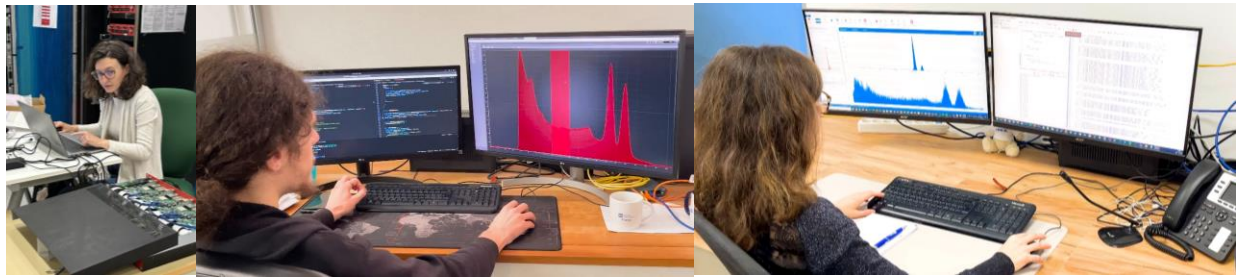
Our mission
- Design and develop advanced electronic solutions
- Support cutting-edge scientific experiments

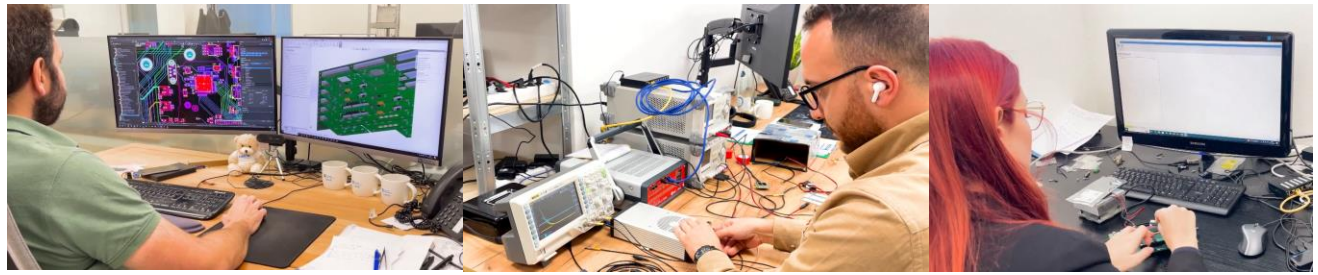
Research areas
- Study of matter's structure in large colliders
- Astrophysics
- Aerospace
- Nuclear Medicine

Operating model
- Function as a true research laboratory
- Strong focus on innovation

Collaborations
- Partnerships with leading scientific institutions worldwide
- Contributing expertise to advance science and technology in strategic, pioneering fields







Projects in charge of Nuclear Instruments

- **SPARTAN: SPace pARTicle trAcking with Neural-networks**

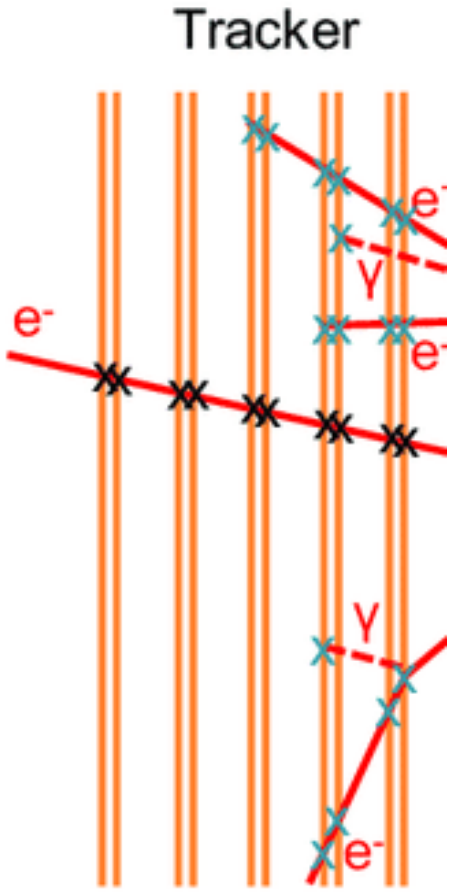
Development of a real-time particle tracker for space applications using machine learning and FPGA technology. The project aims to reconstruct particle trajectories with high precision and low energy consumption, enabling operation in dense event environments like cosmic radiation detectors.

- **LEGIMaC: Low-Energy Gamma Imaging via Machine Learning in Calorimeters**

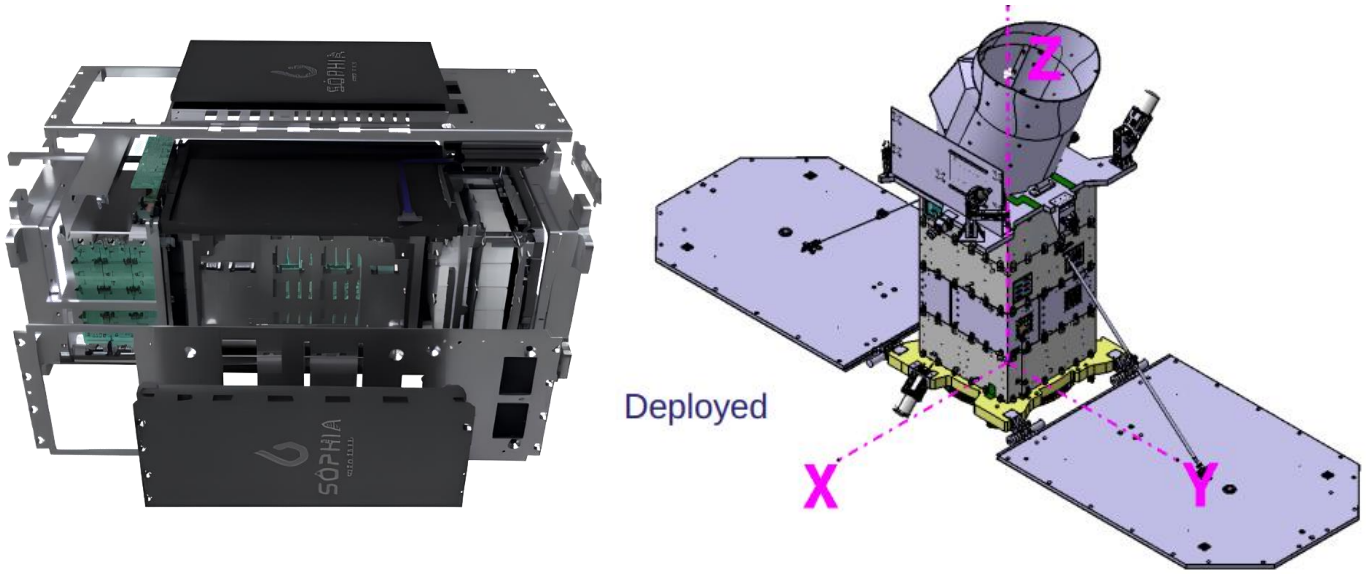
Creation of ML algorithms for detecting ultra-low energy gamma events in space calorimeters. The focus is on distinguishing scintillation signals from dark noise and resolving event pileup, even at high rates, through pulse shape analysis on GPUs and FPGAs

SPARTAN: What is Particle Tracking?

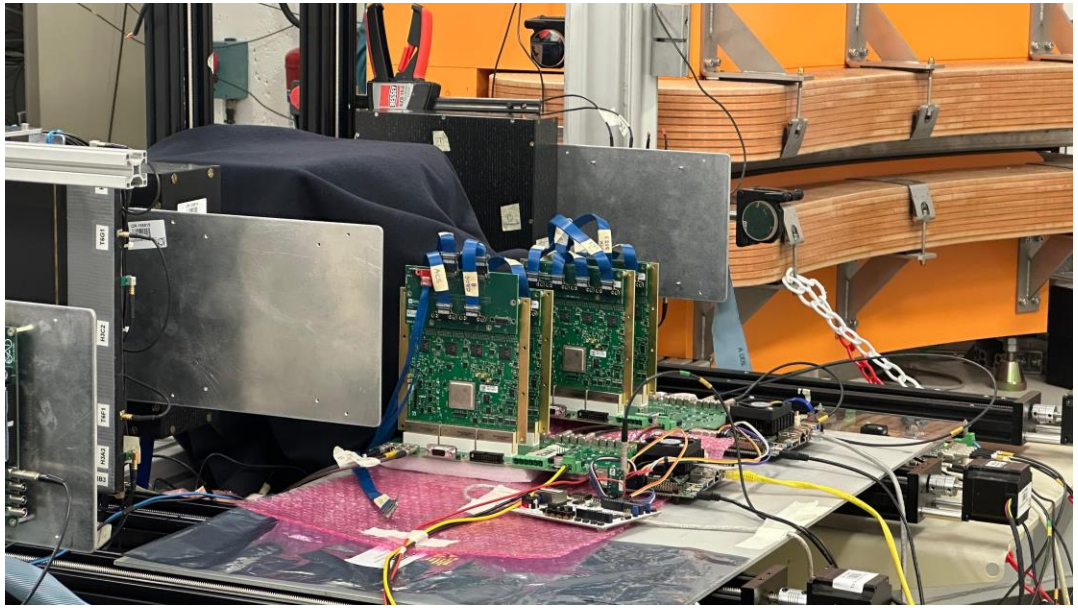
- Particle tracking is the process of reconstructing the trajectories of charged particles as they pass through a detector.
- It allows physicists to determine key properties of particles, such as momentum, charge, and origin, by analyzing their paths.
- Charged particles interact with detector materials, leaving signals (hits) that are used to map their curved paths, especially within magnetic fields.



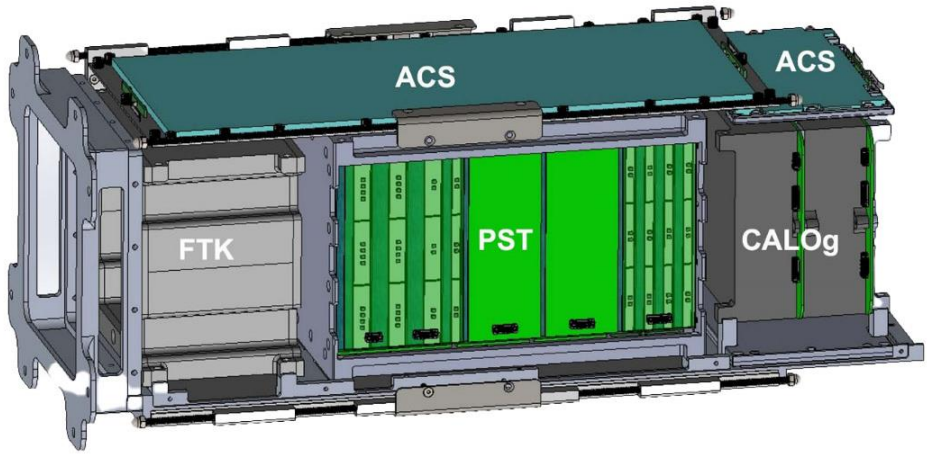
Zire payload on Nuses Satellite

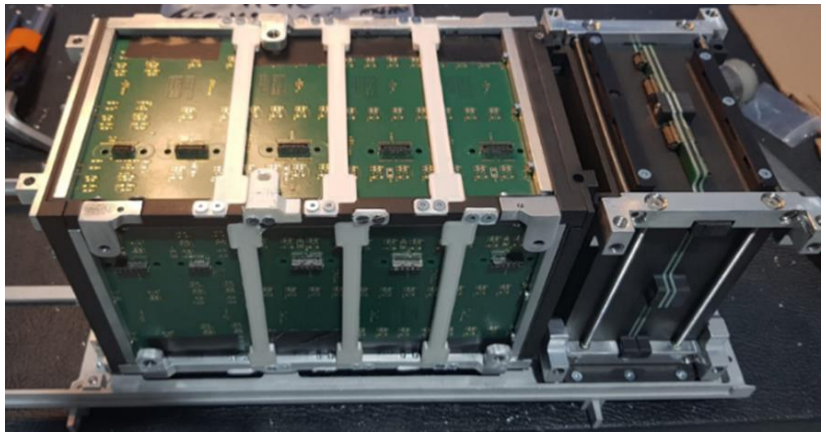


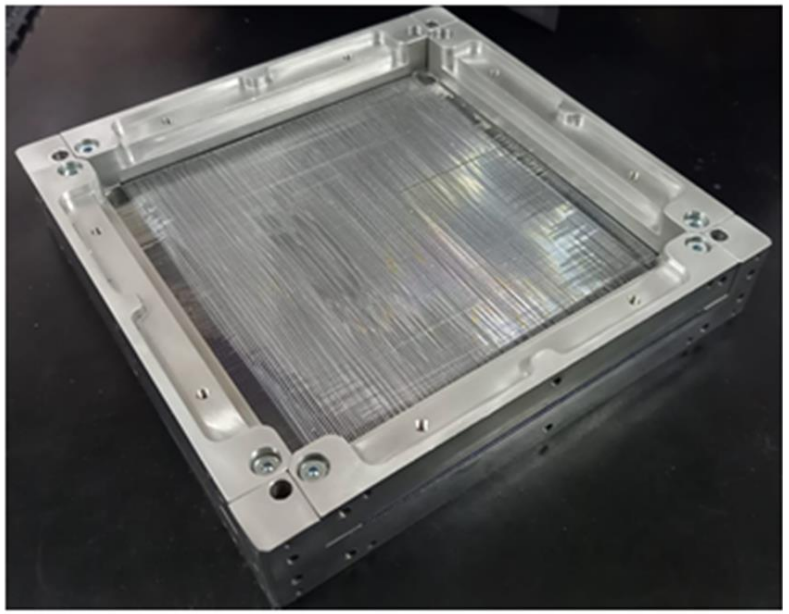
3D model of the satellite and of Zirè payload



Zire payload on Nuses Satellite



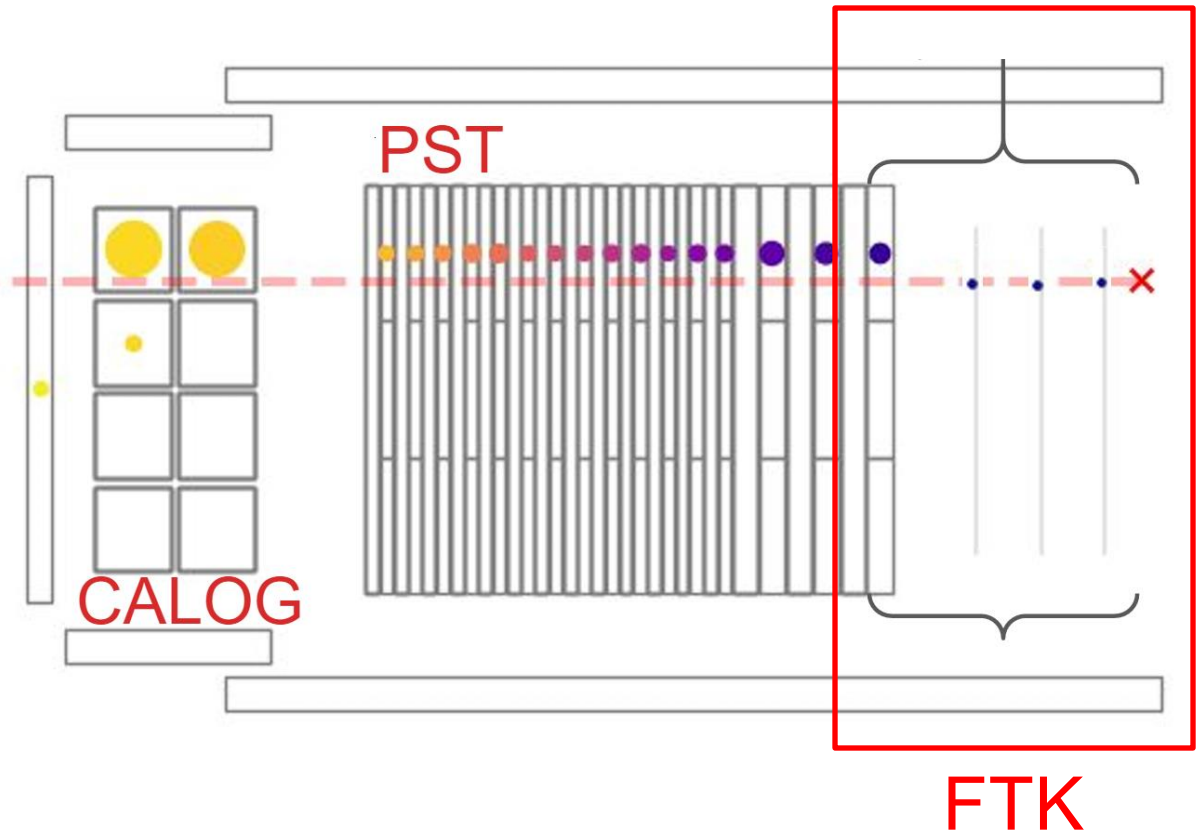




FTK Fiber Tracker

M.N. Mazziotta, F.Gargano, R.Pillera, G. Panzarini, L. Lorusso, A. Liguori

Zire payload on Nuses Satellite



- Fiber tracker → recostruction with ML
- Low power solutions → less than 10W

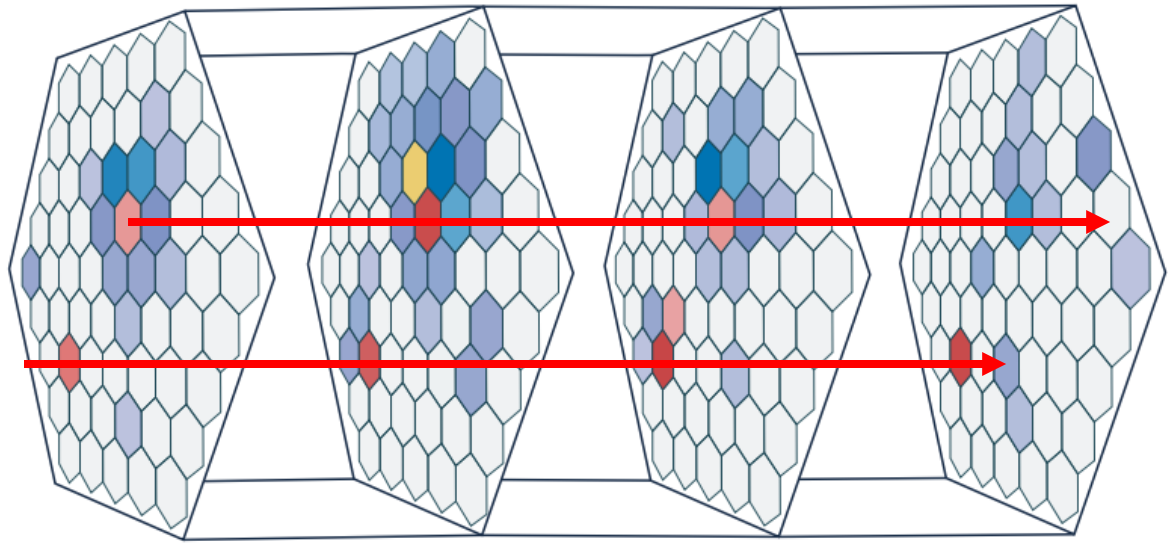
How Neural Networks Identify Particle Tracks

Input: Detector Hits

- Charged particles traversing a detector leave behind signals, or "hits," in various detector layers.
- These hits are recorded as spatial coordinates, representing the particle's interaction points.

Graph Construction

- Each hit is represented as a node in a graph.
- Edges are established between nodes based on spatial proximity, forming potential track segments.



Detector Input with possible tracks

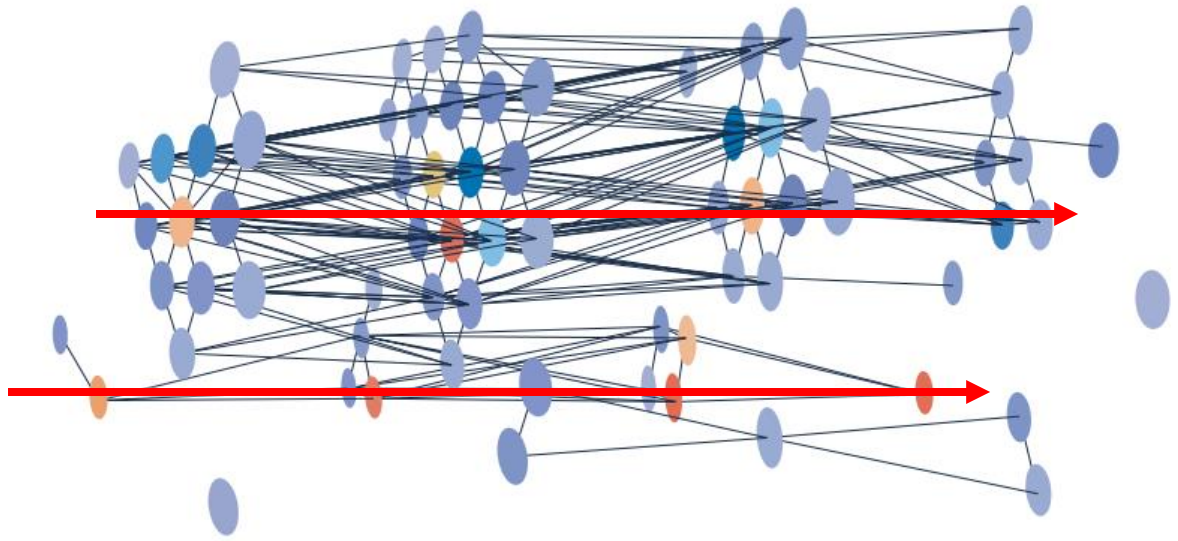
How Neural Networks Identify Particle Tracks

Graph Neural Network (GNN) Processing

- A Graph Neural Network analyzes the constructed graph to evaluate the likelihood that connected nodes (hits) belong to the same particle trajectory.
- Through iterative message passing between nodes, the GNN learns to identify patterns consistent with particle tracks.

Track Reconstruction

- Nodes with high confidence scores are selected to assemble complete particle trajectories.
- The network effectively filters out noise and resolves overlapping tracks, even in dense environment

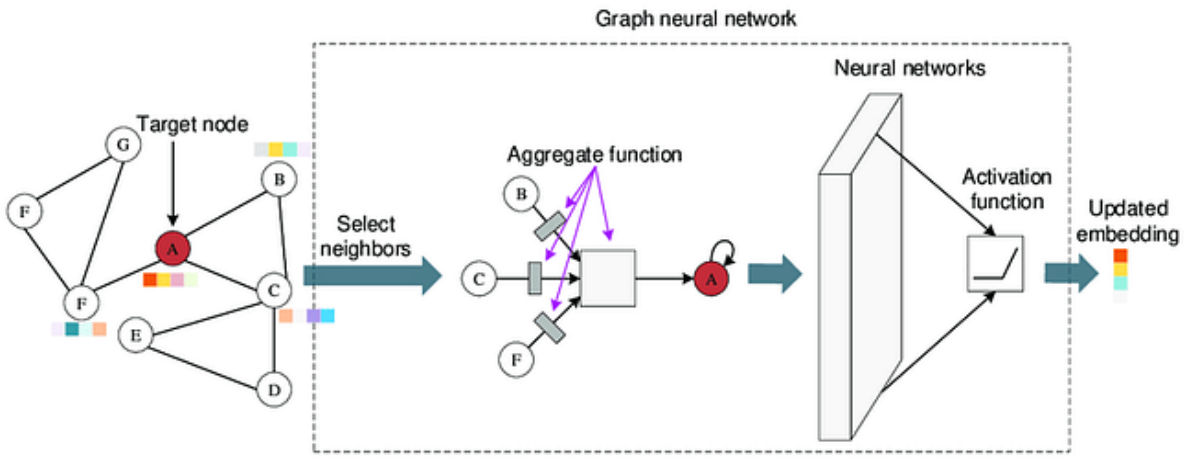


Edge connected each other and possible tracks reconstructed by the GNN

SageCONV Layers

How GNN Works

- **Message Passing**: Each node collects ("receives messages from") its neighbors.
- **Aggregation**: The received messages are combined using a function (e.g., mean, sum).
- **Update**: The node's feature is updated using the aggregated information and its own features.



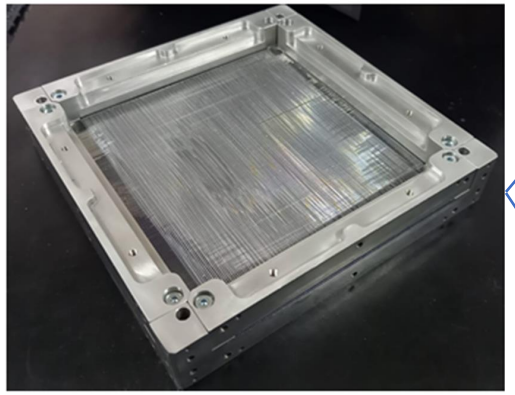
SAGEConv is a GNN layer introduced by GraphSAGE that enables **inductive learning**: SAGEConv learns a transferable aggregation function, enabling it to make accurate predictions on nodes and graphs never seen during training.

- **Neighbor Sampling**: Selects a fixed-size set of neighbors for each node → scalable to large graphs.
- **Aggregation Functions**:
  - Mean (default): Simple average of neighbor features
  - Max, LSTM, or even learned functions for richer modeling
- **Update Rule**: Combines the node's own features with the aggregated neighborhood representation

How Neural Networks Identify Particle Tracks

Aspect	Convolutional Neural Networks (CNNs)	Graph Neural Networks (GNNs)
Data Structure	Operate on regular, grid-like data structures (e.g., images), where spatial relationships are fixed.	Operate on graph-structured data, accommodating arbitrary sizes and complex topologies, ideal for irregular detector geometries.
Application in Particle Tracking	Transform detector hits into 2D or 3D images; apply convolutional filters to detect patterns corresponding to particle trajectories.	Model detector hits as nodes in a graph; use message passing to aggregate information from neighboring nodes, capturing complex spatial relationships.
Advantages	- Well-established architectures with extensive tooling and support. - Efficient for data with regular spatial structures.	- Naturally handle irregular and sparse data. - Adaptable to complex detector geometries. - Capture intricate relationships between hits.
Limitations	- May struggle with sparse data and complex topologies.	- Potentially higher computational complexity. - May require more sophisticated training and tuning.

Data Preprocessing Pipeline for Particle Tracking



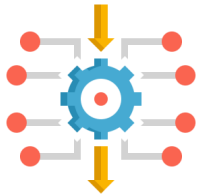
X layers

Clusters

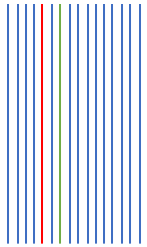
Pre-Processing

Nodes
Edges

Inference



Y layers

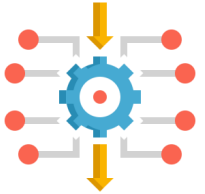


Clusters

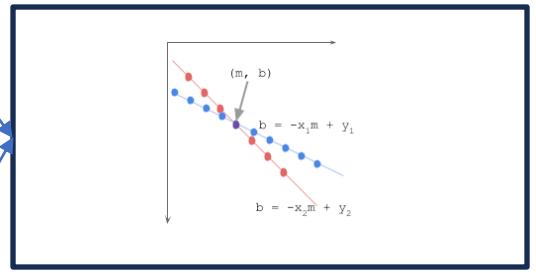
Pre-Processing

Nodes
Edges

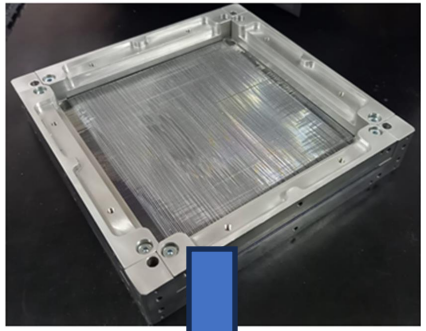
Inference

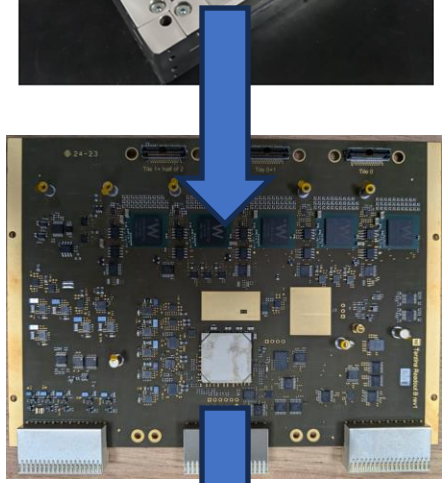


Track Fitting



Data Preprocessing Pipeline for Particle Tracking

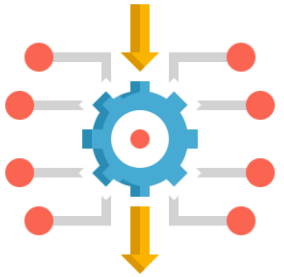




Store data on file

DATA SOURCE

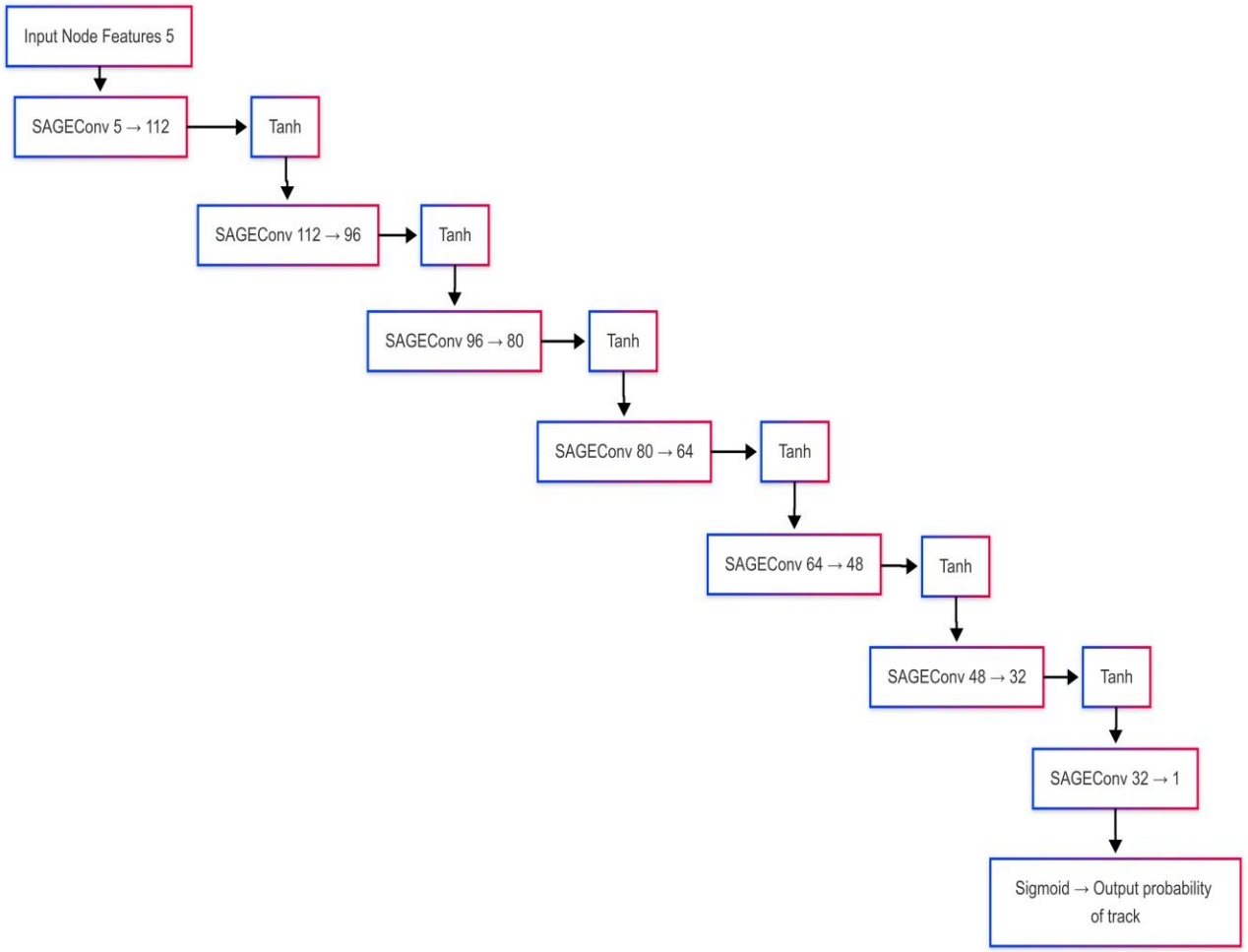
DATA CLEANING	NODE FEATURES	EDGE CONSTRUCTION
• Remove events with insufficient layer coverage (less than 4 layers). • Drop rows with missing values	• Hit positions • Uncertainties • Energy deposition • Layer information	• Create edges by connecting hits in consecutive layers • Calculate edge features such as angular differences between hits



Inference using SageCONV

The SageConv model is a scalable GNN architecture built using GraphSAGE convolution layers, specifically designed to process particle hit graphs across multiple detector layers.

Parameter	Value
Input Features	5 per node (e.g., x, z, PE, uncertainty, layer)
Output	1 (binary classification: node on track or not)
Hidden Size	112 neurons (initial layer)
Number of Layers	7 SAGEConv layers
Feature Reduction	16 units per layer



How it works

1.Input Graph

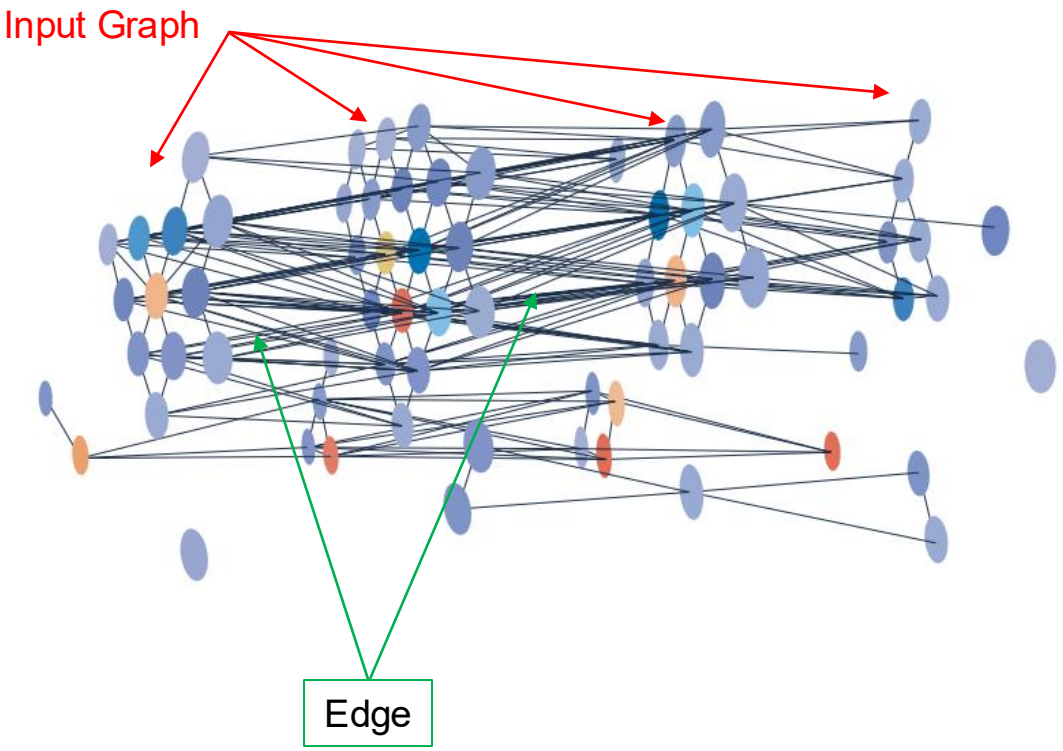
- Each node corresponds to a hit.
- Edges connect hits between consecutive detector layers.
- The graph includes geometric and signal-based features.

2.Layer Stack (7x SAGEConv)

- The first SAGEConv maps 5 input features → 112 hidden units.
- Each following layer **reduces** the hidden dimension by 16 units:
  - Layer 2: 112 → 96
  - Layer 3: 96 → 80
  - …
  - Layer 7: 32 → **1 output**

3.Activation Functions

- All layers (except the last) apply **Tanh** to introduce non-linearity.
- Final layer uses **Sigmoid** to output a value ∈ [0, 1] = **track probability**.



Why SAGEConv

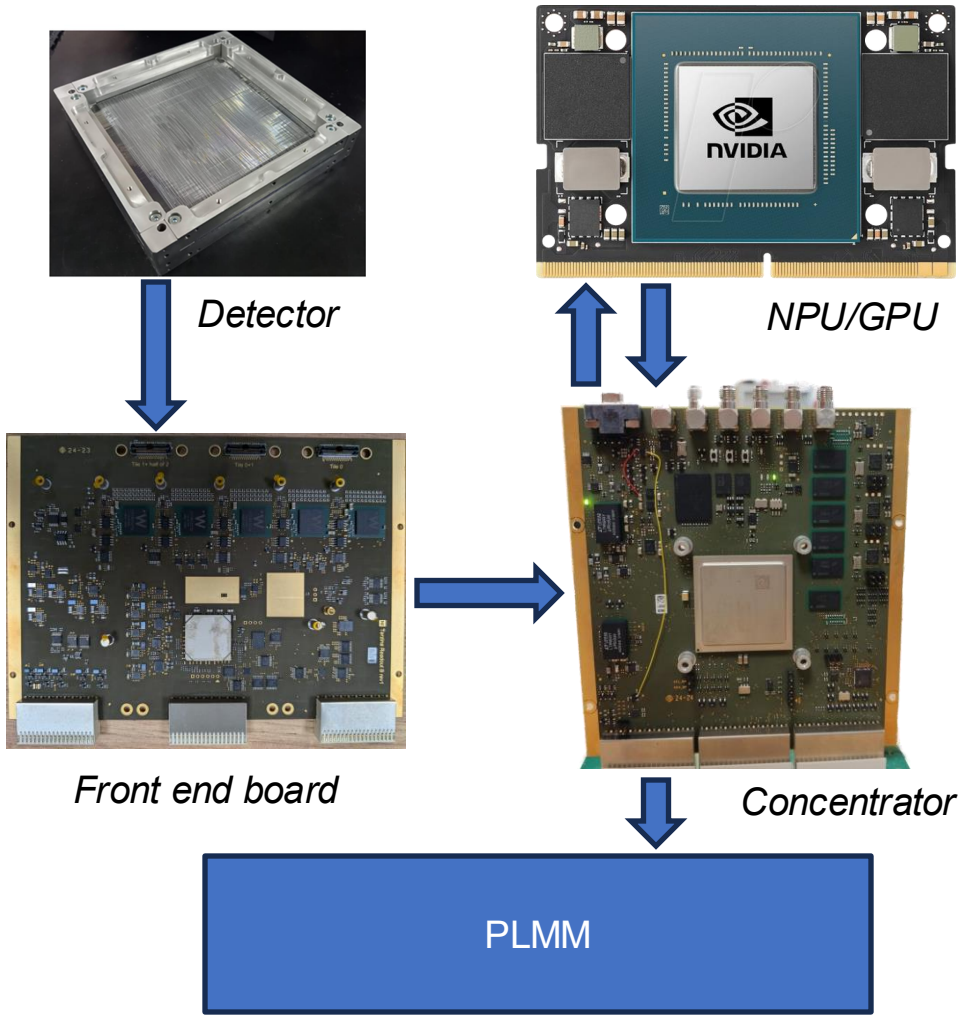
- It **aggregates neighbor features** efficiently using mean or learned functions.
- Perfect for graphs with **irregular topology**, like sparse particle events.
- Works in **inductive mode**, so it can generalize to unseen graphs.
- After training, each node's output represents the **probability of belonging to a particle track**.

Hardware Implementation Strategy

System Architecture

- **Hybrid Solution**: FPGA + NPU
  A mixed architecture combines a Xilinx Zynq Ultrascale+ FPGA with an NVIDIA Jetson Orin Nano to optimize the execution of the GNN model.

- **Functional Roles**:
  - **FPGA (Zynq Ultrascale+)** handles:
    - Front-end data acquisition
    - Preprocessing (e.g., cluster building, filtering)
    - Real-time data preparation and formatting
  - **NPU (Jetson Orin Nano)** performs:
    - GNN inference (GraphSAGE model)
    - Track reconstruction
    - Data packaging for acquisition PC



Challenges in Full GNN Implementation on FPGA

Architectural Constraints

- **Non-pipelined Design**: The structure of the SAGEConv-based GNN is not easily pipelined, making it unsuitable for direct implementation on pure FPGA without significant performance penalties.
- **Resource Usage**: The complexity of Graph Neural Networks requires a large number of logic gates and memory blocks, which scales poorly with the number of nodes and channels in the graph

Performance Bottlenecks

- **Sequential Computation**
  Unlike CNNs, GNNs require message passing and aggregation steps that are inherently sequential, limiting the effectiveness of parallel execution on FPGA.
- **Edge Feature Handling**
  The GNN model processes not only node features but also edge features—an added challenge for logic-based hardware without large shared memory and flexible control logic.

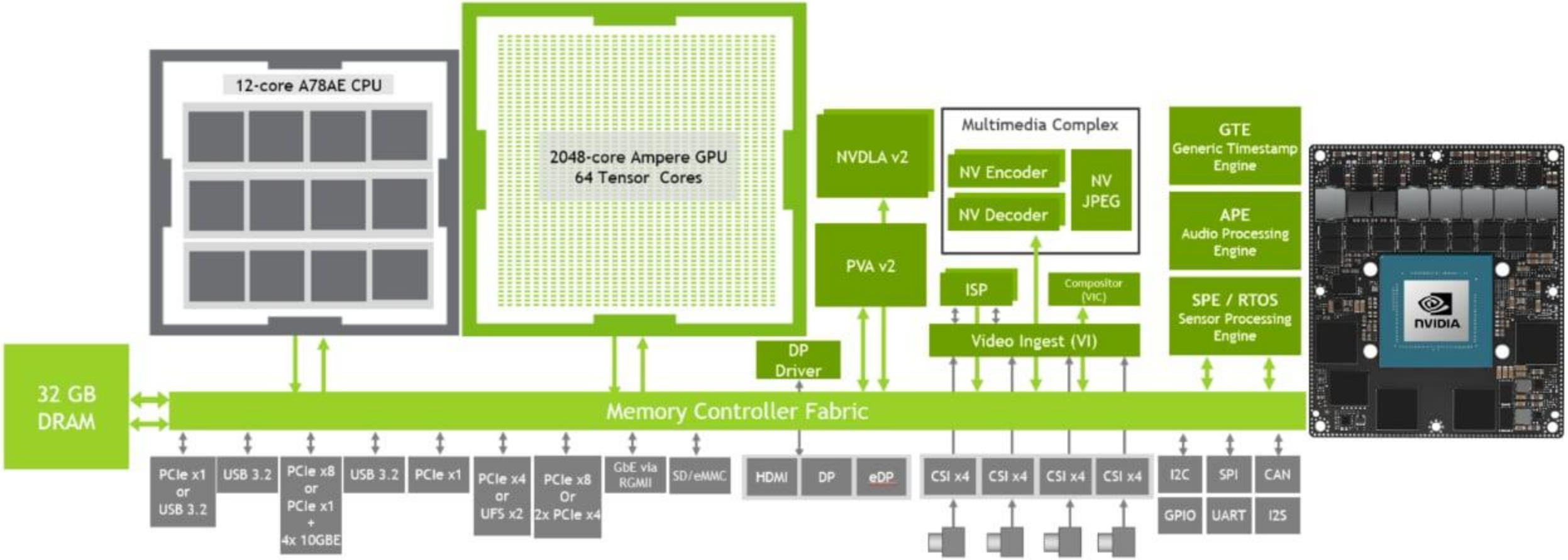
Challenges in Full GNN Implementation on FPGA

Toolchain Limitations

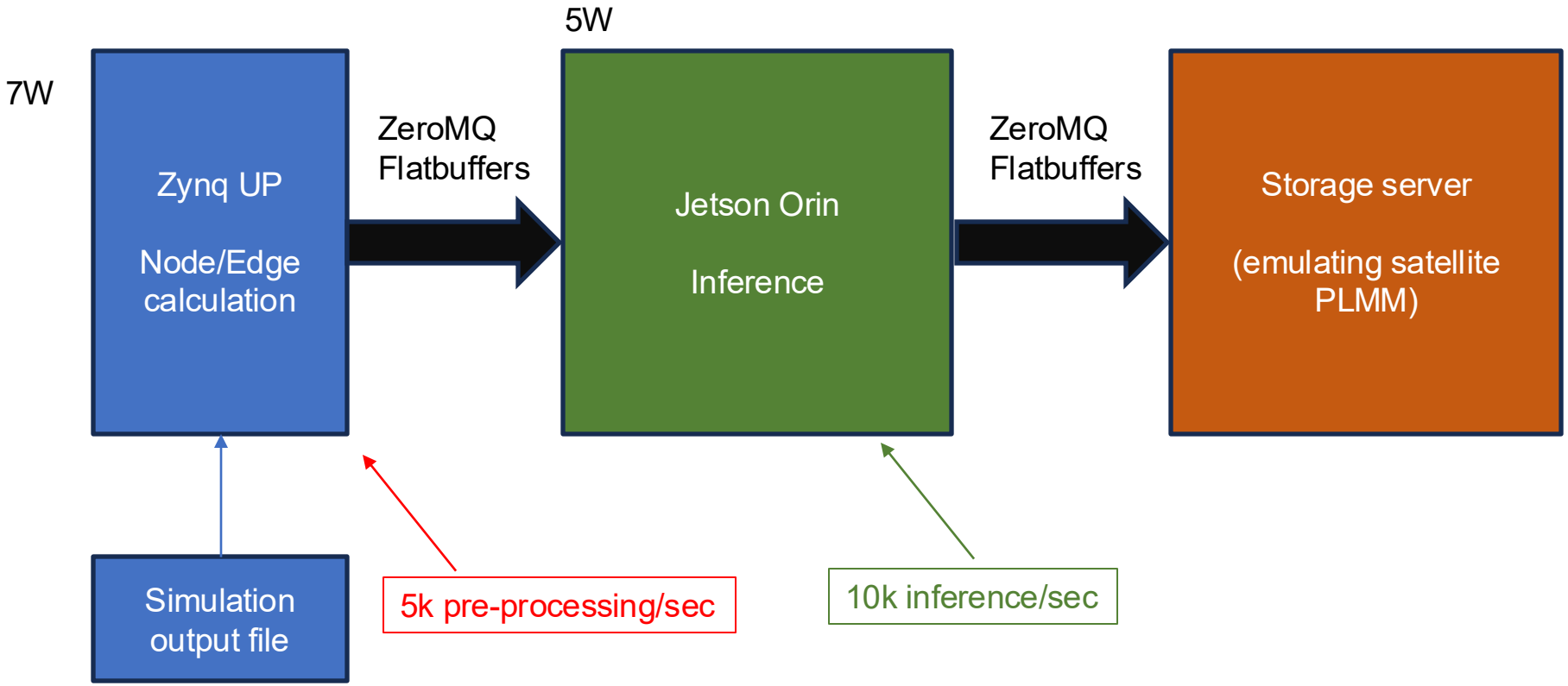
Existing tools like Vitis AI and hls4ml are not fully optimized for implementing complex GNN layers, especially when dynamic graph structures and attention mechanisms are involved

Due to the limitations above, a hybrid architecture was chosen: FPGA (Zynq Ultrascale+) for preprocessing, and NPU (Jetson Orin Nano) for inference, achieving high throughput and low power consumption

NVIDIA Jetson Orin Architecture



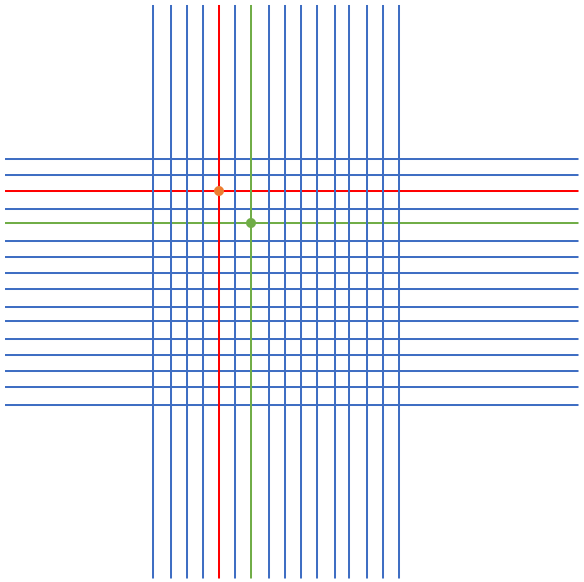
Inference system: Performance



Alternative Architecture – 2D CNN for Particle Tracking

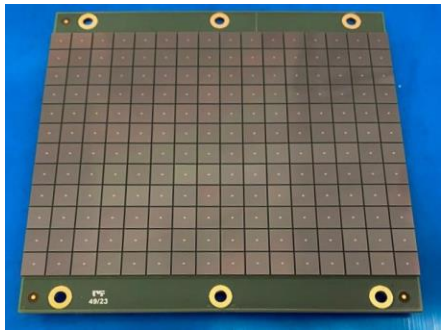
Concept:

- Replace the graph-based approach with a 2D convolutional neural network (CNN) that processes detector data as image-like arrays.
- Each detector layer is treated as a 2D plane of pixels: (N x N) matrix.



Input Representation:

- A 4-channel image, where each channel corresponds to one detector layer.
- Active pixels represent either:
  - Fiber intersection over threshold, or
  - Actual pixel hits (in pixelated detectors).



Advantages:

- Compatible with HLS4ML, allowing direct synthesis on FPGA.
- Standard CNN architectures are simpler to deploy and optimize on embedded systems.

2D CNN for Particle Tracking

CNN Structure Description

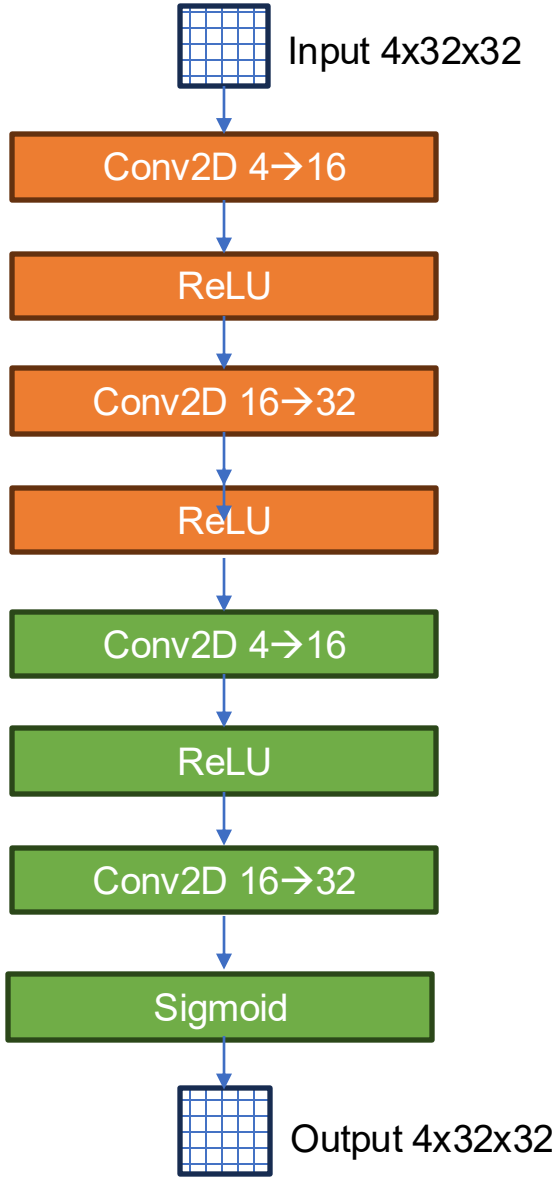
- The CNN is designed to process 4 detector layers simultaneously, each treated as a 32×32 pixel image.
- The model follows a **U-shaped encoder-decoder** pattern (though shallow), typical of segmentation tasks:

Encoder:

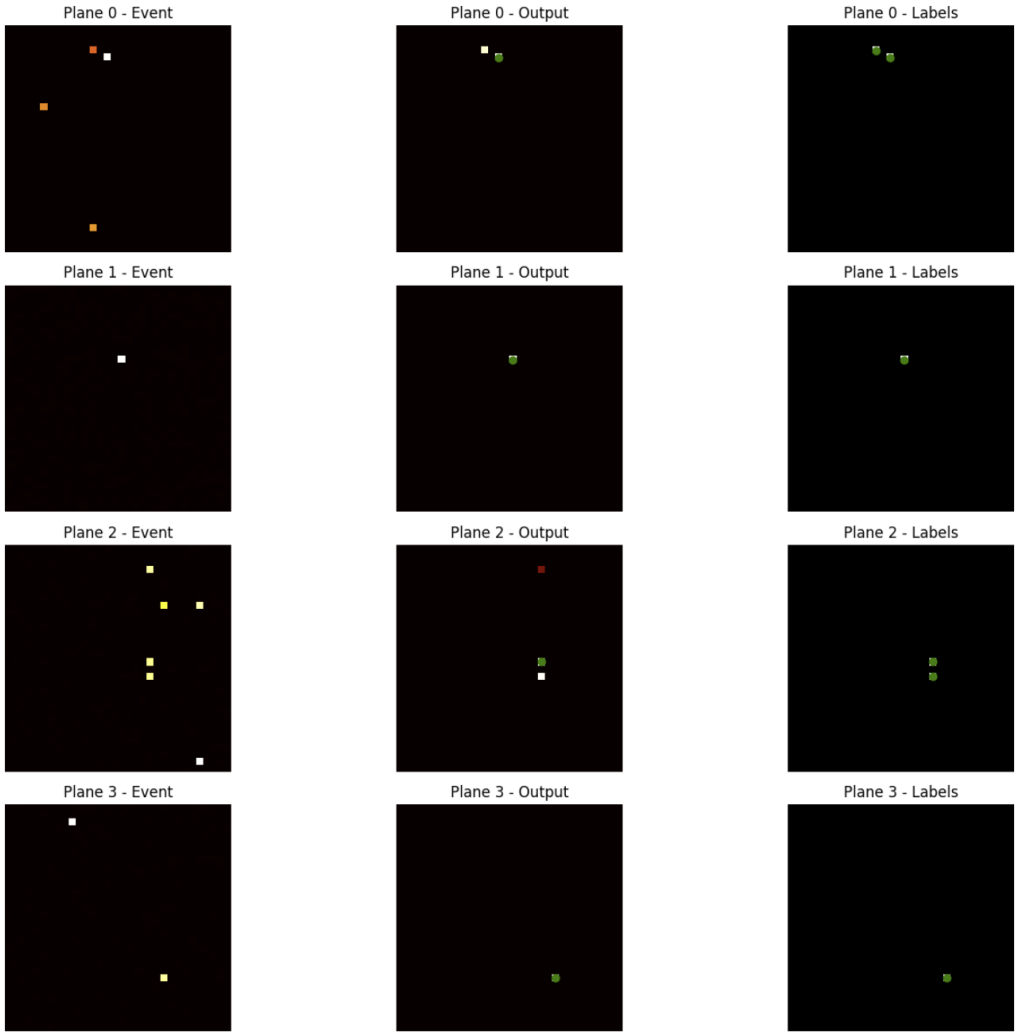
1. **Conv2D (4 → 16 channels)**
   Captures local patterns across each layer.
2. **ReLU Activation**
3. **Conv2D (16 → 32 channels)**
   Learns higher-level features combining all layers.
4. **ReLU Activation**

Decoder:

1. **Conv2D (32 → 16 channels)**
   Begins reconstruction to match the input resolution.
2. **ReLU Activation**
3. **Conv2D (16 → 4 channels)**
4. **Output**: 4-channel mask with sigmoid-activated values indicating track probabilities.

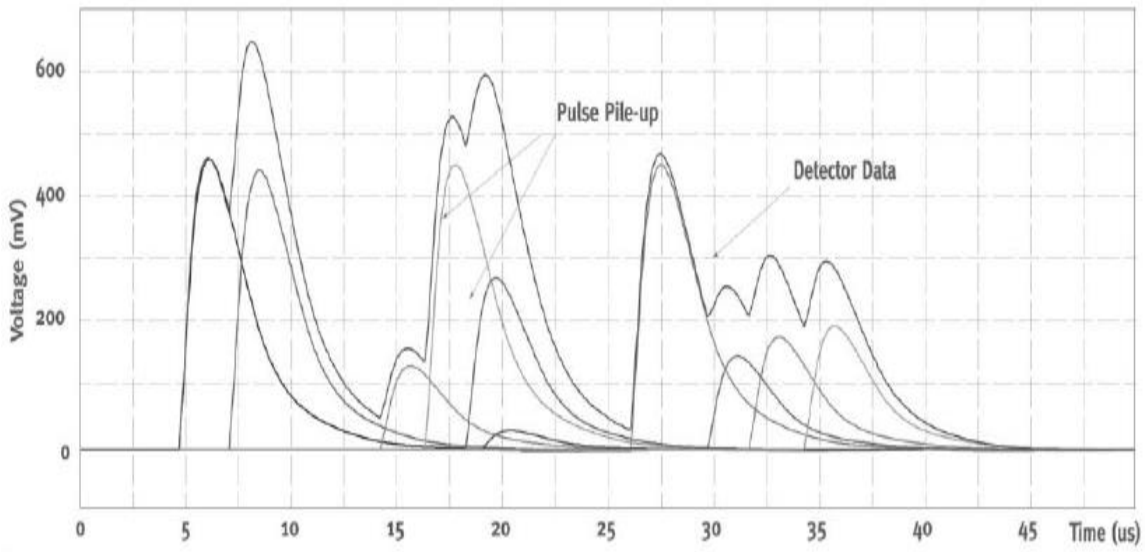


Alternative Architecture – 2D CNN for Particle Tracking



LEGIMAC: What is Pile-up

- Pile-up occurs when multiple events happen too close in time for the detector to resolve them as separate.
- Instead of recording distinct pulses, the system sees a single distorted or merged signal.



Challenge	Effect
Energy reconstruction	The total signal is **not a linear sum** of the two → energy is **overestimated**
Timing resolution	The peak shifts → wrong estimation of arrival time
Counting accuracy	You may count **one event instead of two**, especially if one is low energy
Pulse shape distortion	Makes it harder to distinguish real signals from noise or background

LEGIMAC: What is Pile-up

- LEGIMaC aims to detect ultra-low energy gamma events in space.
- These are frequent and closely spaced, especially during bursts or cosmic showers.
- Pile-up dramatically reduces: Detection sensitivity, Spectral resolution, Trigger efficiency

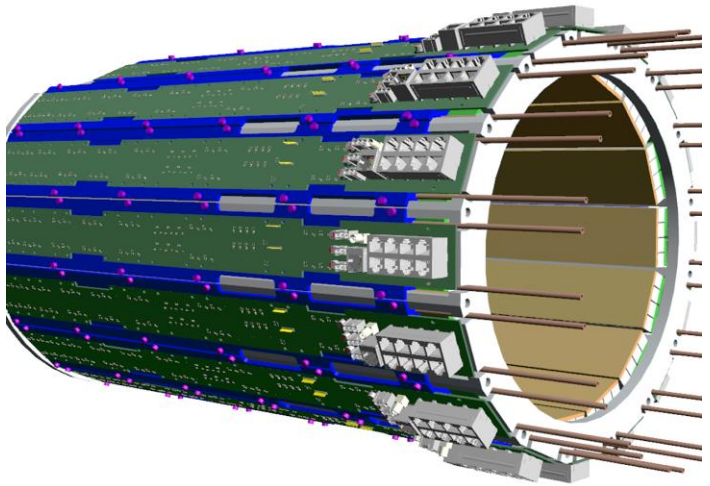
Use **machine learning**, especially neural networks, to:

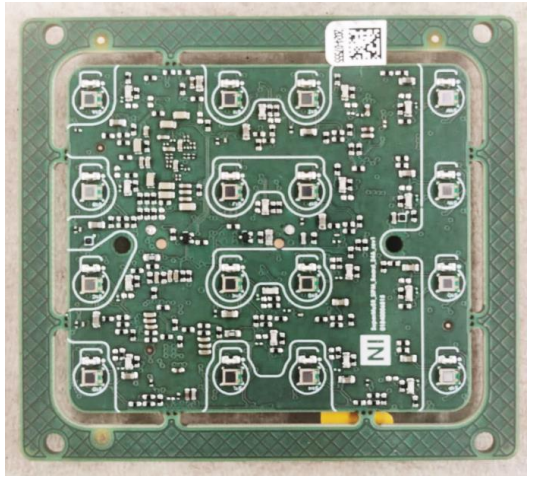
- Deconvolve overlapping signals
- Identify and **separate individual events**
- Reconstruct true amplitude and timing of each pulse

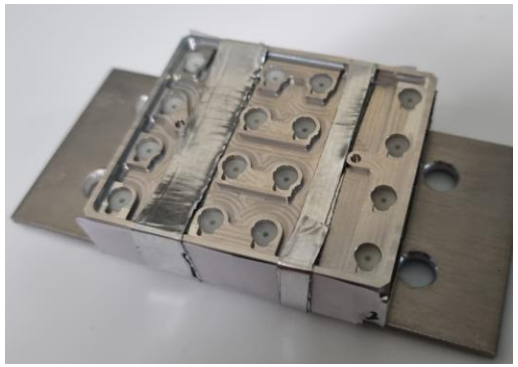
SuperMuSR Detector at ISIS

We use data from the **SuperMuSR experiment** as a **realistic benchmark**:

- **Setup**: Muon spectrometer at the **ISIS pulsed neutron and muon source**.
- **Detector**: Fast scintillators coupled to **SiPMs (Silicon Photomultipliers)**.
- **Environment**: Muons arrive in **well-defined bursts** (up to MHz), producing rapid chains of interactions in the sample.





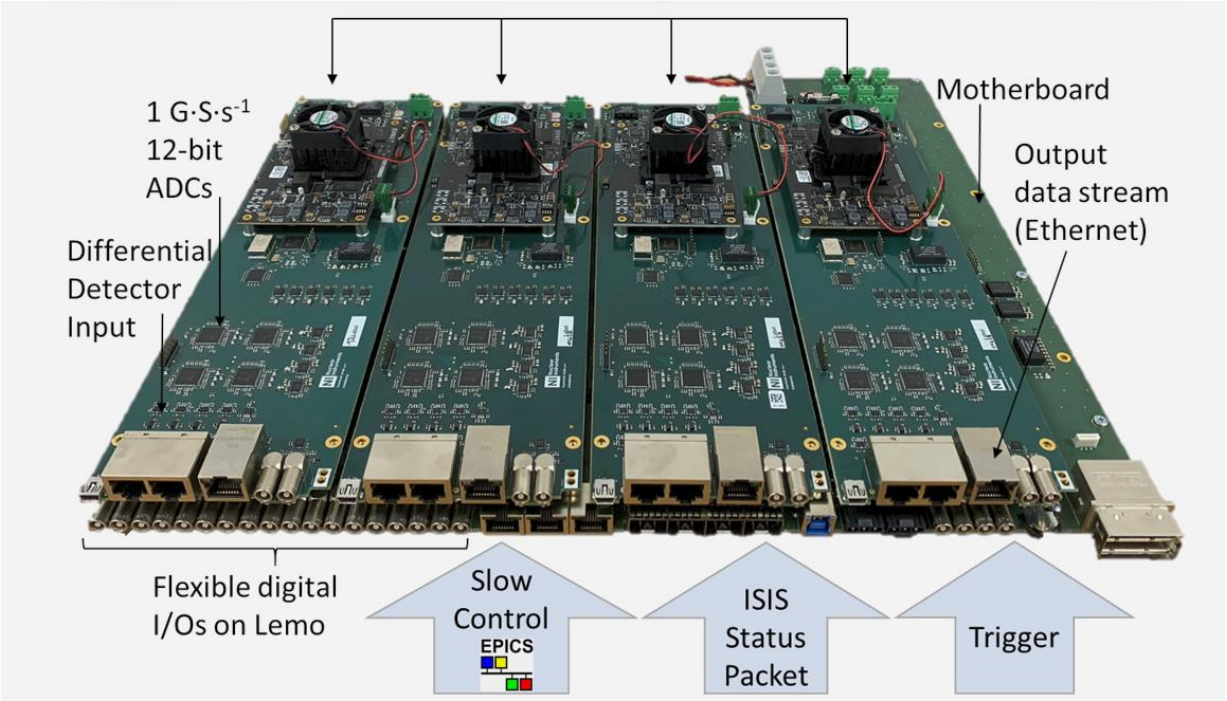


Why it's relevant:

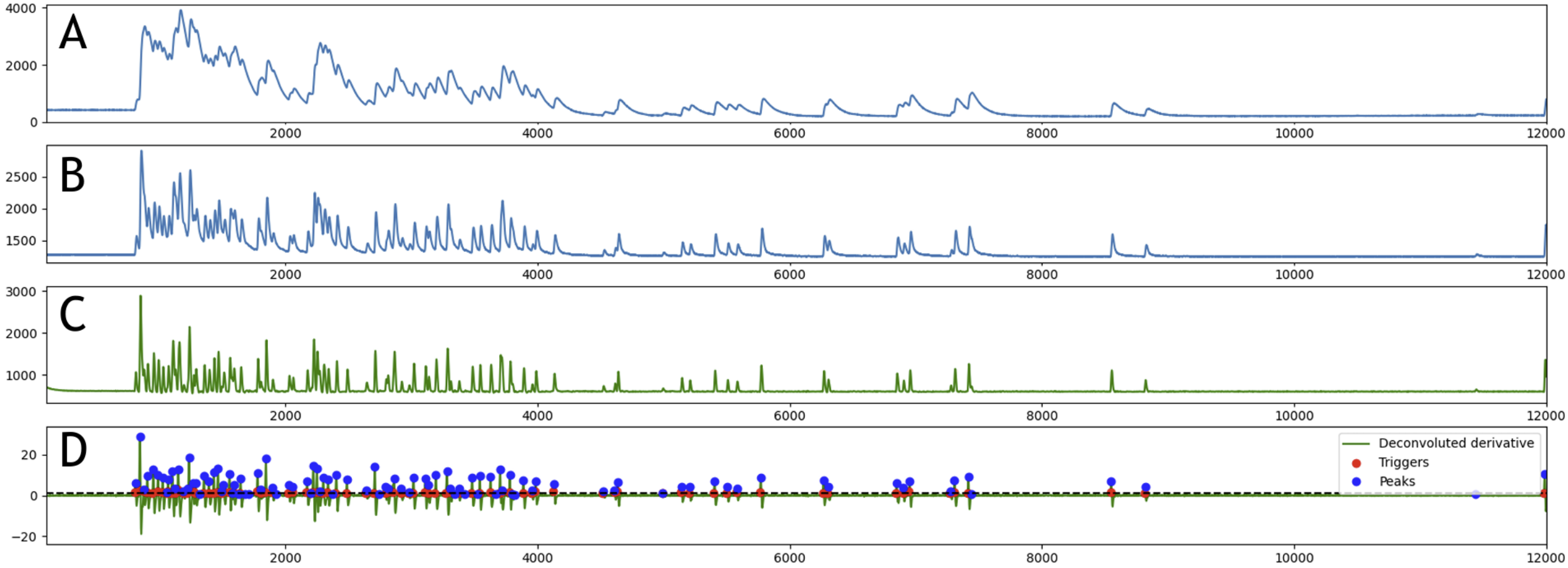
- Mimics the **burst-like, impulsive nature** of space radiation LEGIMaC targets.
- Provides a **rich dataset** with real pile-up cases to train and validate algorithms.

Hardware platform

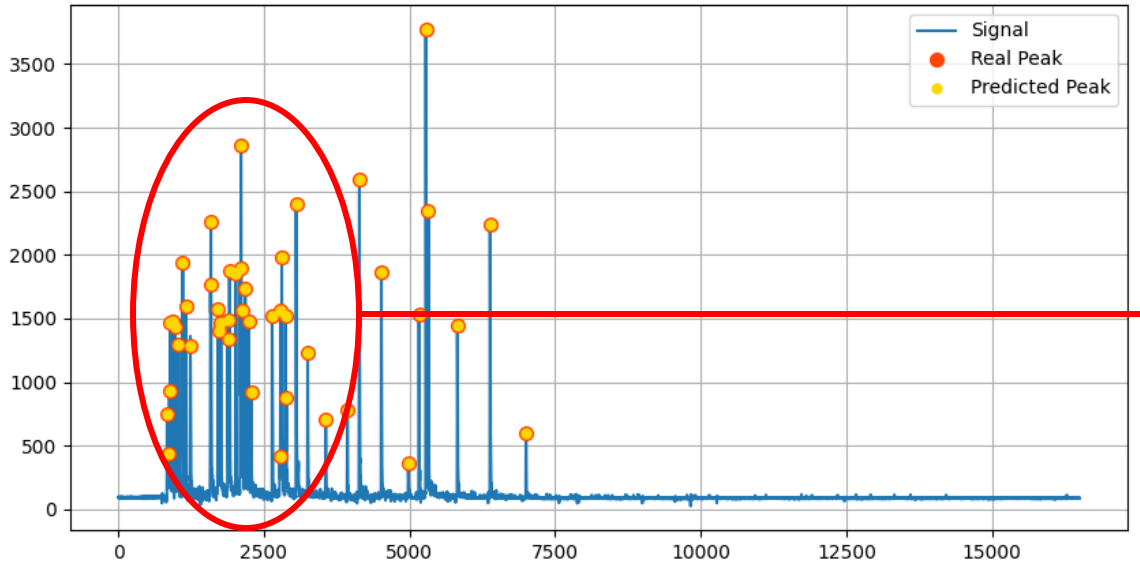
- Digitizer: Nuclear Instruments DAQ121
- 4x FPGA: Zynq Ultrascale + XCZU7EV
- Operating clock frequency: 250 MHz
- Operating sample rate: 1 GSPS
- Channels per digitizer: 32
- Integrated 4x4 cores ARM64 CPU with EPICS and Kafka directly running on the device
- 4 Gbps connectivity on copper
- 10 Gbps on fiber
- 32 + 4 LVDS
- 24 x TTL Lemo I/O
- Integrated LV generator
- Integrated BIAS generator up to 80V

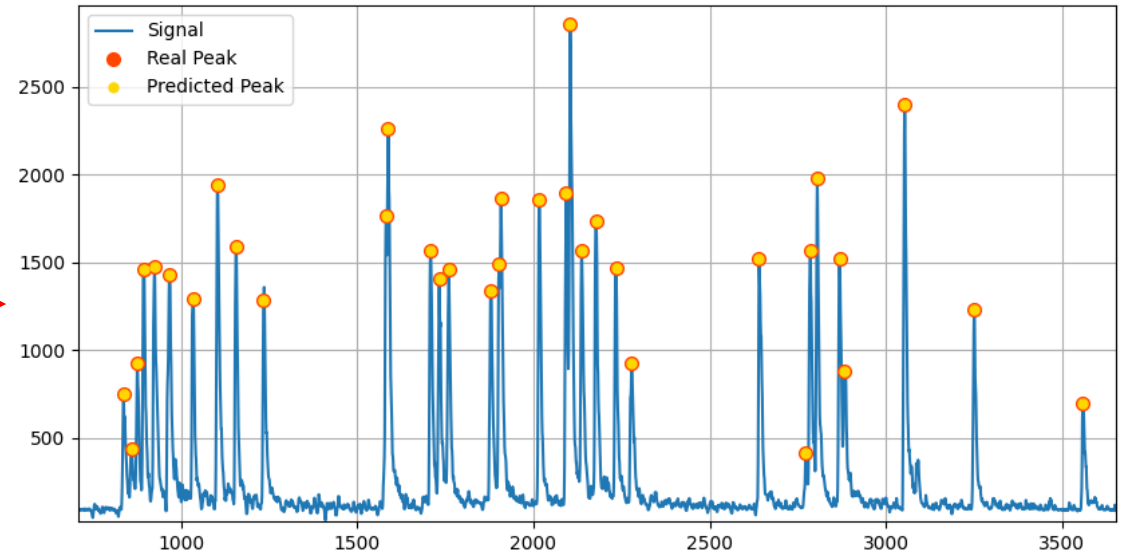


Deconvolution algorithm



CNN-1D

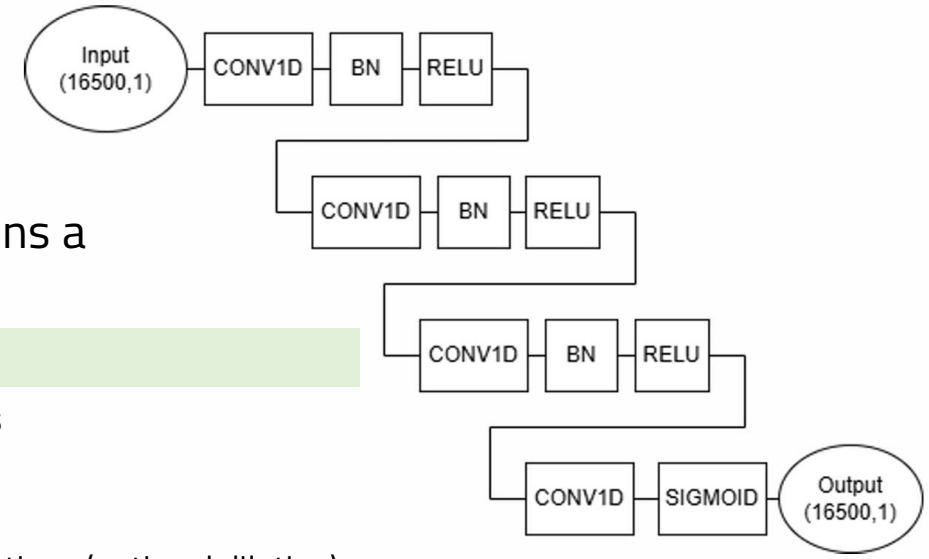

Signal
Real Peak
Predicted Peak
0
500
1000
1500
2000
2500
3000
3500
0
2500
5000
7500
10000
12500
15000


Signal
Real Peak
Predicted Peak
500
1000
1500
2000
2500
1000
1500
2000
2500
3000
3500

CNN-1D

Model Objective

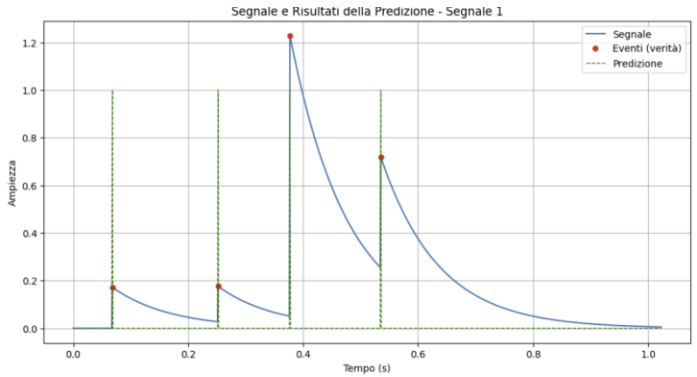
- Analyze **1D waveforms** of length 1024 samples.
- Detect and separate **overlapping pulses** (pile-up).
- Output a **probability mask** over time: where each sample likely contains a valid signal.



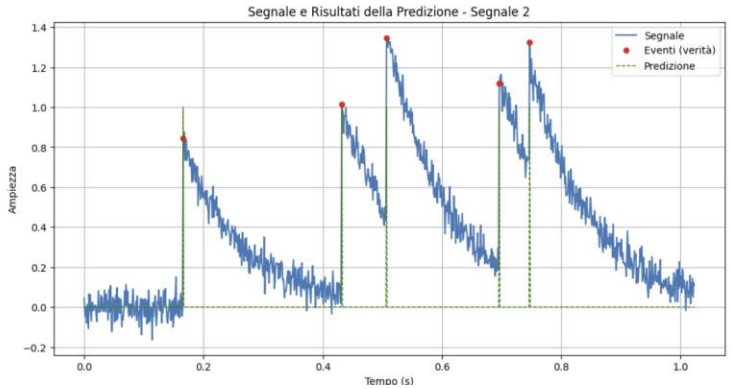
Layer	Type	Description
conv1	Conv1D(4)	5-point convolution, 4 filters — learns edge & slope patterns
relu1	ReLU	Introduces non-linearity with reduced bit-width
conv2	Conv1D(8)	3-point convolution, 8 filters — captures short-range interactions (optional dilation)
relu2	ReLU	
conv3	Conv1D(16)	Deeper layer with 16 filters — models overlapping or decaying tails
relu3	ReLU	
conv4	Conv1D(8)	Reduces back to 8 filters — focuses on compressed representation
relu4	ReLU	
output	Conv1D(1)	Final 1x1 convolution — returns per-sample score
sigmoid	Sigmoid	Converts score to [0, 1] probability (signal present vs. noise)

Training

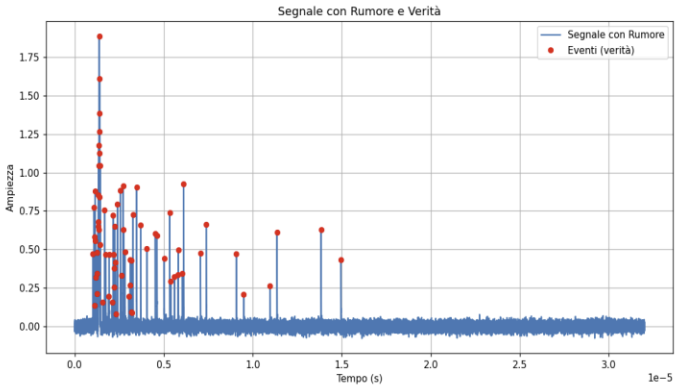
Modelized data



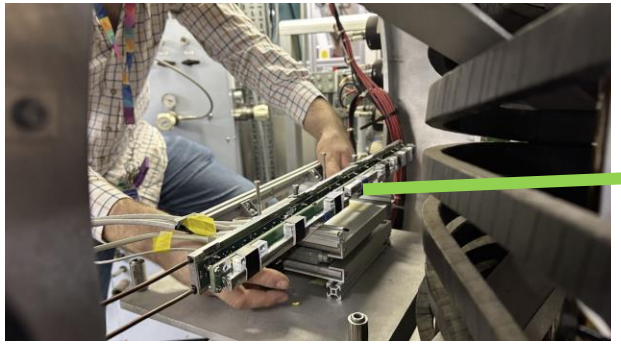
Noise

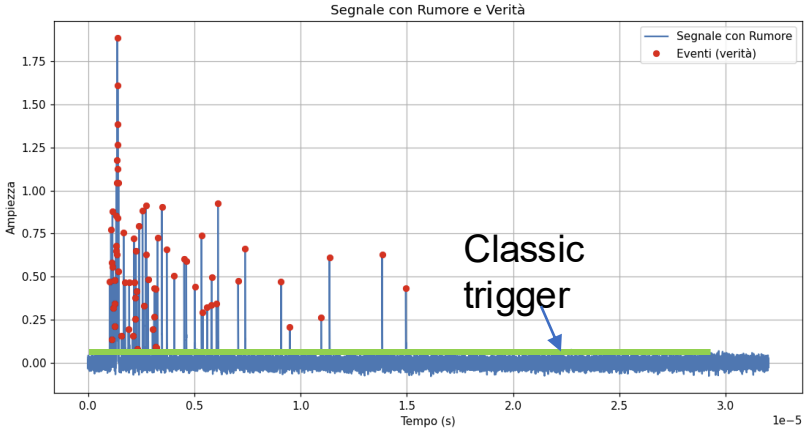


Time Distribution



Read Data



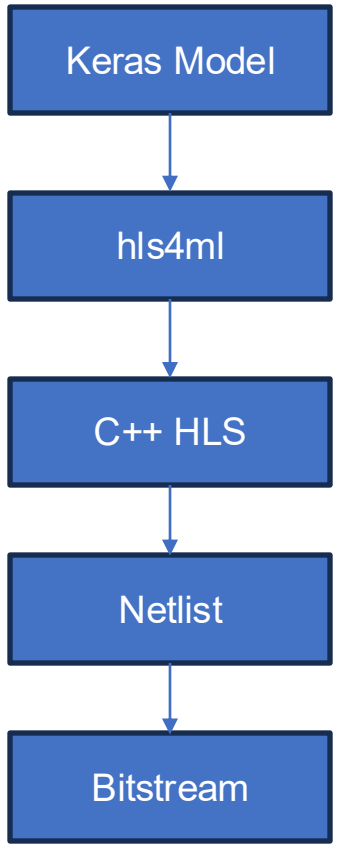


HLS4ML

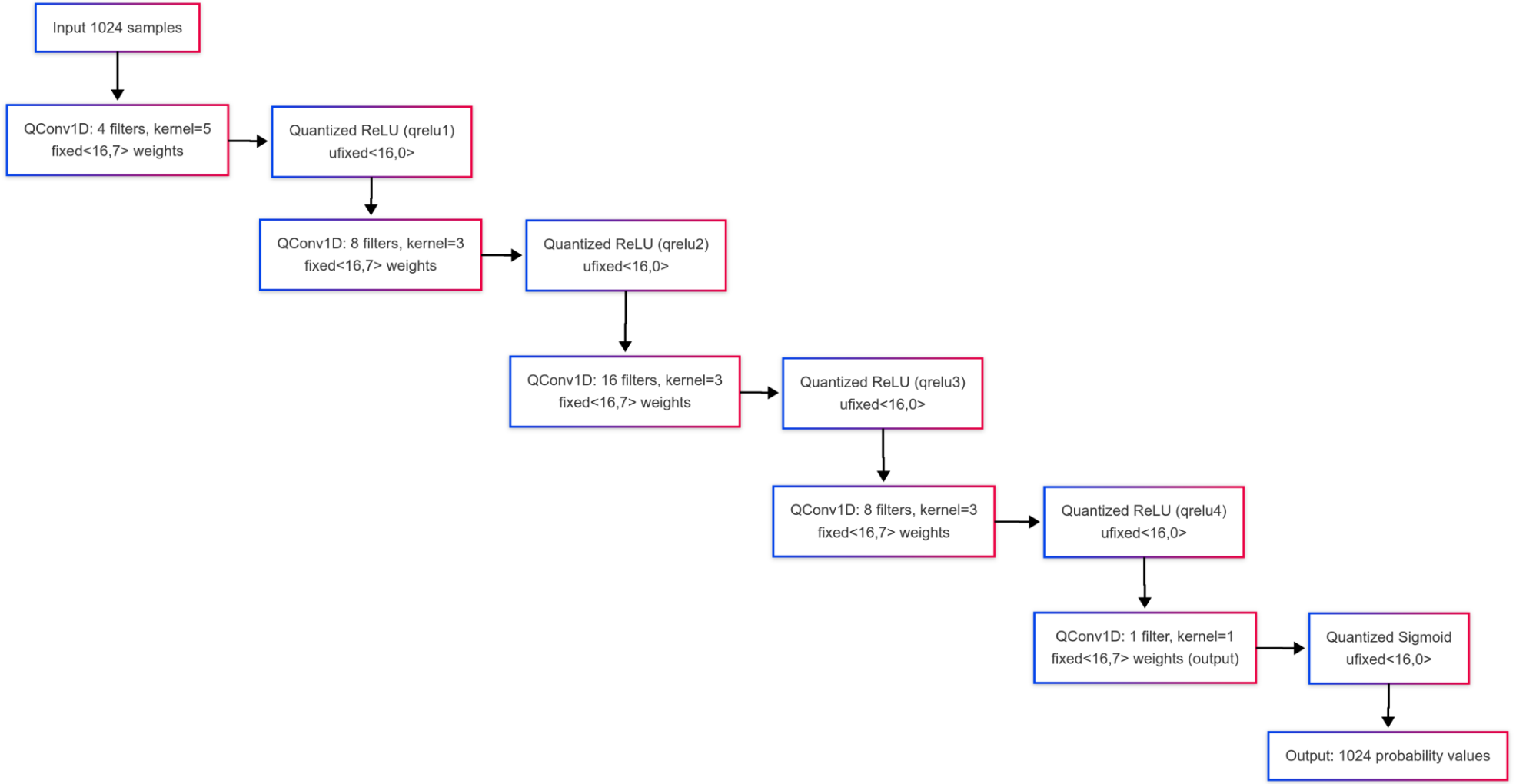
- **hls4ml** is an open-source tool that converts **trained neural networks** into **synthesizable firmware** for **FPGAs** and **ASICs**, using **High-Level Synthesis (HLS)**.
- It bridges the gap between **machine learning frameworks** (like Keras, PyTorch, QKeras) and **hardware description languages** (like VHDL or Verilog), enabling ML deployment on edge devices with limited power and latency constraints.
- Unlike GPU/CPU/NPU inference engines, **hls4ml transforms the model into a true hardware architecture** — not a software process
- A **custom, dataflow-pipelined architecture**, fully tailored to the neural network. Neural network becomes **a real circuit**, like a digital filter or FSM
- **No CPU cores, no external memory access, no generic ML accelerator** Inference happens **clock-by-clock**, at the speed of the FPGA fabric.
- **Fully pipelined**: One result every N clock cycles

LEGIMAC user case

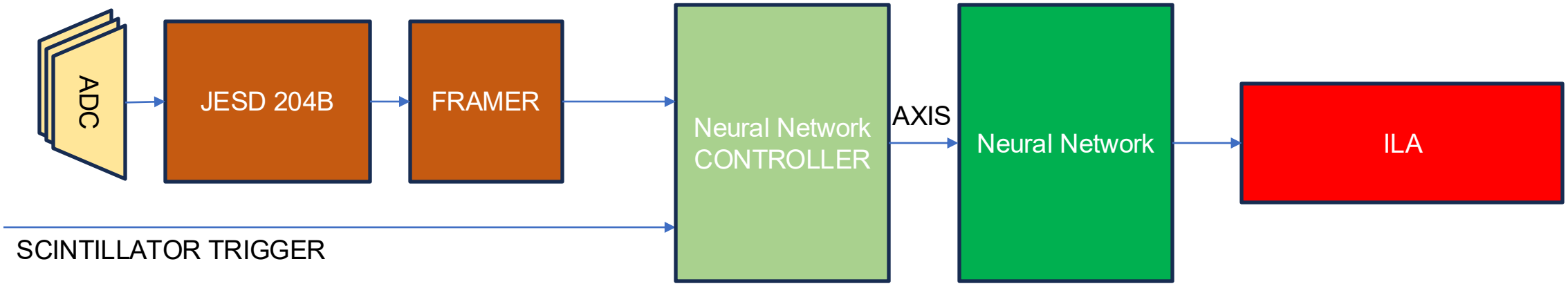
- Quantized with Qkeras
- Compiled via hls4ml → Vitis Backend
- Deployed as hardware logic inside a Xilinx Zynq Ultrascale+ FPGA

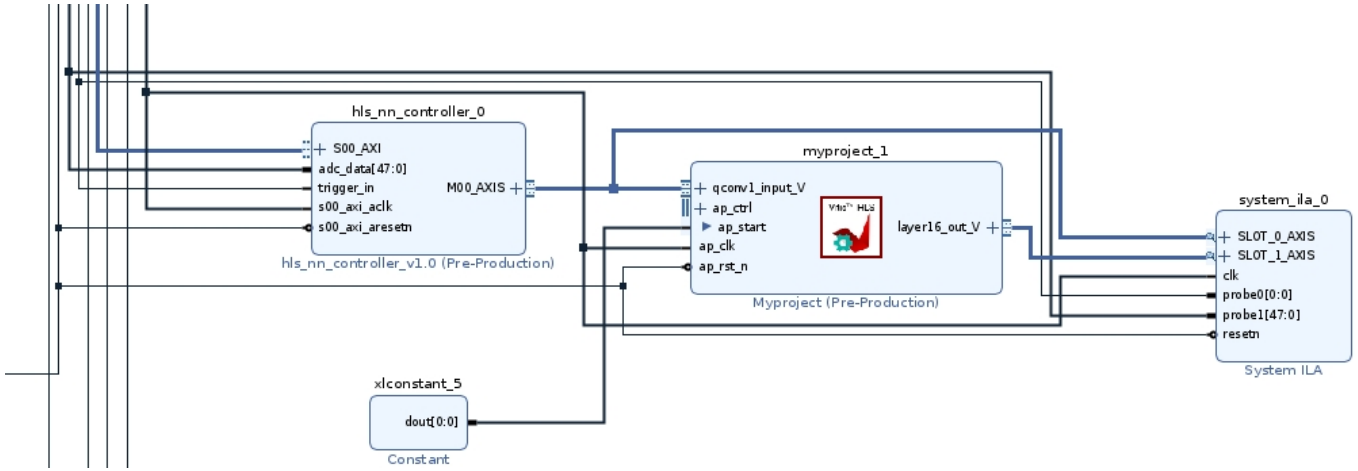


LEGIMAC HLS Implementation



LEGIMAC HLS Implementation





LEGIMAC HLS Implementation

Design Runs | IP Status | DRC | Methodology | Power | Timing | **Utilization**

Hierarchy

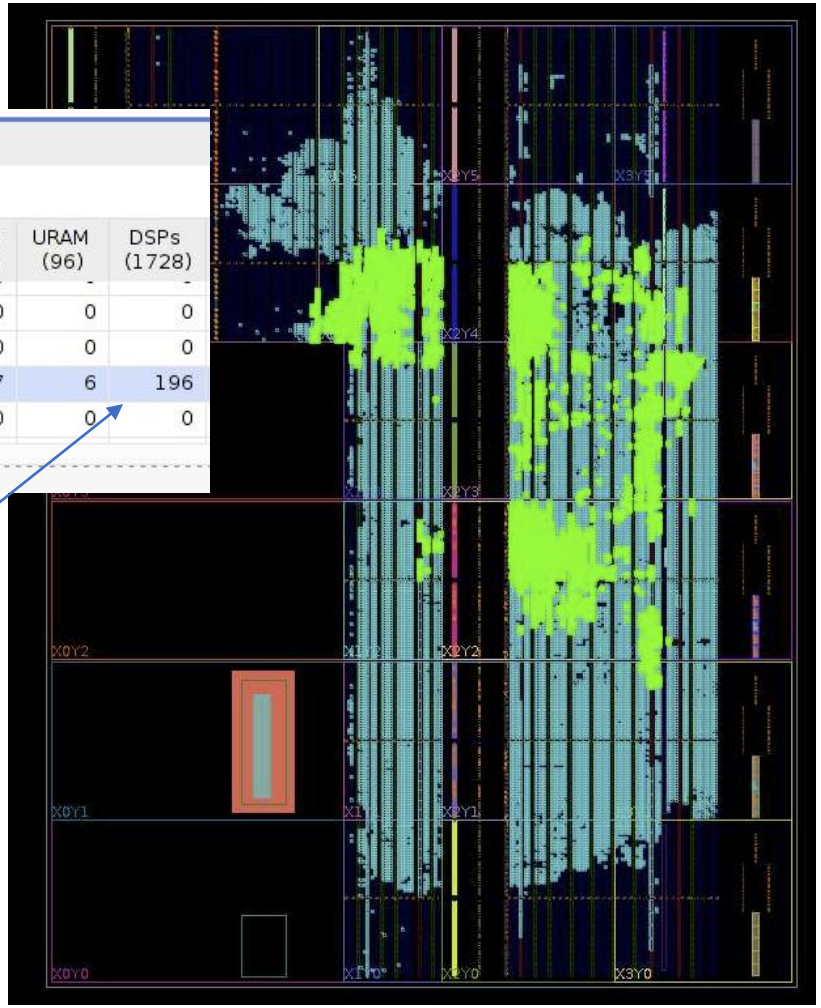
Name	CLB LUTs (230400)	CLB Registers (460800)	CARRY8 (28800)	F7 Muxes (115200)	F8 Muxes (57600)	CLB (28800)	LUT as Logic (230400)	LUT as Memory (101760)	Block RAM Tile (312)	URAM (96)	DSPs (1728)
jesd204_phy_2 (uz7ev_evcc_	569	929	4	2	0	177	569	0	0	0	0
jesd204_phy_3 (uz7ev_evcc_	569	929	4	2	0	169	569	0	0	0	0
myproject_1 (uz7ev_evcc_ba	7261	9119	622	0	0	1906	7261	0	57	6	196
nism_ad9234_adc_shuf_0 (u	227	494	0	64	32	77	227	0	0	0	0

Clock Frequency: 250 MHz

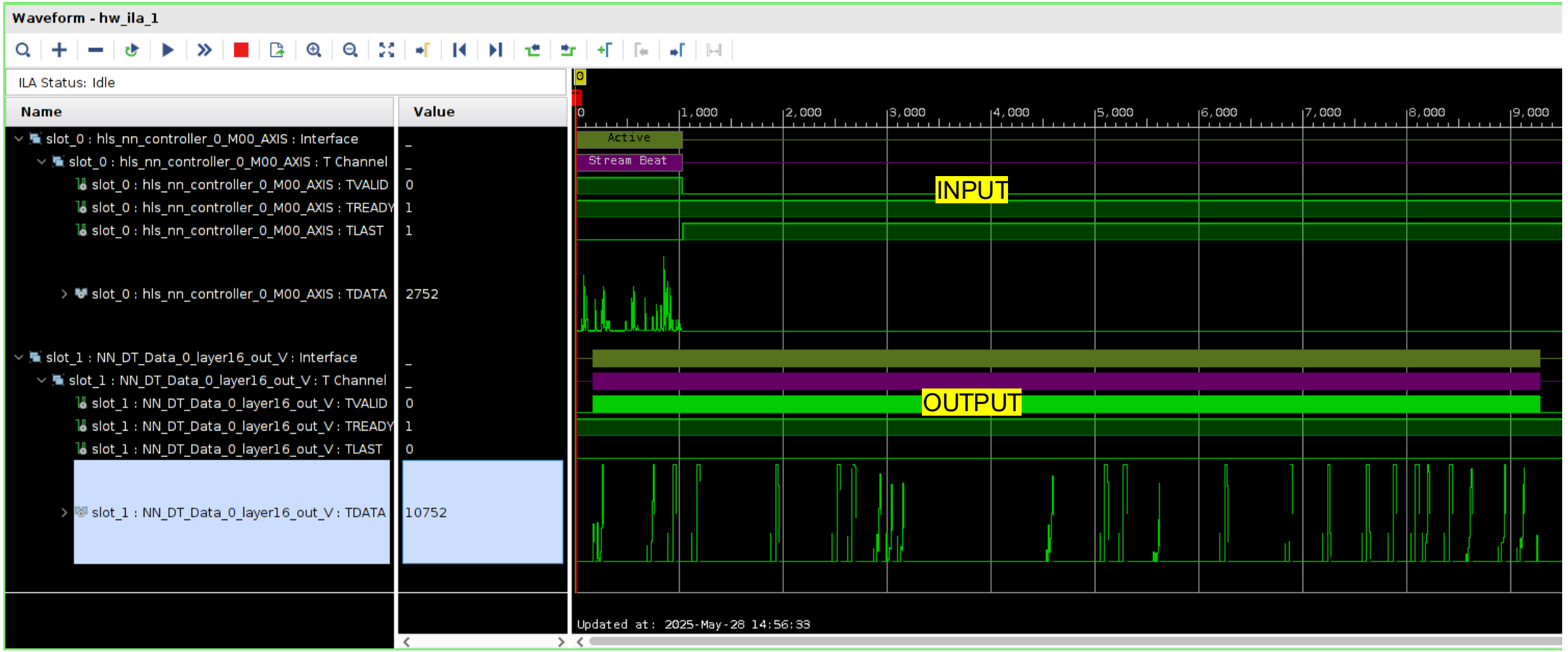
LUT 3.5%

RAM: 18.2%

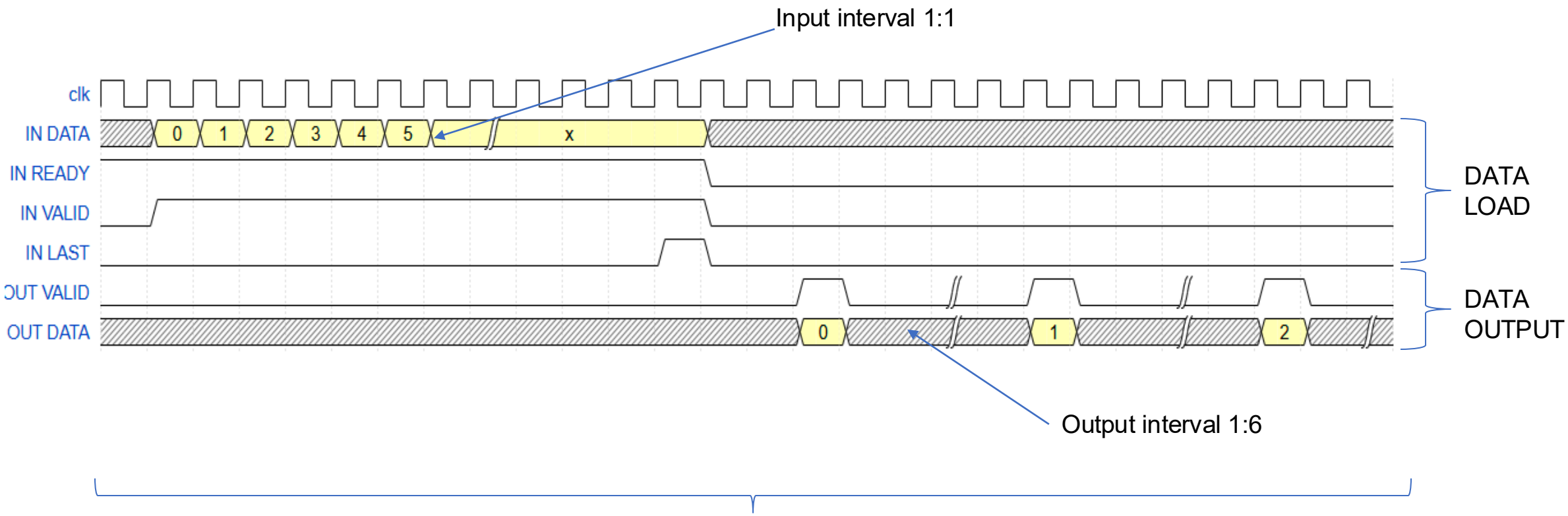
DSP: 11.3%



LEGIMAC HLS Output



LEGIMAC HLS Timing

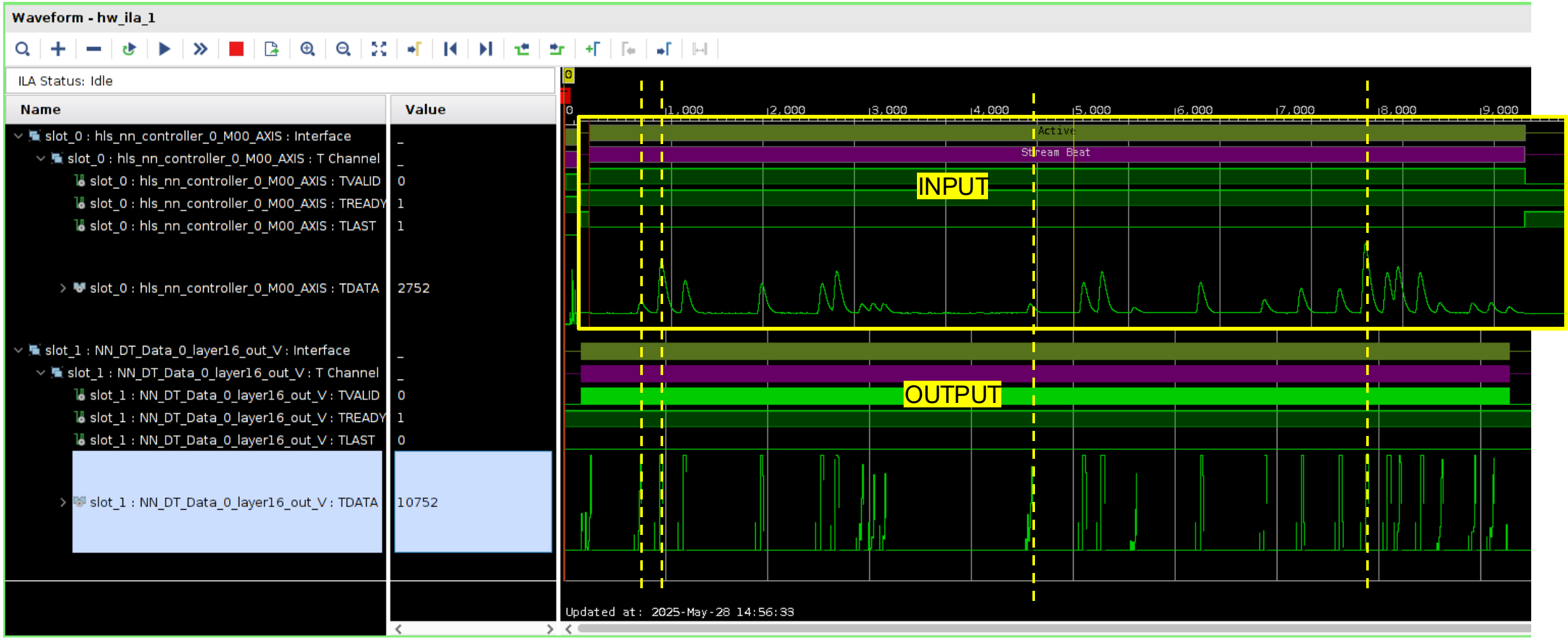


Total = 1024 + (1024 * 6) + overhead = 7500 clock cycles → 30us (clock 250MHz) = 34000 inference/s

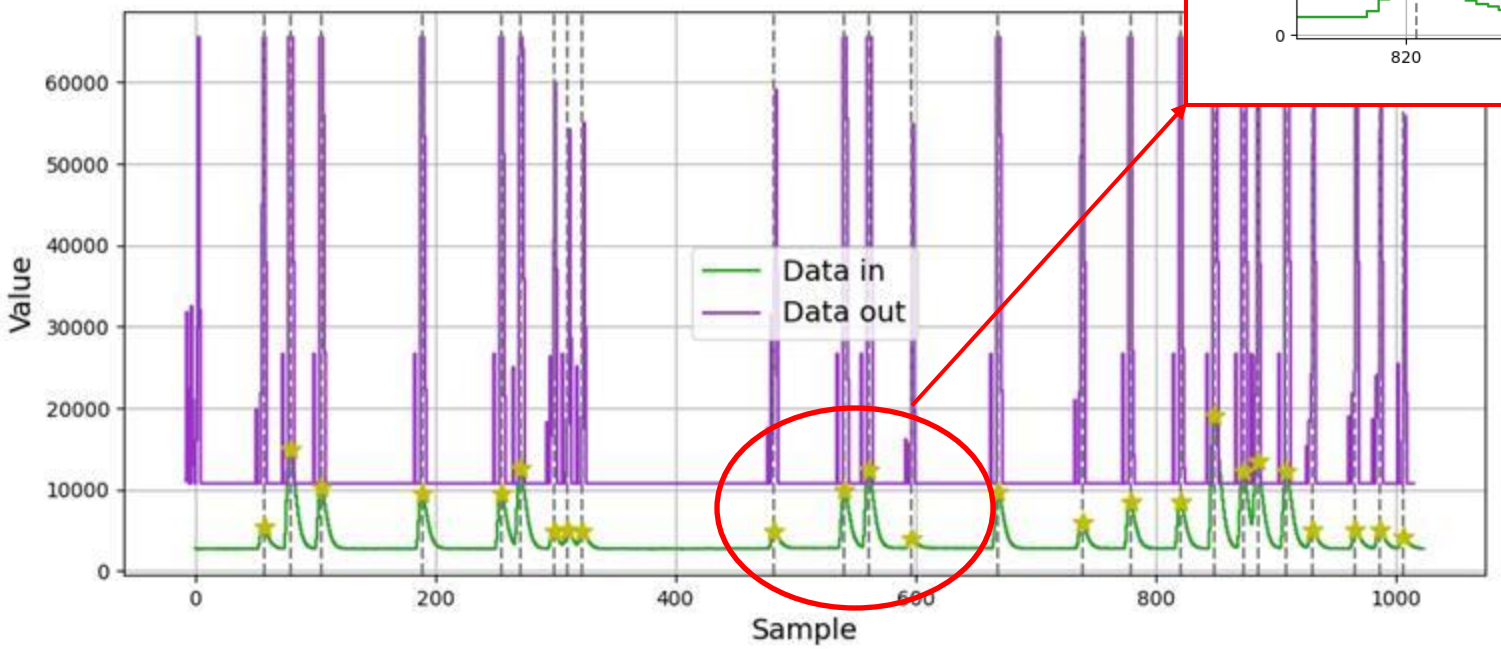
4 engine x FPGA = 135 k inference/s → 135Msamples/s

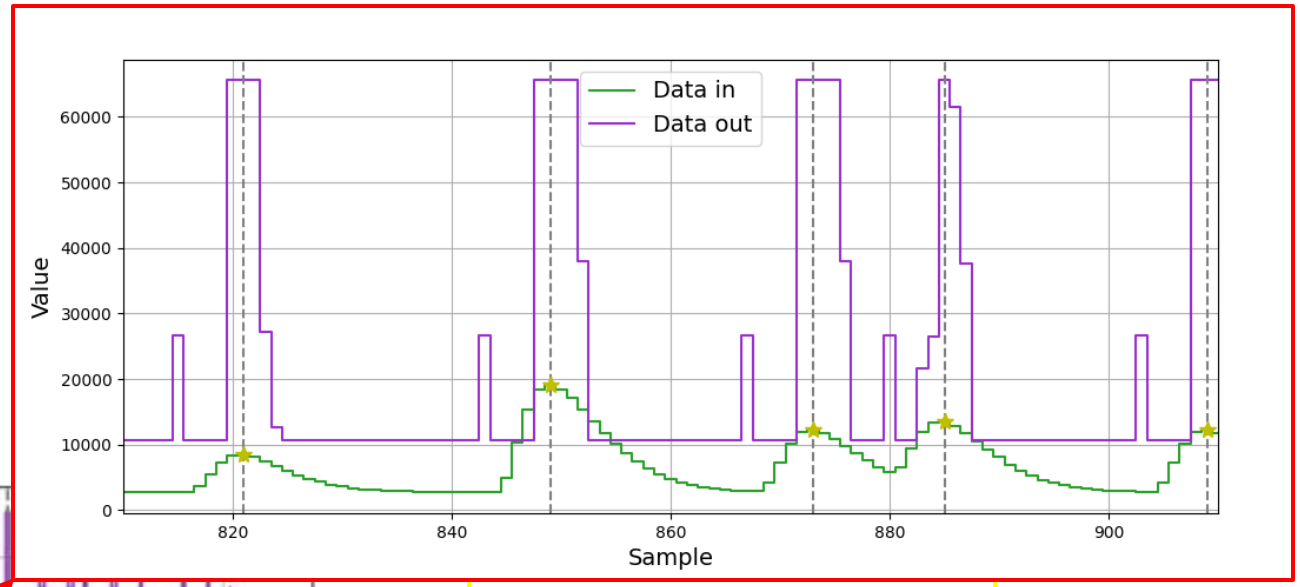
LEGIMAC HLS Output

Input is Time rescaled 1:6 to match output pipeline cycle

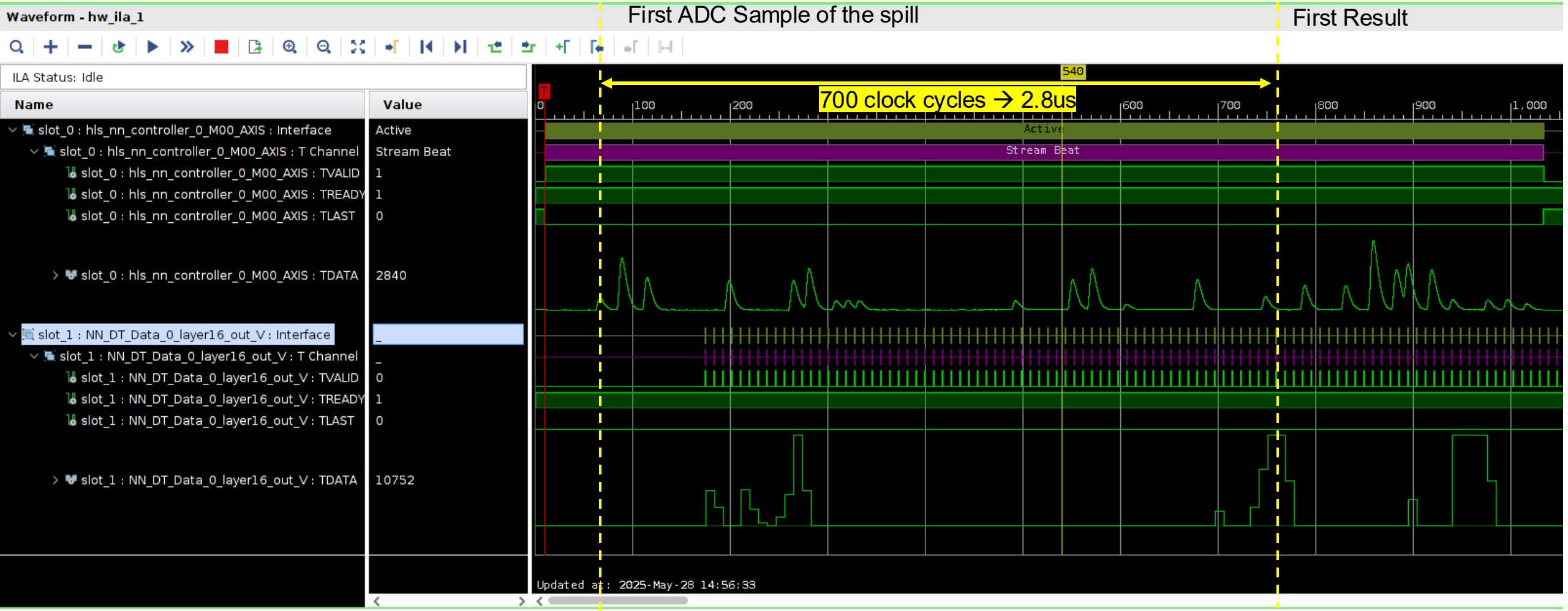


LEGIMAC Processed Output





LEGIMAC Realtime and low latency

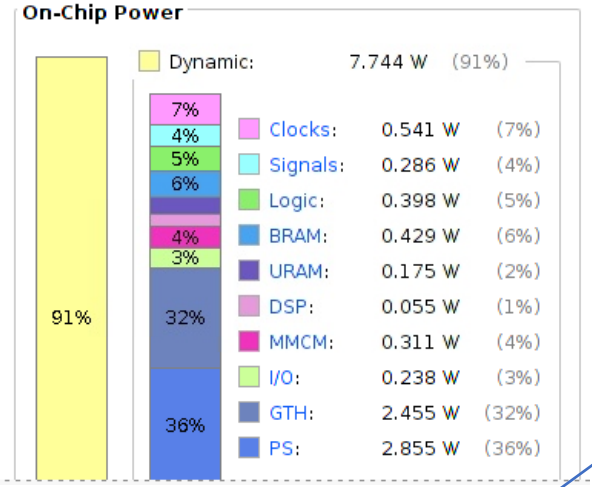


LEGIMAC HLS Implementation

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**8.542 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**36.7°C**
Thermal Margin:	63.3°C (45.1 W)
Effective θJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



330mW for 35000 inference/s!!!

Utilization	Name	Clocks (W)	Signals (W)	Data (W)	Clock Enable (W)	Set/Reset (W)	Logic (W)	BRAM (W)	URAM (W)	DSP (W)
7.744 W (91% of total)	systop									
7.649 W (90% of total)	bd (uz7ev_evcc_base_wrapper)	0.497	0.28	0.261	0.015	0.003	0.387	0.4	0.175	0.055
7.649 W (90% of total)	uz7ev_evcc_base_i (uz7ev_evcc_base	0.497	0.28	0.261	0.015	0.003	0.387	0.4	0.175	0.055
2.857 W (33% of total)	zynq_ultra_ps_e_0 (uz7ev_evcc_base_	<0.001	0.002	0.002	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.631 W (7% of total)	jesd204_phy_0 (uz7ev_evcc_base_jes	0.01	0.006	0.006	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.628 W (7% of total)	jesd204_phy_1 (uz7ev_evcc_base_jes	0.01	0.003	0.003	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.627 W (7% of total)	jesd204_phy_3 (uz7ev_evcc_base_jes	0.011	0.002	0.002	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.627 W (7% of total)	jesd204_phy_2 (uz7ev_evcc_base_jes	0.011	0.002	0.002	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.425 W (5% of total)	smartconnect_0 (uz7ev_evcc_base_s	0.09	0.102	0.091	0.01	<0.001	0.233	<0.001	<0.001	<0.001
0.346 W (4% of total)	myproject_1 (uz7ev_evcc_base_myprc	0.048	0.013	0.013	<0.001	<0.001	0.01	0.155	0.066	0.055
0.306 W (4% of total)	dgtz_adc120_0 (uz7ev_evcc_base_dg	0.054	0.018	0.018	<0.001	<0.001	0.02	0.116	<0.001	<0.001
0.28 W (3% of total)	axi_chip2chip_0 (uz7ev_evcc_base_a	0.011	0.001	0.001	<0.001	<0.001	0.001	0.001	<0.001	<0.001
0.191 W (2% of total)	supermusr_mp_0 (uz7ev_evcc_base_	0.056	0.021	0.019	0.001	0.001	0.019	0.095	<0.001	<0.001

Comparison with GPU

Model	RAM	Inference/s	Power
FPGA Implementation	257 kbytes	34000	0.35W
GTX4060	30 Mbytes / 8Gb	30000	100W
Jetson Orin Nano	30 Mbytes / 4Gb	11500	5W