CMS PHASE-2 · LEVEL-1 TRIGGER

FPGA-Aware Graph Neural Networks

for the CMS Phase-2 Muon Trigger

Co-designing model architecture and firmware — and making it reproducible with ARC, a contract-driven verification framework.

Santiago Folgueras · ICTEA – Universidad de Oviedo

III ICTEA Research Days

Universidad de Oviedo
INTREPID
Funded by the European Union
erc
European Research Council
Established by the European Commission

MOTIVATION

MODEL THREAD

Real-time reconstruction under HL-LHC conditions

40 MHz

BUNCH-CROSSING RATE

A new event every 25 ns — algorithms must be fully pipelined (target II = 1).

12.5 µs

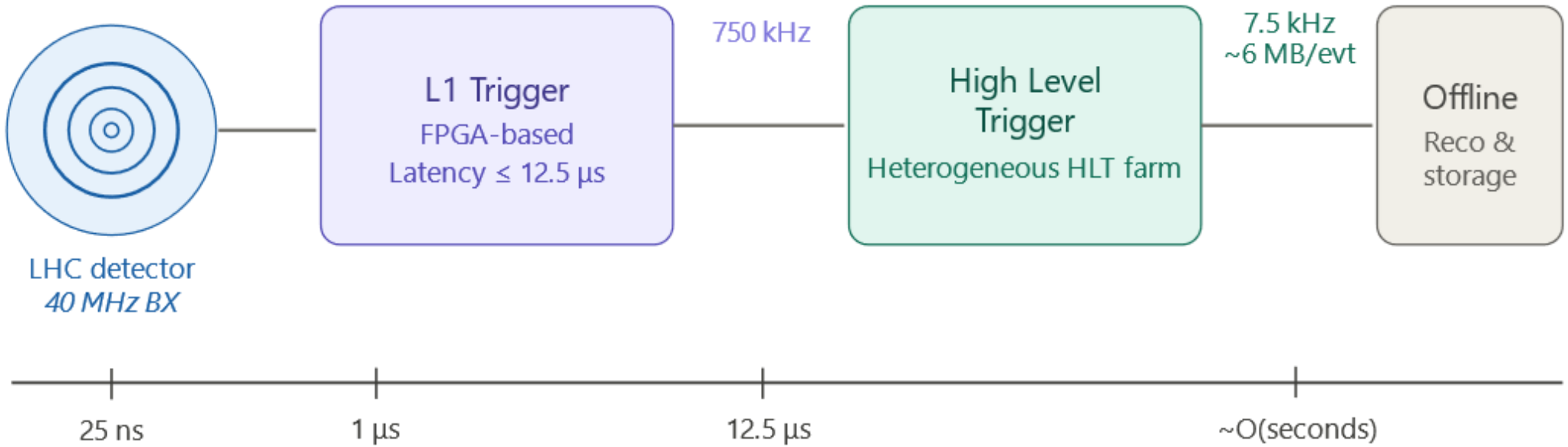
L1 LATENCY BUDGET

Total fixed window; each algorithm gets only a sub-µs slice to produce its result.

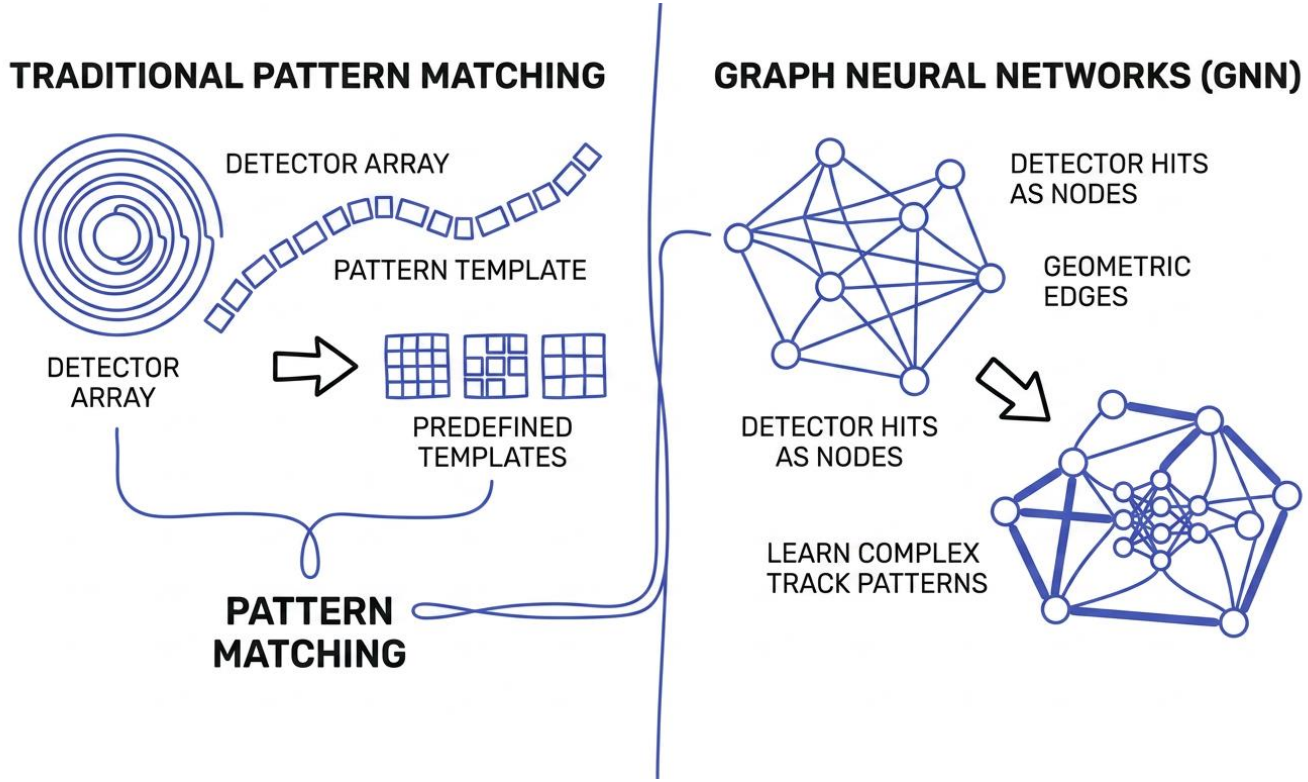
LUT · DSP · BRAM

FIXED RESOURCE BUDGET

Trigger boards have a hard ceiling; designs share the FPGA with the rest of the menu.



Graph Neural Networks for muon track-finding



NODES

Each detected particle hit becomes a node in the graph — preserving the detector's spatial structure.

EDGES

Potential connections between hits — based on proximity and expected trajectories — form the graph edges.

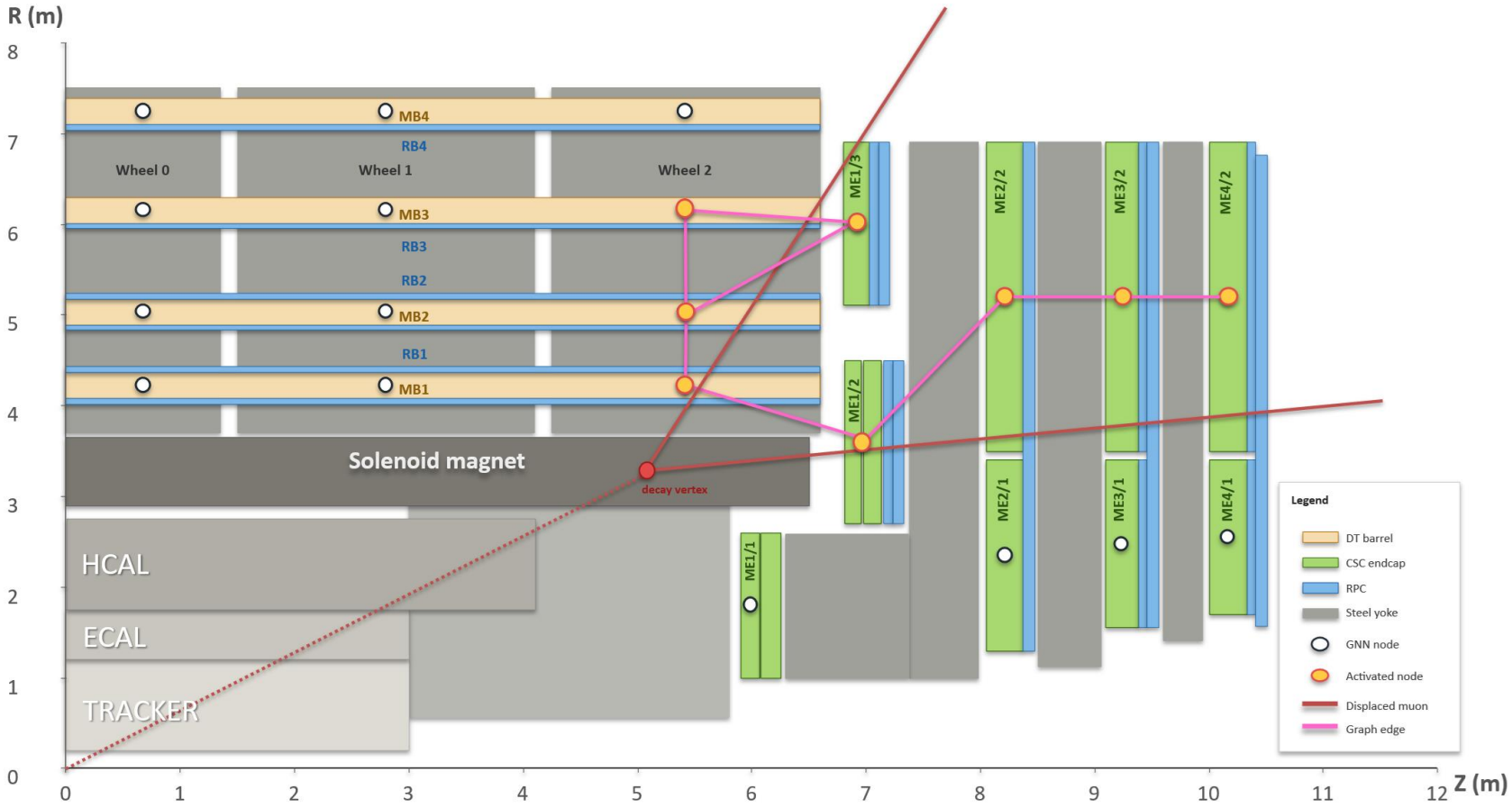
LEARNING

The GNN learns patterns of genuine muon tracks, filtering out background noise while leveraging detector geometry natively.

Trade-off: data-driven recognition outperforms fixed algorithms — but variable graph sizes and tight FPGA latency budgets are the engineering challenge.

Graph Neural Networks for muon track-finding

Detector stubs form a sparse relational pattern. A GNN represents stubs as nodes and learned compatibility as graph edges — a natural candidate for track-finding.



Detector stubs
sparse hits across layers

Graph
nodes + learned edges

Track decision
GNN classification

APPROACH

Two threads that must co-evolve

Model exploration

- **Quantisation —** float32 → INT8-PO2
- **Graph representation —** fixed-size, bounded nodes/edges
- **Structure —** latency-driven architectural choices

Interface verification

- **Contracts —** interfaces described as YAML
- **Generated DUTs —** top + testbenches, not hand-written
- **Reusable runtime —** one verification flow, every variant

GraphSAGE proxy

2-layer · mean aggregation · feature projection · HLS-friendly fixed loop bounds · simplified root-feature handling

Each node looks at its neighbours

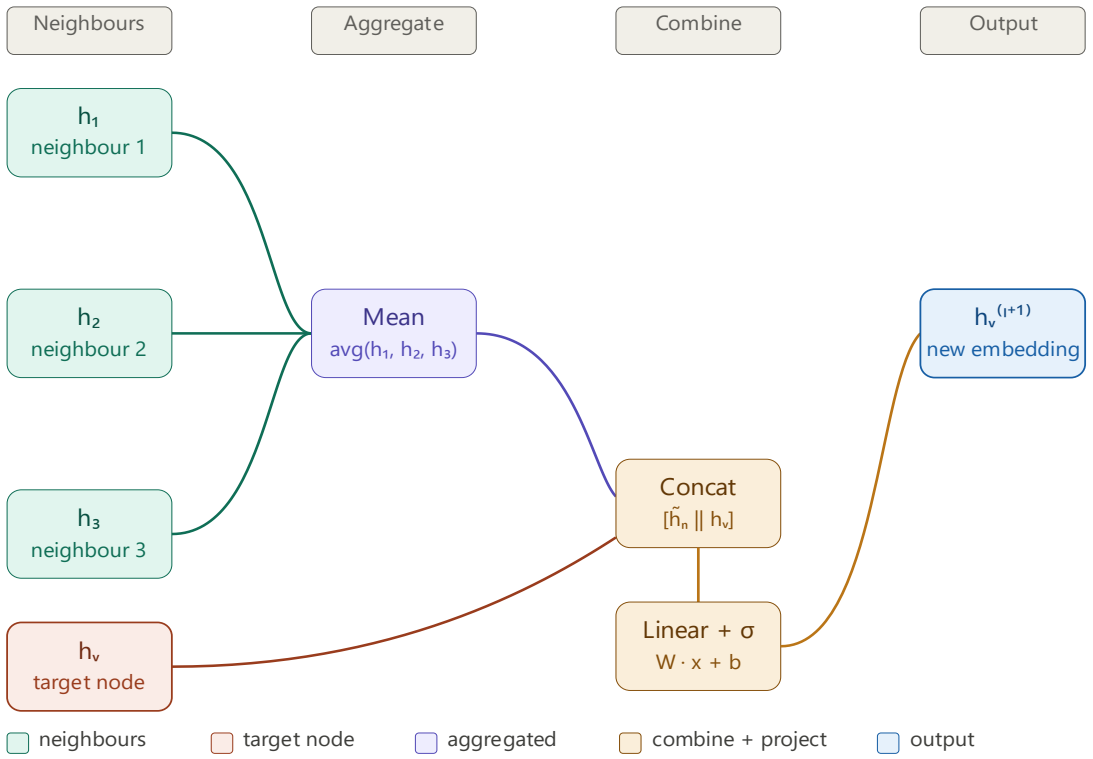
For every node, GraphSAGE collects information from the nodes connected to it.

Neighbour features are aggregated

It combines neighbour information using an operation like mean, sum, or max, producing one summary vector.

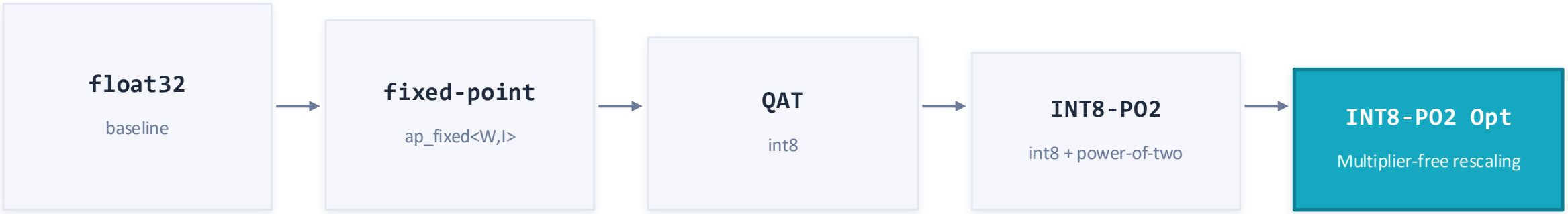
The node updates itself

The node mixes its own features with the aggregated neighbour features through learnable weights.



From float32 down to INT8-PO2

We sweep precision along a spectrum, trading numerical headroom for fewer DSPs and lower latency — and verify accuracy at each step.



more precision · more resources

fewer DSPs · lower latency

Why INT8-PO2 is the FPGA-friendly end

Power-of-two scales turn multiplications into bit-shifts — so DSP-hungry multipliers collapse into cheap shift-and-add logic, cutting resource use and latency while keeping accuracy within target.

Meeting L1 resource & latency budgets

Design	Acc.	Type	Latency (cyc)	Latency (ns)	DSP	FF	LUT	Freq.
Float32	78.0%	FP32	603	1,670	**34,880 (283%)**	**4.45M (128%)**	**2.14M (123%)**	—
Fixed-PTQ	75.7%	Fixed	77	213	6,976 (56%)	262k (7%)	110k (6%)	397 MHz
INT8-PO2	75.0%	INT8	28	77.6	5,320 (43%)	254k (7%)	188k (10%)	450 MHz
QAT-v2	76.8%	INT8	55	152.3	6,760 (55%)	561k (16%)	282k (16%)	493 MHz
★INT8-PO2 Opt. best	**75.0%**	**INT8**	**19**	**52.6**	**2,560 (20%)**	**232k (6%)**	**477k (27%)**	**502 MHz**

LATENCY

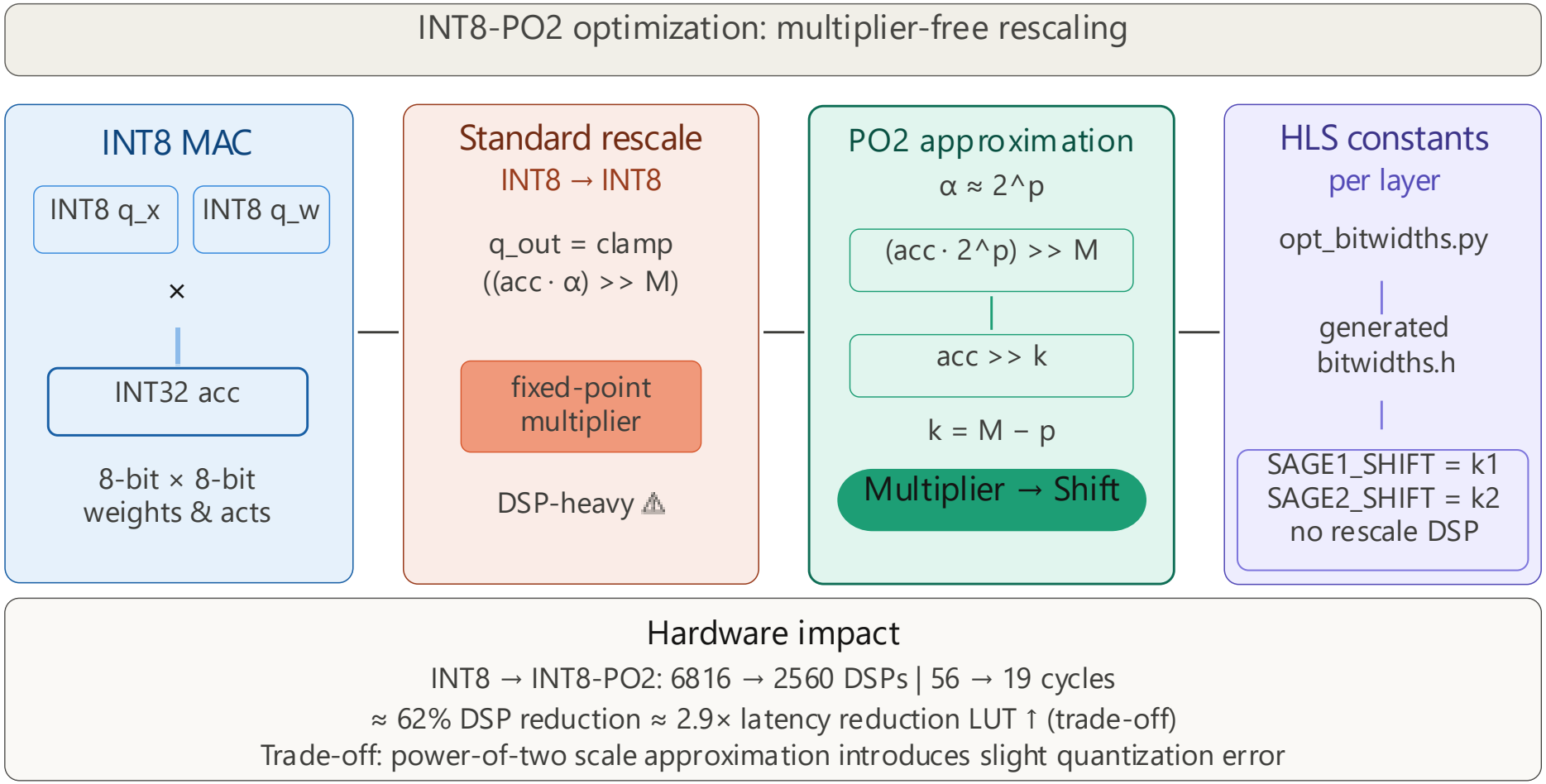
≈ 52 ns

Initiation interval = 1 · well inside the per-algorithm slice of the 12.5 µs window

L1-compatible

Resulting GNN configurations meet L1-trigger-compatible resource and latency budgets — INT8-PO2 collapses DSP use while holding accuracy.

Meeting L1 resource & latency budgets



INT8 quantisation

Post-training calibration. Each INT32 accumulation needs a fixed-point rescale.

PO2 trick

Each rescale factor ≈ 2^p — multiplications become arithmetic shifts.

HLS emission

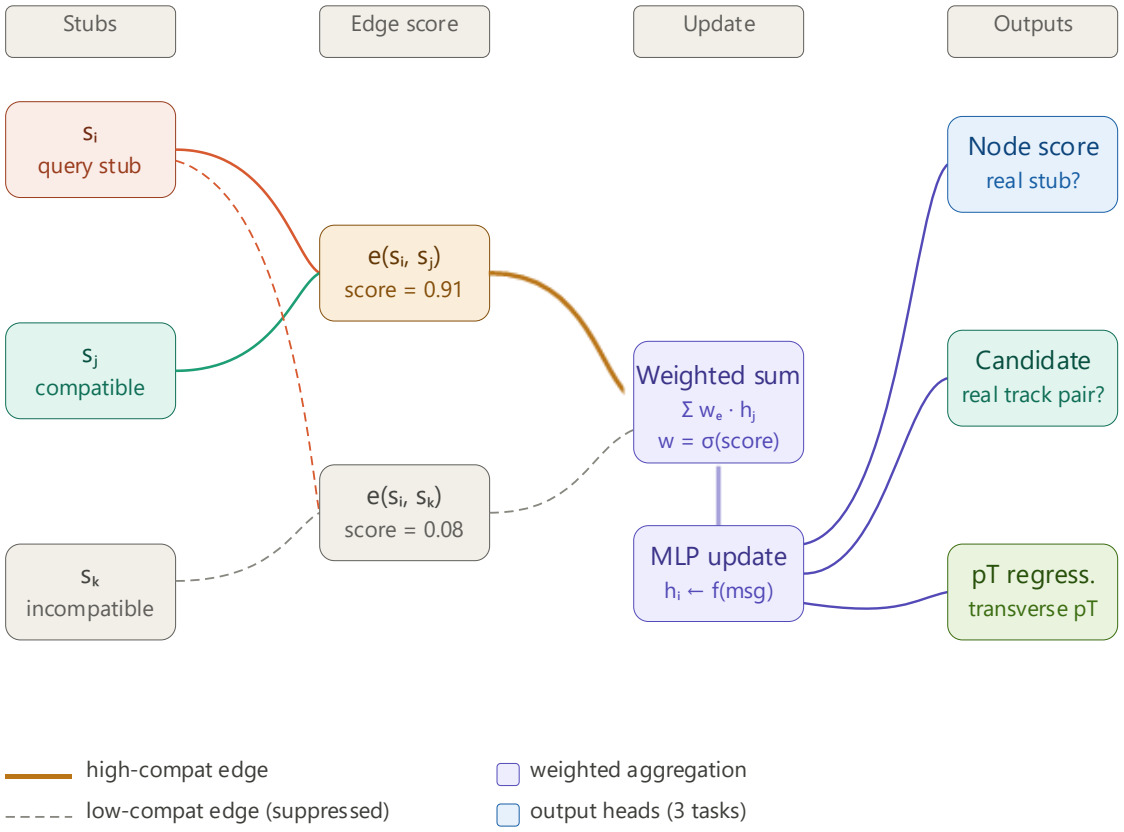
Shift constants written to the HLS header and consumed by the FPGA kernel.

Trade-off

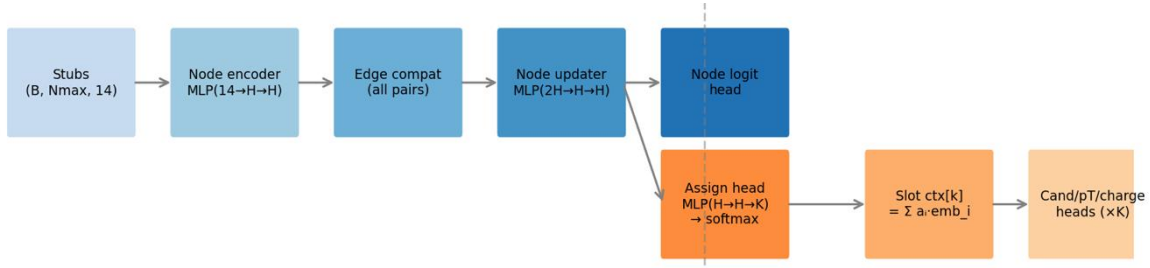
DSP ↓ & latency ↓ but LUT ↑ and small accuracy loss.

Improving performance: EdgeCompatNet Architecture

This is well suited for CMS track finding because the main challenge is not only detecting individual stubs but identifying which sparse detector hits form a coherent track under high pile-up and background.



ARCHITECTURE:



HIDDEN DIMENSION: H = 64

h = 64 is the sweet spot between accuracy and FPGA feasibility. Larger widths exceed DSP/LUT budgets; smaller ones lose expressivity.

EDGECOMPAT PIPELINE

Stubs treated as nodes; pairwise compatibility scores learned via edge MLP. Weighted aggregation + MLP update produce per-stub embeddings.

Three output heads:

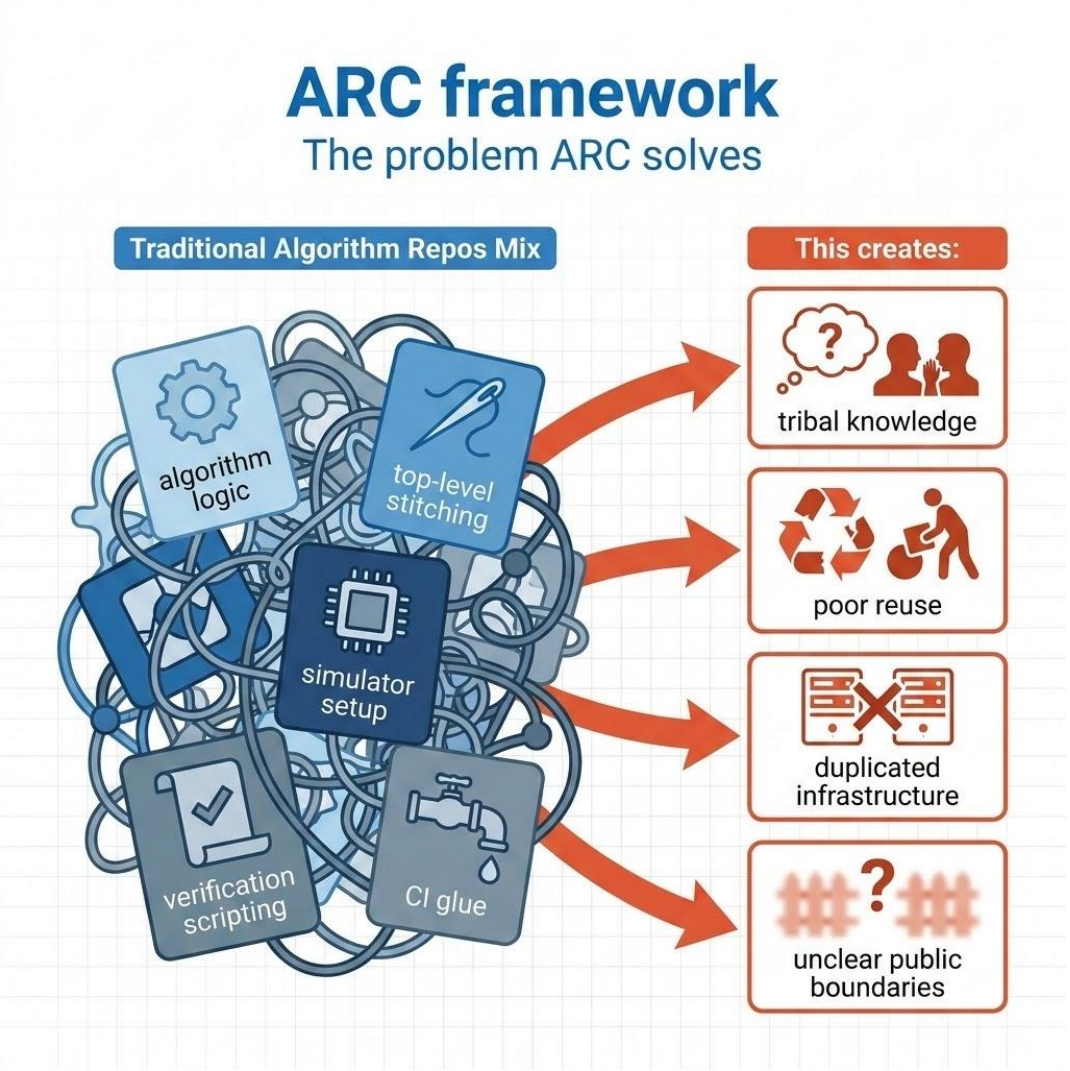
- node score (real stub?),
- candidate score (real track pair?),
- p_T regression.

VERIFICATION · MOTIVATION

Exploration multiplies the verification burden

Every precision and structural variant is a different DUT. The classic hardware repo entangles four things in one place — and you re-solve them for each variant:

- **Algorithm logic** — the actual HLS / RTL
- **Top-level stitching** — hand-wired connectivity
- **Simulator setup** — compile / elaborate glue
- **Testbench scripting** — stimulus + checking by hand



The fix: generate integration & verification from contracts instead of hand-crafting them. → ARC

ARC · CORE IDEA

Contracts separate meaning from mechanics

ARC is a plugin-agnostic, contract-driven framework: algorithm teams describe their modules, interfaces, topology and datasets as contracts — the framework turns those into generated DUTs and a shared verification runtime.

Plugin owns MEANING

- Algorithm & HLS / RTL source
- Semantic interface contracts
- Design topology & datasets
- Stimulus + optional semantic checks

Framework owns MECHANICS

- Generate the stitched DUT & metadata
- Orchestrate HLS synthesis flows
- Generate & run csim / xsim verification
- CI gating, independent of algorithm

“The plugin owns meaning; the framework owns mechanics.”

From interface YAMLs to verified DUTs

PLUGIN-AUTHORED CONTRACTS

- `modules.yml`
- `interfaces/*.interface.yaml`
- `designs/design.yml`
- `design.verification.yml`

consumes

ARC FRAMEWORK

topgen

load contracts · derive wiring · generate the stitched DUT

HLS helpers

generate TCL · run HLS stages · collect metrics

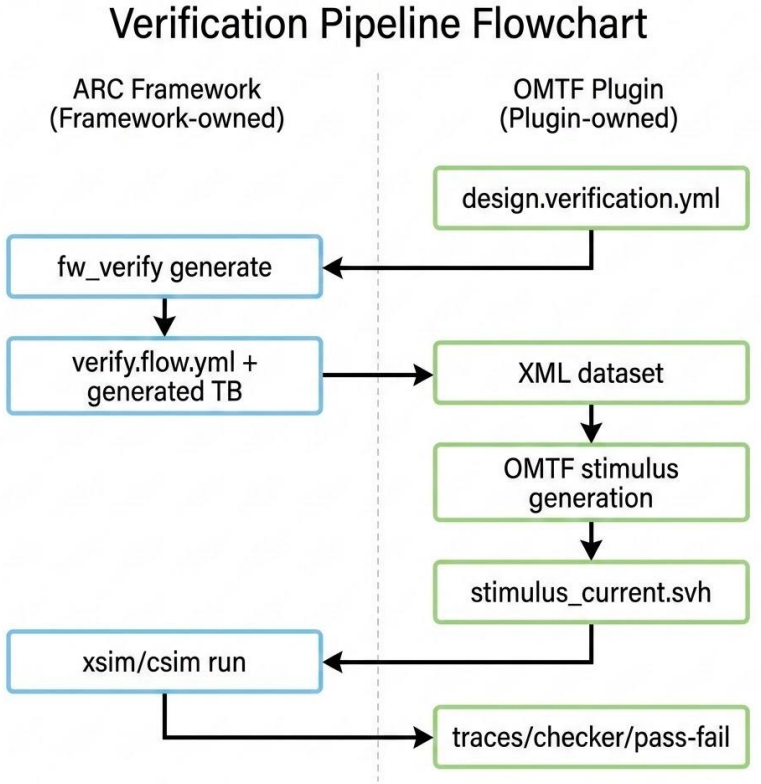
fw_verify

per-flow runtime · run csim / xsim · preflight

generates

GENERATED ARTEFACTS

- `algo_top.v`
- `build_manifest.json`
- `port_map.yaml`
- `tb_*.sv`
- `verify.flow.yml`
- `wave.tcl`



Every GNN variant becomes a reproducible artefact

GNN VARIANTS

- `float32 baseline`
- `INT8`
- `INT8-PO2`
- `structure A`
- `structure B`

each = one contract set

PER-VARIANT, GENERATED

Generated DUT + verification flow

stitched top, testbenches and runtime per variant

Regression-friendly CI

re-run the whole sweep; catch interface breakage early

Clear ownership

algorithm semantics vs integration mechanics stay separate

Contract-bound artefacts

replace fragile, hand-written testbenches

SUMMARY

A viable path to ML track-finding in Phase-2

FPGA-ready GNN

GraphSAGE proxy on Cora quantised from float32 to INT8-PO2, hitting 75% acc. at 20% DSP (h = 64) within L1 latency.

ARC framework

Contract-driven integration turns each GNN variant into a reproducible artefact with auto-generated testbenches.

Co-evolution

ML exploration and HW verification advance together — EdgeCompatNet is already being targeted for CMS deployment.

Next: Broaden quantisation sweep · port EdgeCompatNet to FPGA via HLS · integrate CMS OMTF geometry into ARC · validate on Phase-2 simulation.

Thank you