Hardware-accelerated GNN-based hit filtering for the Belle II Level-1 trigger

Published by IOP Publishing for Sissa Medialab

Received: *November 6, 2025*
Accepted: *January 9, 2026*
Published: *February 11, 2026*

Topical Workshop on Electronics for Particle Physics
Rethymno, Crete, Greece
6–10 October 2025

Hardware-accelerated GNN-based hit filtering for the Belle II Level-1 trigger

G. Heine[a,*] **F. Mayer**[b] **M. Neu**[b] **J. Becker**[b] **and T. Ferber**[a]

[a] *Institute of Experimental Particle Physics, Karlsruhe Institute of Technology, Wolfgang-Gaede-Straße 1, 76131 Karlsruhe, Germany*
[b] *Institute for Information Processing Technology, Karlsruhe Institute of Technology, Engesserstraße 5, 76131 Karlsruhe, Germany*

E-mail: greta.heine@kit.edu

Abstract: We present a hardware-accelerated hit filtering system employing Graph Neural Networks (GNNs) on Field-Programmable Gate Arrays (FPGAs) for the Belle II Level-1 Trigger. The existing track trigger exhibits reduced efficiency, in particular for short and endcap region tracks, due to its requirement of aligned hits in the majority of detector layers. An efficient hit filter enables the subsequent track trigger to operate with relaxed requirements, allowing higher track finding efficiency without increasing the Level-1 trigger rate. The proposed GNN system exploits spatial and temporal correlations among sense wire hits for detector-level background suppression. The network is optimized for high-throughput hardware operation via quantization, pruning, and static graph-building. Sector-wise spatial parallelization permits scaling to full-detector coverage while satisfying Belle II latency and throughput requirements. At 32 MHz throughput and sub-microsecond latency, the system achieves an offline-validated 83 % background hit rejection at 95 % signal hit efficiency. This work demonstrates scalable, low-latency GNN-based hit filtering on FPGAs for real-time data reduction in high-luminosity collider environments.

Keywords: Data reduction methods; Pattern recognition, cluster finding, calibration and fitting methods; Trigger concepts and systems (hardware and software); Particle tracking detectors

ArXiv ePrint: 2511.04731

*Corresponding author.

https://doi.org/10.1088/1748-0221/21/02/C02007

Contents

1 Introduction

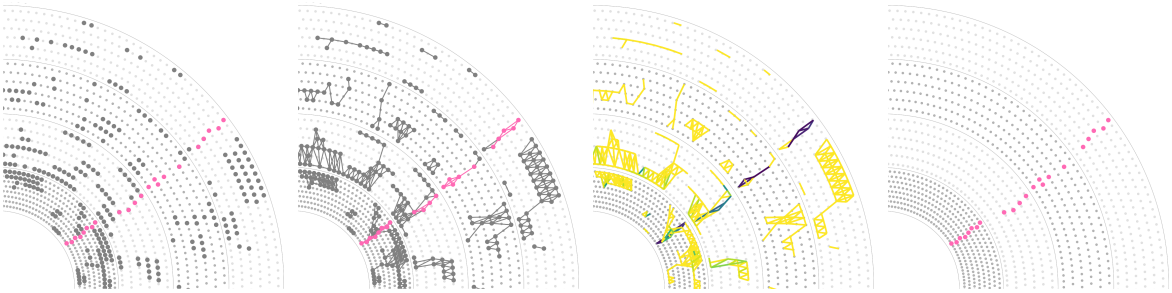
The Belle II experiment [1] at the SuperKEKB electron-positron collider (Japan) studies flavour and dark sector physics at the $\Upsilon(4S)$ resonance energy of 10.58 GeV. As SuperKEKB approaches an instantaneous luminosity of $6 \times 10^{35}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, the resulting rise in beam-induced backgrounds places increasing demands on the Belle II trigger system.

The Data Acquisition (DAQ) system manages a readout throughput up to $3\,\mathrm{GB\,s}^{-1}$ [2], employing a two-level trigger system to process the approximately 250 MHz bunch crossing rate. The Level-1 Trigger (TRG) system utilizes Field-Programmable Gate Arrays (FPGAs) for real-time event selection with a total latency budget below 4.4 µs while reducing event rates to a maximum of 30 kHz. The Central Drift Chamber (CDC) trigger [3] is one of four sub-trigger components that analyze data from dedicated sub-detector Front-End Electronics (FEE) boards and forward outputs to the global trigger system for the final trigger decision [4]. It reconstructs charged particle trajectories from 14 336 sense wires per clock cycle of the $f_{\mathrm{sys}} = 127.216\,\mathrm{MHz}$ system clock, with an internal data clock of $f_{\mathrm{CDC}} = 31.804\,\mathrm{MHz}$, to derive kinematic track properties and suppress background.

Increasing luminosity directly raises detector occupancies and background levels. The average number of unassigned CDC hits per event, n_{extraCDC}, has risen from about 200 in 2021 to about 1900 in 2024, with projections up to about 2800 for future high-luminosity running. The current Belle II CDC trigger algorithm mitigates these backgrounds by requiring aligned hits in a majority of the CDC layers and by down-scaling the trigger rate of specific track topologies. These measures effectively suppress background but also constrain the overall track finding efficiency. Internal studies suggest even stricter criteria may be necessary in the future with rising background levels to stay within the DAQ limits.

Graph Neural Networks (GNNs) offer a promising solution for background rejection by leveraging both individual hit characteristics and the spatial and temporal patterns in hit topologies. Recent work on deploying GNNs for track reconstruction on FPGAs [5, 6] and on ultra-fast architectures for jet tagging [7] has demonstrated the feasibility of real-time GNN inference in high-rate collider environments. While these works primarily focus on track-level reconstruction or jet tagging

2026 JINST 21 C02007



(a) before filtering (b) graph representation (c) GNN classification (d) after filtering

Figure 1. Overview of the GNN-based hit filtering process: (a) CDC hits before filtering with signal hits (pink) and background hits (grey), (b) hits are represented as graphs with edges connecting spatially compatible hits, (c) our GNN performs edge or node classification to identify signal patterns where a dark colour denotes signal-like and light colour background-like classification, and (d) classification outputs are mapped back to individual hits for filtering.

leveraging higher-level objects, our approach is distinguished by its focus on hit-level filtering directly on CDC sense wire data. This enables real-time detector-level background suppression, achieving sub-microsecond latencies prior to track reconstruction in the Belle II TRG.

This paper describes the GNN-based hit filtering algorithm (section 2) and its hardware implementation (section 3), and evaluates performance in terms of hit classification efficiency, resource utilization, and latency (section 4), concluding in section 5.

2 GNN-based hit filtering

We developed a GNN-based hit filtering system developed for deployment in the Belle II TRG system. The algorithm is designed with hardware constraints in mind but is first developed, trained, and validated in software using offline data. It involves a four-stage process shown in figure 1 consisting of (1) reading and preprocessing of CDC hits, (2) generating a graph representation of the hits, (3) applying GNN classification on these graphs, and (4) a final hit filtering based on the GNN classification scores.

2.1 Graph representation, network architecture and compression

The CDC hit data (figure 1(a)) is transformed into a graph representation for GNN processing (figure 1(b)). In each event, hits are represented as graph nodes characterized by their x and y coordinates as well as Analog-to-Digital Converter (ADC) values, corresponding to the accumulated charge in the drift cell. Edges are formed using geometric rules to control connectivity and reduce computation [8]: (1) For same layer connections ($\Delta l = 0$), wires with distance $\Delta w = \pm 1$ connect, (2) adjacent outgoing layers ($\Delta l = +1$) connect direct neighbours ($\Delta w = 0$), and (3) next-to-next layers ($\Delta l = +2$) connect wires offset by $\Delta w = -1, 0$, or $+1$. This pattern nearest-neighbour graph building mirrors the expected topology of particle trajectories. Edge features Δr, $\Delta\phi$, and ΔTDC capture hit relationships, indicating if hits are originating from the same trajectory. Static graph features (x, y, Δr, $\Delta\phi$) are pre-computed in FPGA firmware to reduce online inference effort.

The GNN utilized in this work is a compressed variant of the Interaction Network [9]. It comprises three Multi-Layer Perceptron (MLP) blocks: two edge blocks $R1$ and $R2$, and a node block O, in addition to edge-to-node aggregation mechanisms, which form the basis of the hardware-accelerated

2026 JINST 21 C02007

architecture illustrated in figure 2. This architecture employs two message-passing steps for local pattern recognition within a two-node vicinity and is designed for low computational cost. Our implementation can operate in both edge and node classification modes (figure 1(c)), with node classification providing the best performance for the final hit filtering task (figure 1(d)).

The following compression methodologies optimize the model for hardware efficiency. Reducing the hidden layers' depth and width cuts the number of trainable parameters from 610 to 211. We apply quantization-aware training with Brevitas [10], using 4-bit inputs, weights, and activations, 16-bit biases, and 8-bit outputs. Pruning to 50 % sparsity further reduces the network size and unexpectedly improves performance. For FPGA deployment, the aggregation between $R1$ and O is changed from sum to max to prevent overflow, and in inference, the sigmoid activation after $R2$ is replaced by a linear mapping to reduce computation, without degrading performance. Overall, the compression reduces the bit representation from 19 520 bits to 797 bits per instance. The impact on hit-classification performance, relative to the original network, is discussed in section 4.1.

2.2 Training strategy and dataset

For training, we constructed a dataset inspired by [11] to cover the full range of physics signatures and background conditions relevant to Belle II operation. It consists of simulated Monte Carlo (MC) events with varied track topologies, overlaid with real beam-background data. Additionally, real data samples derived from low-bias single-track triggered events [12] and High-Level Trigger (HLT)-selected $\mu\mu(\gamma)$ events[1] are included. The ground truth labels for MC samples utilize MC truth information, whereas real data samples depend on offline track reconstruction to distinguish between signal and background classifications. We apply an offline pre-filtering to the CDC sense wire hits using their accumulated charge and timing digitized by the ADC and Time-to-Digital Converter (TDC) values, respectively (ADC$\geq$ 10, TDC within 500 ns trigger time window). This procedure preserves consistency with the established trigger pipeline, while simultaneously improving the signal-to-background ratio for more robust network training and inference. Node and edge features are normalized to the range from -1 to +1 and undergo 4-bit quantization, matching trigger-level conditions.

The GNN is implemented using PyTorch Geometric [15]. To mitigate class imbalance, a binary cross-entropy loss function with an emphasis on positive sample weighting (weight=10) is employed. The training runs for 50 epochs with a batch size of 1 and optimization facilitated by the AdamW [16] optimizer. The utilized learning rate scheduler starts at 0.01 and decays by a factor of 0.7 every 4 epochs. A dropout rate of 10 % is applied to reduce overfitting, and early stopping with a patience of 10 epochs terminates training once the validation performance plateaus.

3 Hardware implementation

As described in section 1, the TRG imposes strict latency and throughput requirements on the hit filtering algorithm. To fulfill these requirements, we deploy the Interaction Network discussed in section 2 as a dataflow accelerator on an FPGA. In order to facilitate the semi-automated implementation of the network on FPGA, we develop a deployment approach which transforms the compressed network

[1] HLT $\mu\mu(\gamma)$ events are selected by requiring two oppositely charged, back-to-back tracks, each carrying momentum greater than 0.5 GeV/c in the center-of-mass reference frame and matched to an Electromagnetic Calorimeter (ECL) cluster with energy below 0.5 GeV. The total energy of clusters, including possible photons, must be below 2 GeV [13, 14].

2026 JINST 21 C02007

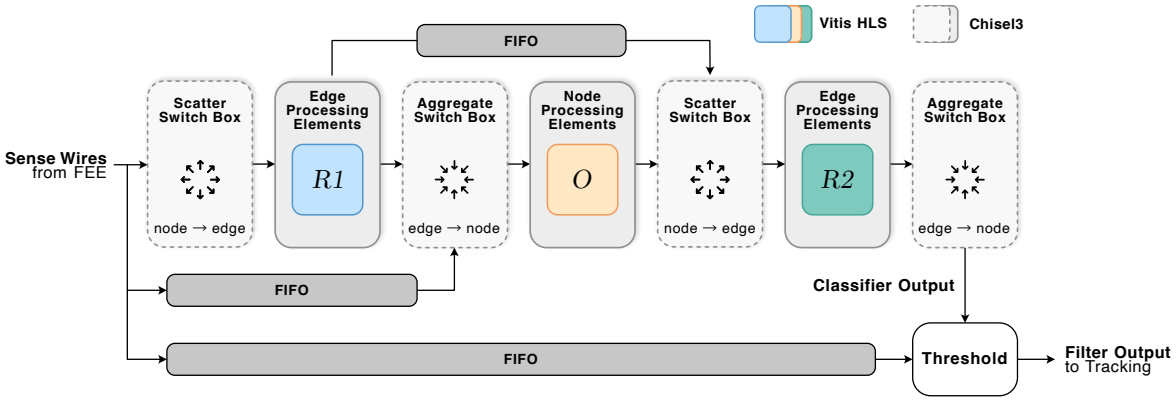


Figure 2. Block diagram of the hardware-accelerated Interaction Network architecture, where Vitis HLS synthesized network blocks are mapped to dedicated PEs. Static graphs generated from FEE-supplied sense wire data are propagated and updated via a series of scatter and aggregate Switch Boxes in-between PEs, realized in Chisel into a register-transfer level design. The final classifier outputs, after threshold application, are sent to downstream tracking modules.

into a register-transfer level design. Our approach maps each layer of the neural network onto a dedicated Processing Element (PE). In total, we design three types of PEs: (1) Scatter Switch Boxes, (2) Aggregate Switch Boxes, and (3) Network PEs. Switch Boxes are realized as hardware generators, implemented in the Chisel design language [17]. They embed the graph data structure into the dataflow accelerator, based on the approach described in [8]. Network PEs contain the trainable weights of the network, implemented using AMD Vitis HLS [18]. To implement the MLP blocks $R1$, $R2$ and O, we leverage architecture templates from the low latency HLS library [19]. Similar to hls4ml [20], we define a reuse factor $R \in \{2^i : i \in \mathbb{N}^+\}$ which defines the spatial parallelism of the dataflow accelerator.

For our network described in section 2, the resulting hardware architecture of our hit filtering algorithm is shown in figure 2. PEs are shown in grey and are connected by either First-In-First-Out buffers (FIFOs) or simple shift registers. All data interfaces are AXI4-Stream [21] compliant and decoupled via ready-valid handshake.

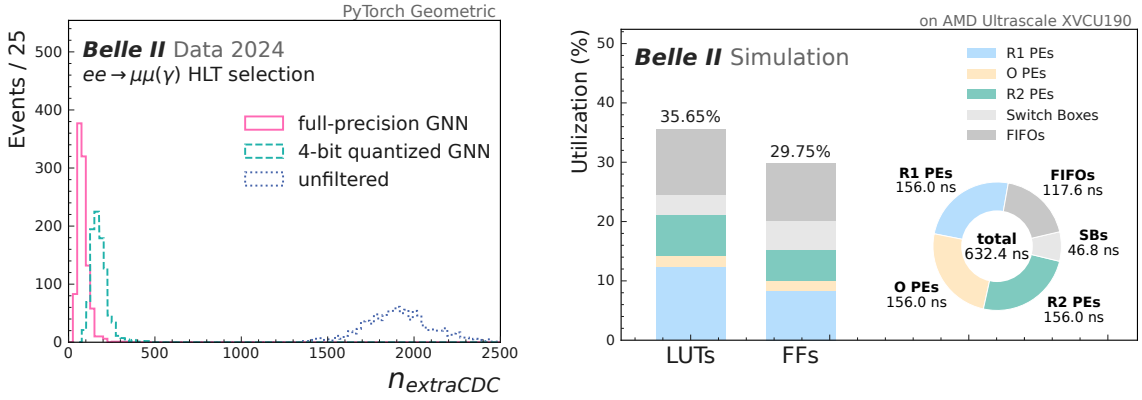
4 Performance evaluation

4.1 Hit filtering performance

We demonstrate the efficacy of the proposed GNN-based hit clean-up through offline re-analysis of Belle II data, focusing on HLT-selected $\mu\mu(\gamma)$ events recorded at the end of 2024. As illustrated in figure 3(a), we compare three configurations of the hit processing pipeline: the uncompressed full-precision GNN model, its quantized 4-bit counterpart, and the unfiltered case. We run this simulation using sense wires of the full CDC detector. For performance evaluation, hits are classified based on their association with offline reconstructed tracks. Signal hits are matched to tracks satisfying quality criteria,[2] while background hits n_{extraCDC} comprise all unmatched hits, with no quality selection applied. Hits matched to tracks failing the quality criteria ($< 5\,\%$ of signal, $< 0.1\,\%$ of all hits) are excluded from performance metrics. Background rejection and signal efficiency are computed as the

[2]Track selection requires transverse momentum $p_T > 0.2\,\text{GeV/c}$, total momentum $p > 0.7\,\text{GeV/c}$, longitudinal $|z_0| < 15\,\text{cm}$ and radial $|d_0| < 15\,\text{cm}$ distance from the interaction point, and at least 7 CDC hits [13, 14].

2026 JINST 21 C02007



(a) Hit filtering performance

(b) Resource utilization and latency

Figure 3. (a) Background-like hit distributions n_{extraCDC} processing Belle II data (HLT-selected $\mu\mu(\gamma)$ events [13, 14] from late 2024) with offline simulation on the full CDC including and excluding the GNN hit filtering. Both the full-precision and 4-bit quantised GNN models achieve a background hit rejection of > 80 % at 95 % signal hit efficiency. (b) FPGA resource utilization and latency per GNN logic block for 495 sense wires and 2163 edges, showing modest LUT/FF use and zero DSPs. The total pipeline latency amounts to 632.4 ns; results are reported from Vivado 2024.2 after routing in out-of-context mode.

fraction of background and signal hits removed and retained, respectively. Both the full-precision and 4-bit quantized models are applied at a working point corresponding to a 95 % signal hit efficiency. The resulting distributions show an 87 % background hit rejection for the full-precision model and 83 % for the 4-bit version, demonstrating small degradation from model compression.

4.2 Experimental setup, resource utilization and timing

As a demonstrator for FPGA implementation, we use one of the 20 CDC sectors described in [8], comprising 495 sense wires and 2163 edges. Defining the system frequency at $f_{\text{GNN}} = 127.216\,\text{MHz}$, in alignment with the TRG system clock, necessitates the processing of at least 124 nodes and 541 edges per clock cycle. With a reuse factor $R = 4$, we achieve the required throughput of 31.804 MHz.

We synthesize, place and route the register-transfer level design depicted in figure 2 using AMD Vivado 2024.2 [22] in out-of-context mode on an AMD Ultrascale XVCU190. For validation, we perform functional verification in a cycle-accurate simulation using CoCoTb 1.9.2 [23] and ModelSim 2023.4 [24]. The design meets all timing constraints with the processing latency and resource usage after routing detailed in figure 3(b). The core accelerator components, including the edge block ($R1$, $R2$) and node block (O) PEs, the Switch Boxes as well as FIFOs including buffers and queues show a total resource utilization of 35.65 % for Look-Up Tables (LUTs) and 29.75 % for Flip-Flops (FFs), without using any Digital Signal Processing (DSP) blocks. The pipeline latency is 163.8 ns per major logic block, totalling 632.4 ns end-to-end.

5 Conclusion

We have developed a GNN-based hit filtering system for the Belle II Level-1 trigger. Reprocessing 2024 Belle II data with a 4-bit quantized GNN yields 83 % background hit rejection at 95 % signal hit efficiency. We demonstrate the technical feasibility by deploying the network as a dataflow accelerator

2026 JINST 21 C02007

on an FPGA. Cycle-accurate register-transfer level simulation verifies functional correctness, and we implement the design out-of-context on an AMD Ultrascale XCVU190. For a demonstrator sector, resource usage is measured at 35.65 % LUTs, 29.75 % FFs, and zero DSPs, with a total pipeline latency of 632.4 ns, satisfying Belle II trigger constraints. This implementation provides a viable basis for future integration into the Belle II trigger. A detailed study of track-level and physics improvement is in progress.

Acknowledgments

We thank the Belle II trigger team for their support and feedback. We also thank T. Koga for comments on the manuscript, and G. D. Pietro for help with the Belle II software framework. This work is funded by BMFTR ErUM-Pro 05H24VK1.

References

[1] Belle-II collaboration, *Belle II Technical Design Report*, arXiv:1011.0352.

[2] S. Yamada et al., *Data Acquisition System for the Belle II Experiment*, *IEEE Trans. Nucl. Sci.* **62** (2015) 1175.

[3] Belle-II collaboration, *Central Drift Chamber for Belle-II*, 2017 *JINST* **12** C06014.

[4] Y.-T. Lai et al., *Design of the Global Reconstruction Logic in the Belle II Level-1 Trigger system*, *Nucl. Instrum. Meth. A* **1078** (2025) 170577 [arXiv:2503.02192].

[5] ATLAS collaboration, *Online track reconstruction with graph neural networks on FPGAs for the ATLAS experiment*, *EPJ Web Conf.* **337** (2025) 01042.

[6] J. Kvapil et al., *Intelligent experiments through real-time AI: Fast Data Processing and Autonomous Detector Control for sPHENIX and future EIC detectors*, *PoS* **ICHEP2024** (2025) 1033 [arXiv:2501.04845].

[7] Z. Que et al., *JEDI-linear: Fast and Efficient Graph Neural Networks for Jet Tagging on FPGAs*, arXiv:2508.15468.

[8] M. Neu et al., *Real-Time Graph Building on FPGAs for Machine Learning Trigger Applications in Particle Physics*, *Comput. Softw. Big Sci.* **8** (2024) 8 [arXiv:2307.07289].

[9] P.W. Battaglia et al., *Interaction Networks for Learning about Objects, Relations and Physics*, arXiv:1612.00222.

[10] G. Franco, A. Pappalardo and N.J. Fraser, *Xilinx/brevitas*, DOI:10.5281/zenodo.3333552 (2025).

[11] L. Reuter et al., *End-to-End Multi-track Reconstruction Using Graph Neural Networks at Belle II*, *Comput. Softw. Big Sci.* **9** (2025) 6 [arXiv:2411.13596].

[12] S. Bähr et al., *The neural network first-level hardware track trigger of the Belle II experiment*, *Nucl. Instrum. Meth. A* **1073** (2025) 170279 [arXiv:2402.14962].

[13] Belle II collaboration, *Belle II analysis software framework (basf2)*, DOI:10.5281/zenodo.16268234 (2025).

[14] Belle-II Framework Software Group collaboration, *The Belle II Core Software*, *Comput. Softw. Big Sci.* **3** (2019) 1 [arXiv:1809.04299].

[15] M. Fey and J.E. Lenssen, *Fast Graph Representation Learning with PyTorch Geometric*, arXiv:1903.02428.

2026 JINST 21 C02007

[16] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*, arXiv:1711.05101.

[17] J. Bachrach et al., *Chisel: constructing hardware in a Scala embedded language*, in the proceedings of the *49th Annual Design Automation Conference*, San Francisco, California, U.S.A., 3–7 June 2012, pp. 1216–1225, ACM [DOI:10.1145/2228360.2228584].

[18] AMD, *Vitis unified software platform*, https://www.amd.com/de/products/software/adaptive-socs-and-fpgas/vitis.html (2025).

[19] M. Neu et al., *Real-Time Graph-based Point Cloud Networks on FPGAs via Stall-Free Deep Pipelining*, arXiv:2507.05099.

[20] FastML Team, *fastmachinelearning/hls4ml*, DOI:10.5281/zenodo.1201549 (2023).

[21] Arm Limited, *AMBA AXI-Stream Protocol Specification*, https://developer.arm.com/documentation/ihi0051/latest/ (2021).

[22] AMD, *Vitis unified software platform*, https://www.amd.com/de/products/software/adaptive-socs-and-fpgas/vivado.html (2025).

[23] S. Hodgson et al., *cocotb: Python-based chip (rtl) verification*, https://github.com/cocotb/cocotb (2025).

[24] AMD, *Modelsim hdl simulator*, https://eda.sw.siemens.com/en-US/ic/modelsim/ (2025).

2026 JINST 21 C02007