TrackGNN: A Highly Parallelized and Self-Adaptive GNN Accelerator for Track Reconstruction on FPGAs

Shuyang Li[1*], Hanqing Zhang[2*], Ruiqi Chen[3], Bruno da Silva[3], Giorgian Borca-Tasciuc[4], Dantong Yu[5], Cong (Callie) Hao[6]
[1]Fudan University, [2]Zhejiang University, [3]Vrije Universiteit Brussel, [4]Rensselaer Polytechnic Institute,
[5]New Jersey Institute of Technology, [6]Georgia Institute of Technology

***Abstract*—Real-time track reconstruction in high energy physics imposes stringent latency constraints, hindering the deployment of graph neural networks (GNNs) on general-purpose platforms. We present TrackGNN[1], an open-sourced GNN accelerator for track reconstruction. Using a dataflow architecture with multiple parallelism and a self-adaptive renaming mechanism, TrackGNN shows 27.6× speedup over CPUs, up to 101.1× over GPUs, and 5.7× over an FPGA overlay. Compared with FlowGNN, the renaming mechanism also reduces end-to-end latency by 1.12-1.16× with negligible resource overhead.**

I. Introduction

GNNs have emerged as a promising approach for track segment classification during track reconstruction, a process to determine track segments produced by the same particle. However, running real-time GNN inference on CPUs and GPUs is challenging due to strict latency constraints. Our work presents an FPGA-based dataflow accelerator for track reconstruction with microsecond-scale latency.

II. Architecture and Experiments

The GNN for track reconstruction accepts clustered hits as an over-connected graph, filtering the outliers and keeping the true hits produced by the same particle through message passing. TrackGNN inherits its highly parallelized dataflow architecture from FlowGNN [1]. As shown in Fig. 1, during the graph loading phase, graph data are streamed through the **Node Network** and **Edge Network** in parallel. The dataflow is managed by an **Adapter Queue** that multicasts the node embeddings to edge processing units.

However, irregular node ordering frequently misaligns adjacency accesses. The key improvement of TrackGNN over the original FlowGNN is a novel **renaming mechanism** illustrated in Fig. 1. The renaming and reindexing process is dynamic and scalable to different graph sizes. Consider the first 4 edges to be processed in parallel are $\{(n_0, n_{22}), (n_0, n_{23}), (n_0, n_{24}), (n_1, n_{22})\}$. We rename the node indices according to their processing orders in the Edge Network. The updated adjacency list becomes $\{(n_0, n_1), (n_0, n_2), (n_0, n_3), (n_4, n_1)\}$. The renamed adjacency list and the updated embedding arrays allow data to flow efficiently through the FIFOs, thus reducing initiation delays and pipeline stalls.

We deploy TrackGNN on the AMD Alveo U50 FPGA, and compare it against an AMD Ryzen 9 9950x CPU, a NVIDIA GeForce RTX 4090 GPU, and a state-of-the-art FPGA-based GNN Overlay [2]. Results are shown in Table I. The average latency of TrackGNN is 7.22μs. TrackGNN achieves a 27.6× speedup over the CPU, up to 101.1× over the GPU, and 5.7× over the overlay, while consistently showing better energy efficiency.

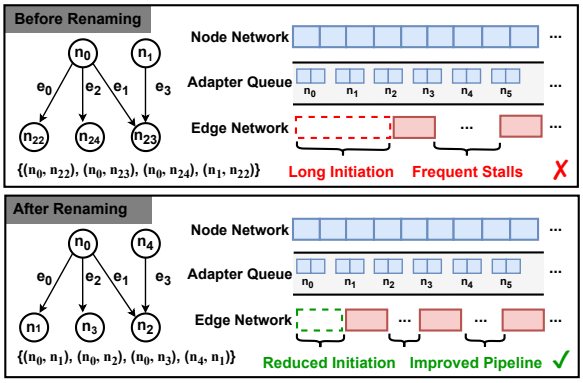

Fig. 1: Renaming reduces initiation time and pipeline stalls.

TABLE I: Performance Comparison of TrackGNN

Metric	CPU	GPU	Graph-OPU [2]	TrackGNN
Latency (μs)	199	730	41.15	7.22
Power (W)	26.92	68.5	15.5	13.37
Energy (mJ)	5.37	50	0.64	0.097

TABLE II: Latency Comparison with FlowGNN [1].

Architecture	Average-sized Graph	Large-Graph
FlowGNN	8.1μs	14.16μs
TrackGNN	7.22μs (1.12×)	12.17μs (1.16×)

To study the performance boost from the self-adaptive renaming, we further conduct an ablation study. From Table II, for average-sized graphs (∼20-70 nodes), TrackGNN achieves a speedup of 1.12× over FlowGNN, and increases to 1.16× for larger graphs(∼70-300 nodes) because large graphs introduce more data irregularities. The resource usage of DSPs and BRAMs remains largely unaffected. The LUT and FF consumption increased by only 3% and 6%, respectively. However, the slight increase of less than 6% resource usage is validated by up to 16.67% reduction in end-to-end latency.

References

[1] Sarkar, R. *et al.*, "FlowGNN: A Dataflow Architecture for Real-Time Workload-Agnostic Graph Neural Network Inference," in *HPCA*, 2023.
[2] Chen, R. *et al.*, "Graph-OPU: A Highly Integrated FPGA-Based Overlay Processor for Graph Neural Networks," in *FPL*, 2023.

*These authors contributed equally to this work.
[1]https://github.com/silvenachen/TrackGNN

2576-2621/25/$31.00 ©2025 IEEE
DOI 10.1109/FCCM62733.2025.00029