EVA: Energy-Efficient Virtual Node Accelerator for Graph Neural Networks

Dhruva Digesh Barfiwala

Parima Devanshu Mehta

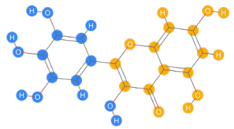
Georgia Tech

ECE 8893 – Parallel Programming for FPGAs

Motivation

Graph Neural Networks

- Handle non-Euclidean data
- Existing Models: GIN, GCN, GAT, GraphSAGE, etc.



Molecules

Social Networks

Existing GNNs fail to distinguish simple graph structures

Existing GNN Inference Accelerators (e.g., GRIP, BoostGCN) are not suitable for real-time inference

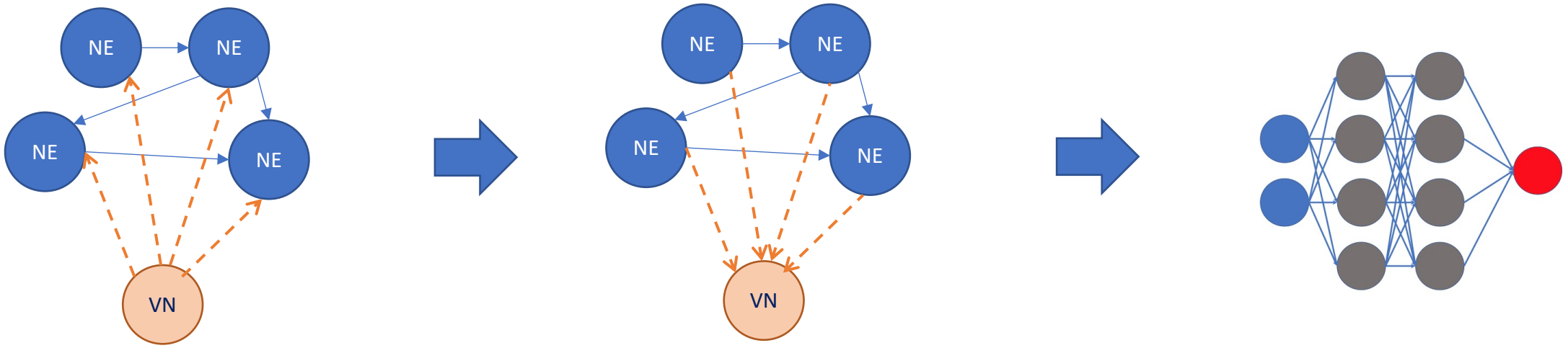
- Critical tasks in chemistry, medicine, etc. demand high accuracy and low inference latency
- Real-time inference of GNNs augmented with Virtual Nodes remains relatively unexplored

Virtual Node Augmentation

- Virtual node is a dummy node connected to all graph nodes
- Enhances the representation power of GNNs by providing a shortcut during message passing

Table 15: **Results for `ogbg-molhiv`.**

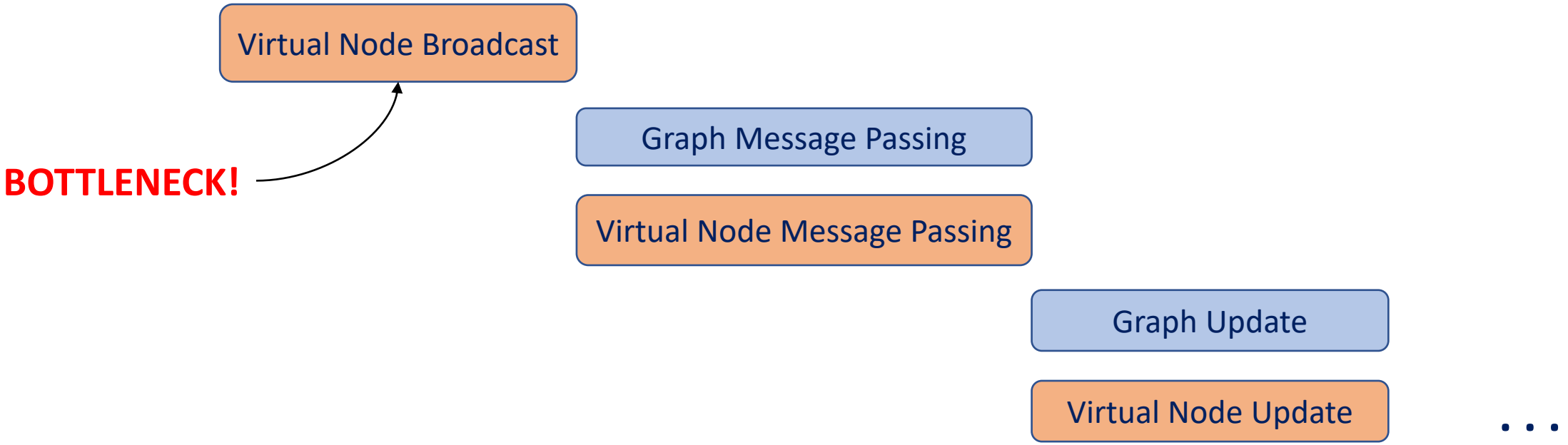
Method	Additional Features	Virtual Node	ROC-AUC (%) Training	Validation	Test
GCN	✗	✓	88.65±1.01	83.73±0.78	74.18±1.22
	✓	✗	88.65±2.19	82.04±1.41	76.06±0.97
	✓	✓	90.07±4.69	83.84±0.91	75.99±1.19
GIN	✗	✓	93.89±2.96	84.1±1.05	75.2±1.30
	✓	✗	88.64±2.54	82.32±0.90	75.58±1.40
	✓	✓	92.73±3.80	**84.79**±0.68	**77.07**±1.49



Virtual Node Broadcast **Virtual Node Message Passing** **Virtual Node Update**

Challenges with Virtual Node

- Virtual node broadcast falls on the critical path for graph message passing
- Latency and resource intensive - involves all nodes of the graph
- One Layer of GNN augmented with Virtual Node:



GOAL

Energy efficient virtual node augmentation to Graph Isomorphism Network (GIN) with minimal latency and area overhead

Setup

Model Training

- Trained GIN-Virtual Model implemented in PyTorch
- Model specifications:
  - # Layers: 5
  - Emb Dim: 100
- Dataset: ogbg-molhiv
- Saved pretrained weights

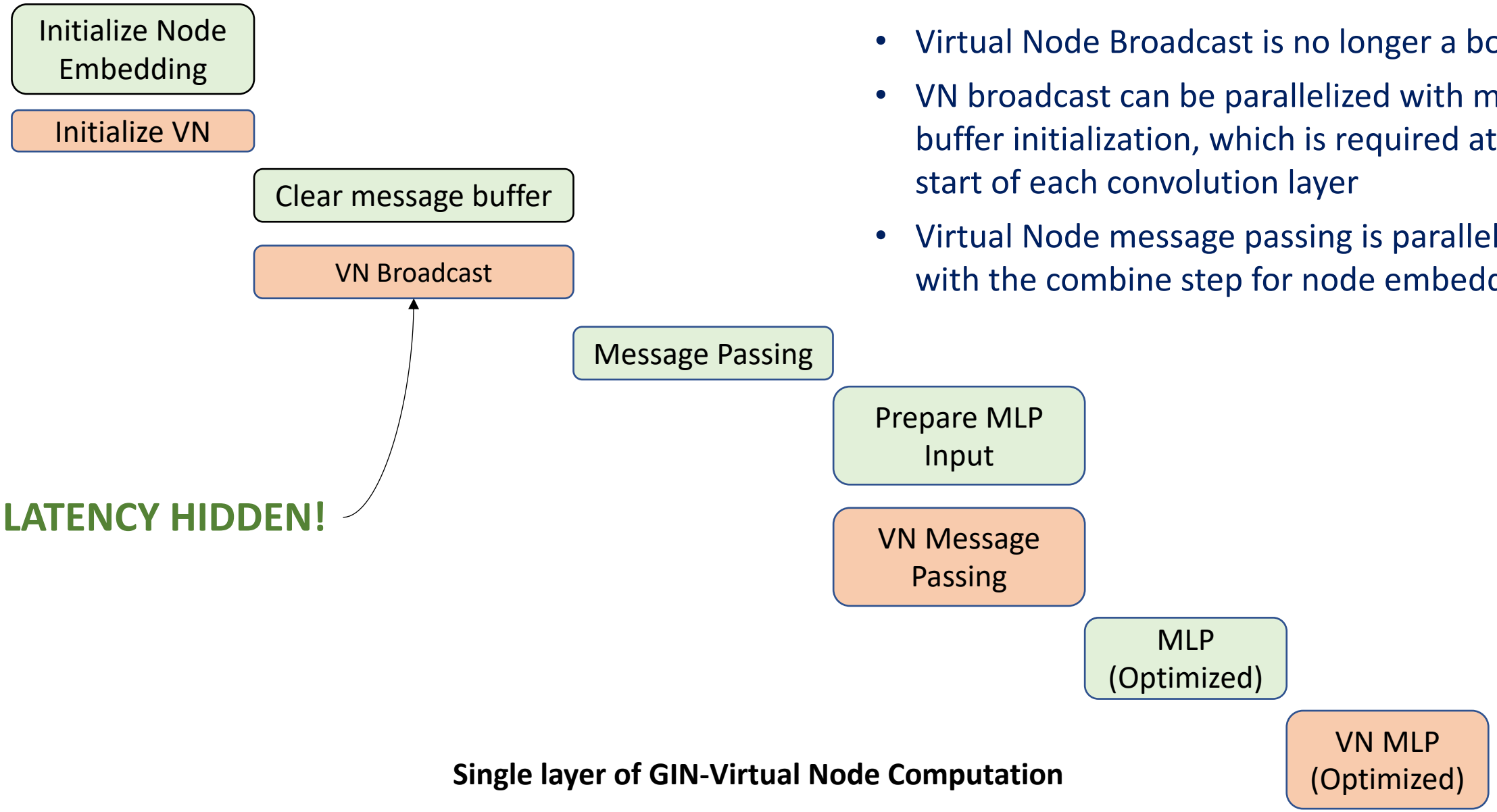
Collect Model Weights and Golden Output

- Employed *weight fusion* to reduce HW complexity
- Organized the fused weights into binaries
- Collected output from PyTorch implementation for Golden C verification

Golden C Implementation

- Translated PyTorch GIN-Virtual model to Golden C
- Automated C header generation
- Verified functional correctness against PyTorch implementation
- MSE of $\mathbf{10^{-4}}$ was observed

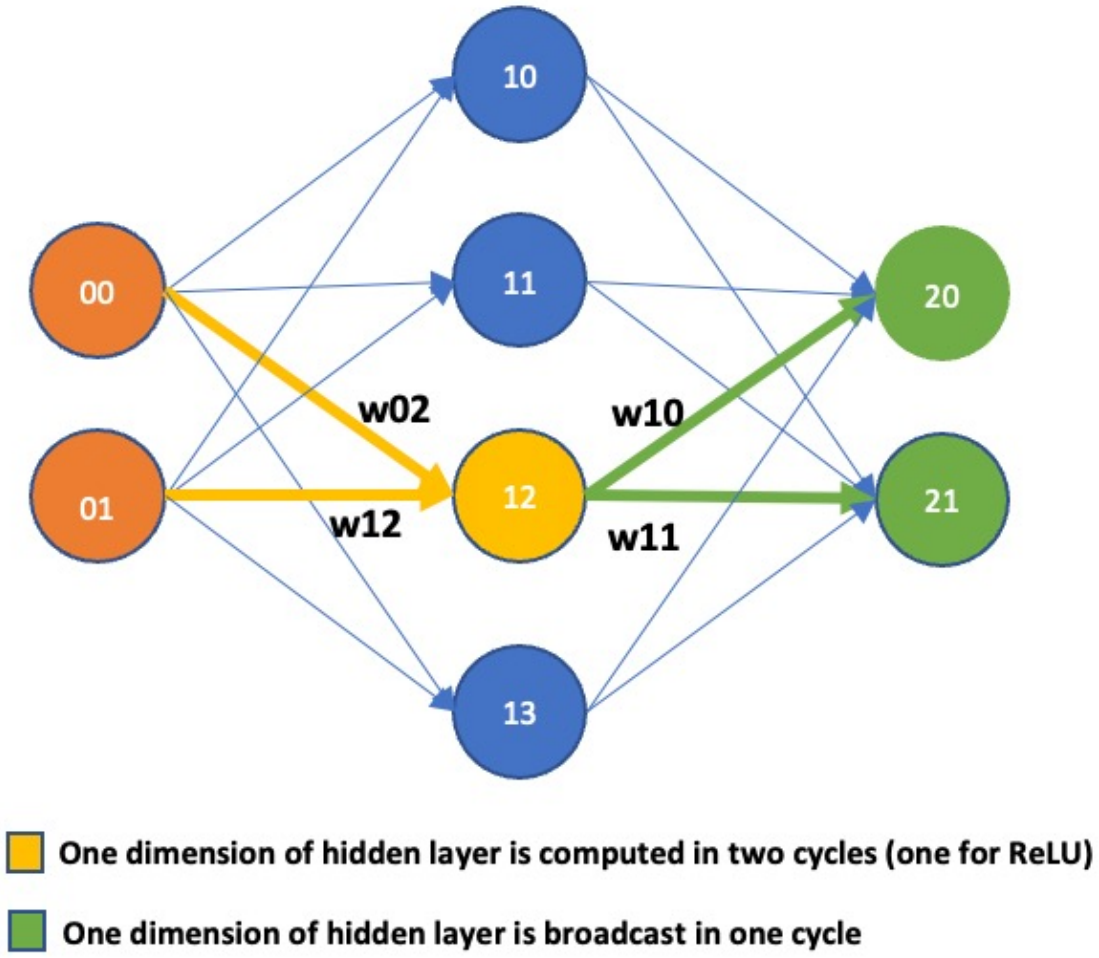
Base HLS with Virtual Node



- Virtual Node Broadcast is no longer a bottleneck
- VN broadcast can be parallelized with message buffer initialization, which is required at the start of each convolution layer
- Virtual Node message passing is parallelized with the combine step for node embeddings

Single layer of GIN-Virtual Node Computation

Base HLS: VN MLP Optimization



- MLP is **pipelined** across the hidden MLP layer dimensions
- MLP input is **partitioned** to ensure each hidden layer dimension can be computed in a single cycle
- MLP output is **partitioned** to allow broadcast of hidden layer in one cycle

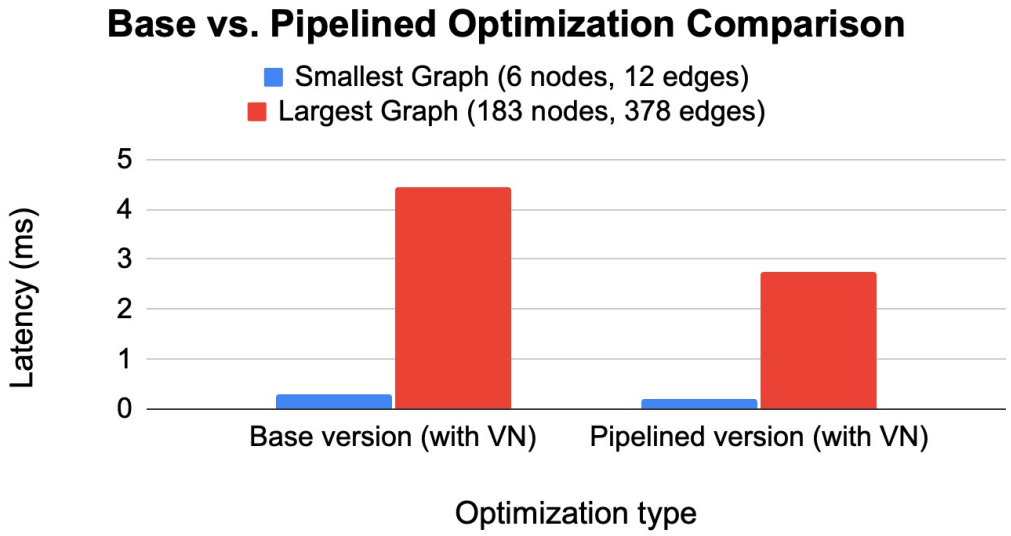
Further Optimization Opportunities

Base HLS Shortcomings:

- No pipelining
- Not scalable for large graphs
- Needs aggressive partitioning to further reduce latency

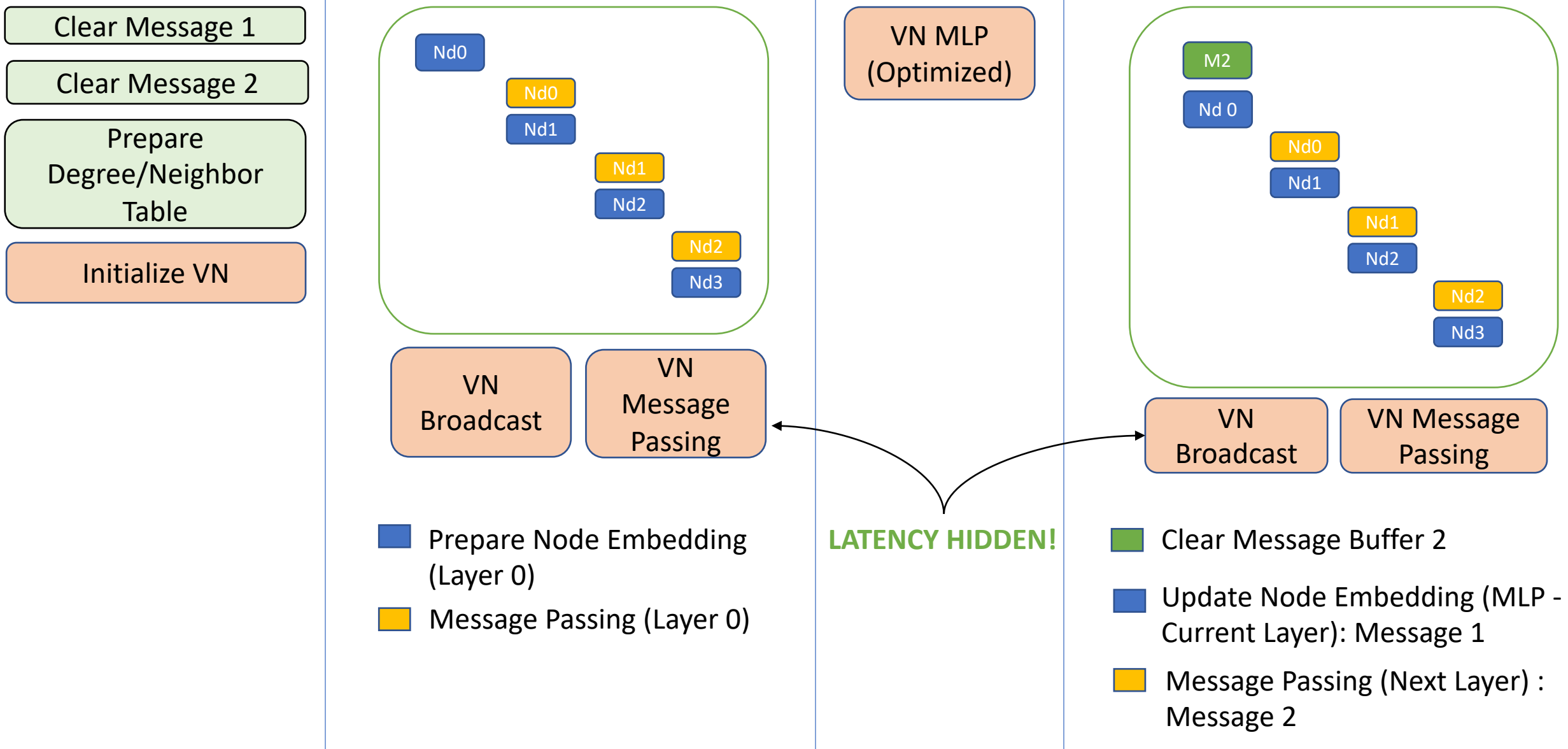
Pipelining to the rescue:

- Inter-layer pipelining
- Suitable for graphs of all sizes
- Requirements:
  - Degree and neighbor table
  - Double buffering

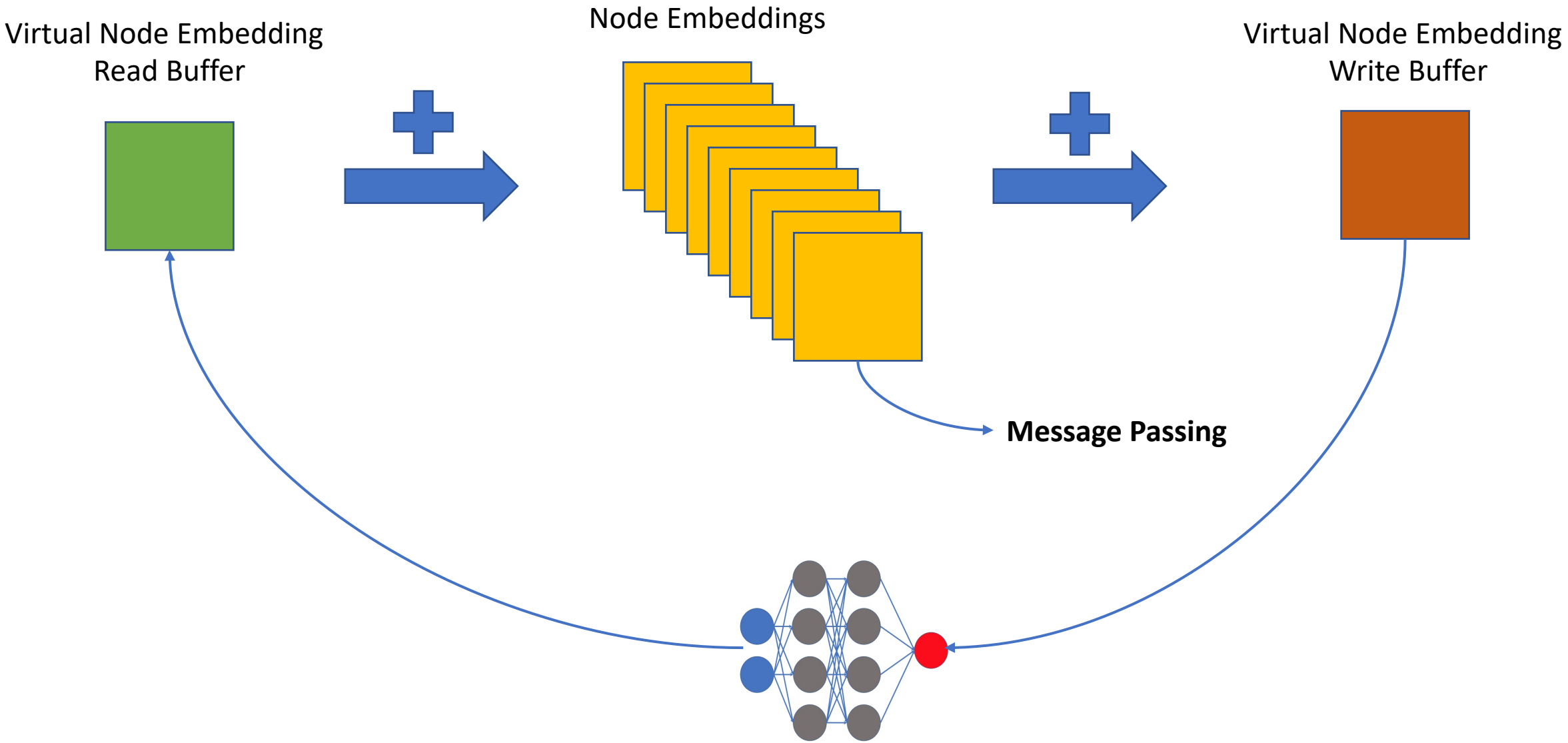


**Note: Pipelined optimization latency is obtained by averaging min-max latencies.*
**Graphs from ogbg-molhiv dataset*

Pipelined HLS with Virtual Node

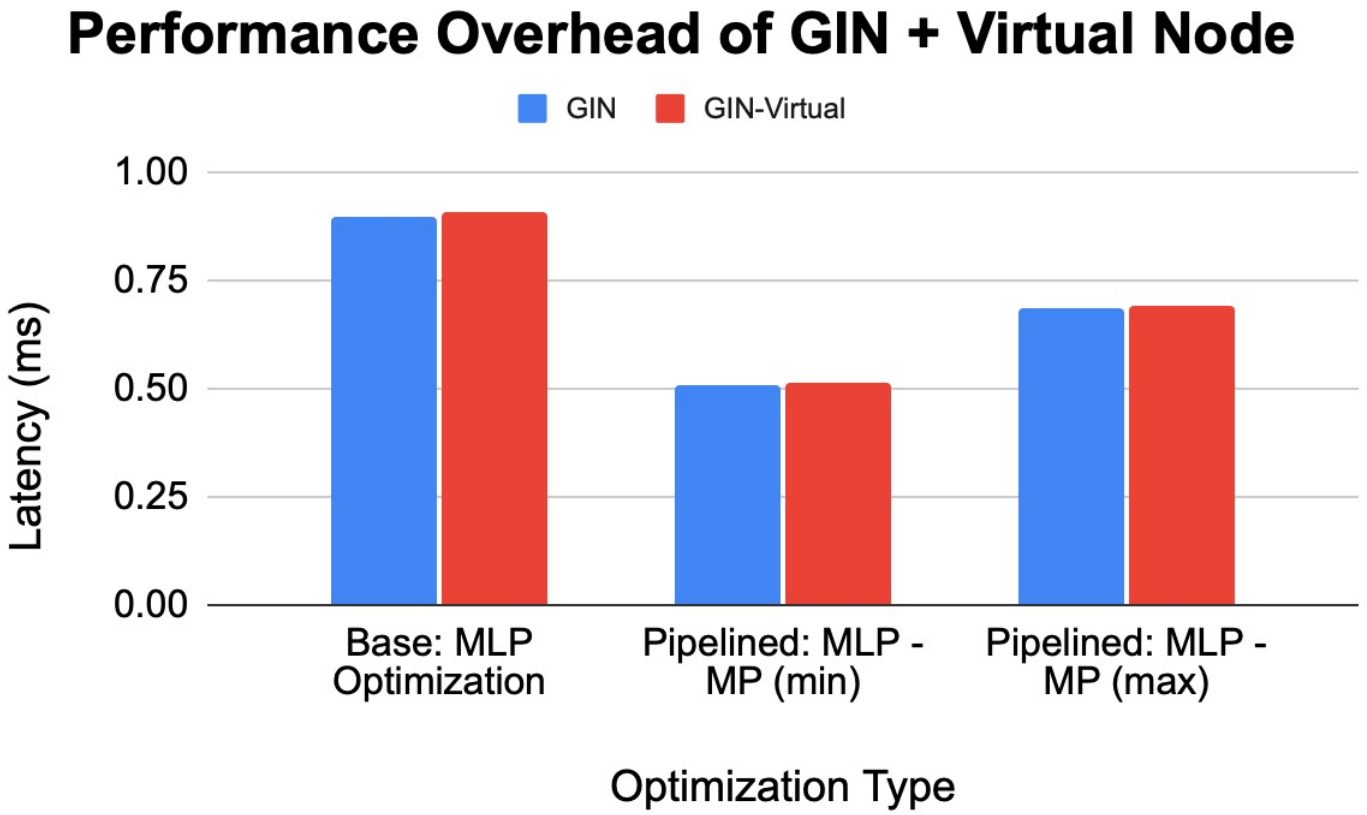


Read-Write Buffers for Virtual Node



Latency Overhead Analysis

Board	Dataset	Graph
Xilinx Alveo U280	ogbg-molhiv	19 nodes, 40 edges



Latency Overhead of **~10 us** due to Virtual Node MLP

Test rocauc (1 epoch):
GIN: 66.4
GIN-Virtual: 68.05

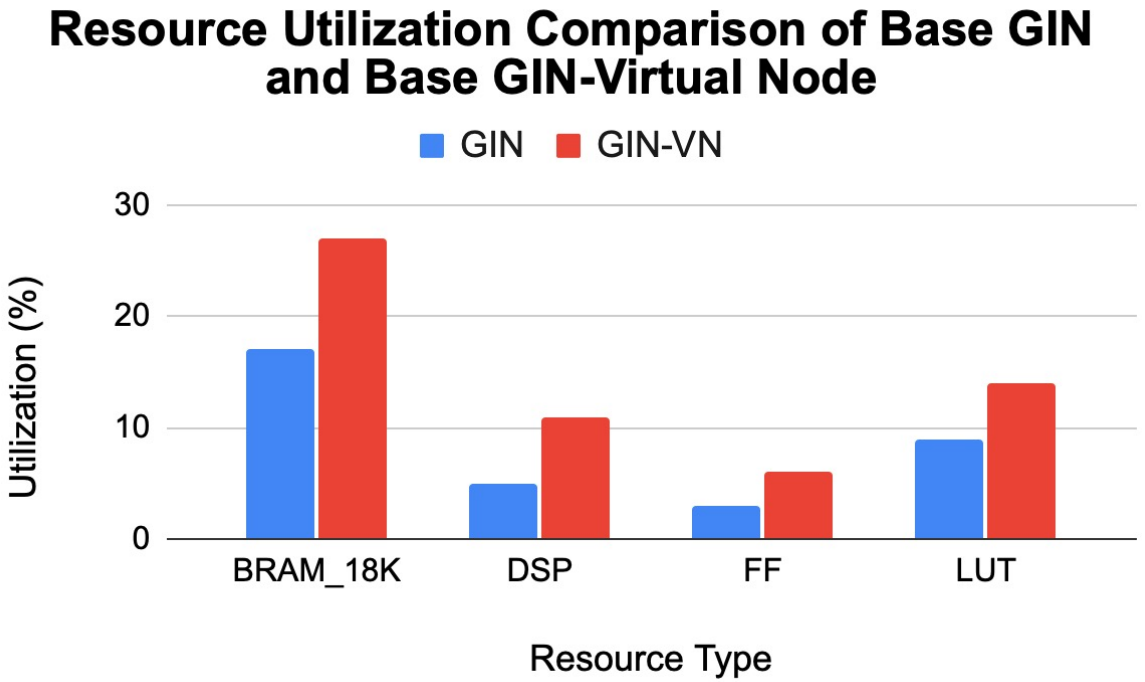
Note: Net compute time estimates obtained from Vitis HLS 2021.1

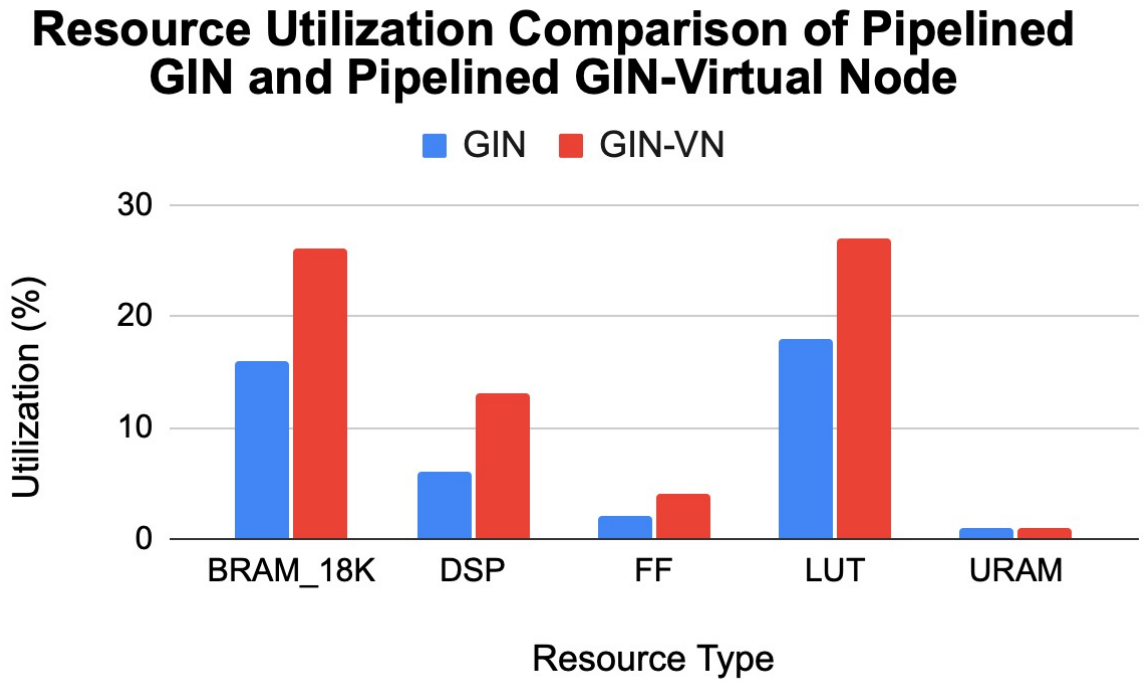
Area Overhead Analysis

- Graph from ogbg-molhiv dataset: 19 nodes, 40 edges

Total Available Resource on Xilinx Alveo U280

BRAM_18K	DSP	FF	LUT	URAM
4032	9024	2607360	1303680	960





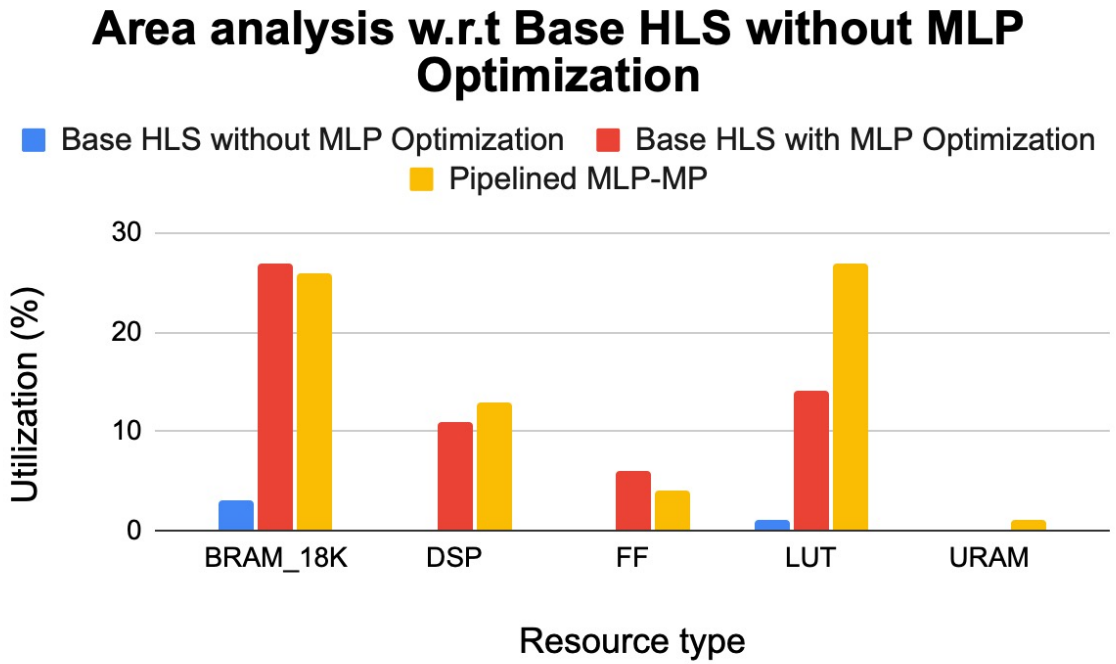
Note: Resource Utilization estimates obtained from Vitis HLS 2021.1

VN: Speed up and Area Analysis

- Comparison against Base HLS (without MLP Optimization) augmented with Virtual Node
- Graph from ogbg-molhiv dataset: 19 nodes, 40 edges

Optimization	None	MLP	Pipelined MLP-MP
Latency (ms)	50.116	0.907	0.6043
Speedup	1	55.25x	82.93x

**Note: Pipelined optimization latency is obtained by averaging min-max latencies.*



Note: Net Compute Time and Resource Utilization on U280 estimates obtained from Vitis HLS 2021.1

Future Work

Multiple VN

Increase the number of virtual nodes and analyze the impact on accuracy, performance, and resource utilization

Intra-layer Pipelining

Explore pipelining between message passing and MLP within the same layer by processing the message for a given destination

Large Graphs

Pipeline loading graphs from DRAM with message passing and explore data streaming

Thank You!

***Acknowledgement**: We thank Prof. Callie Hao and the Sharc Lab at Georgia Tech for providing us with a GIN model to augment Virtual Node for our project.*