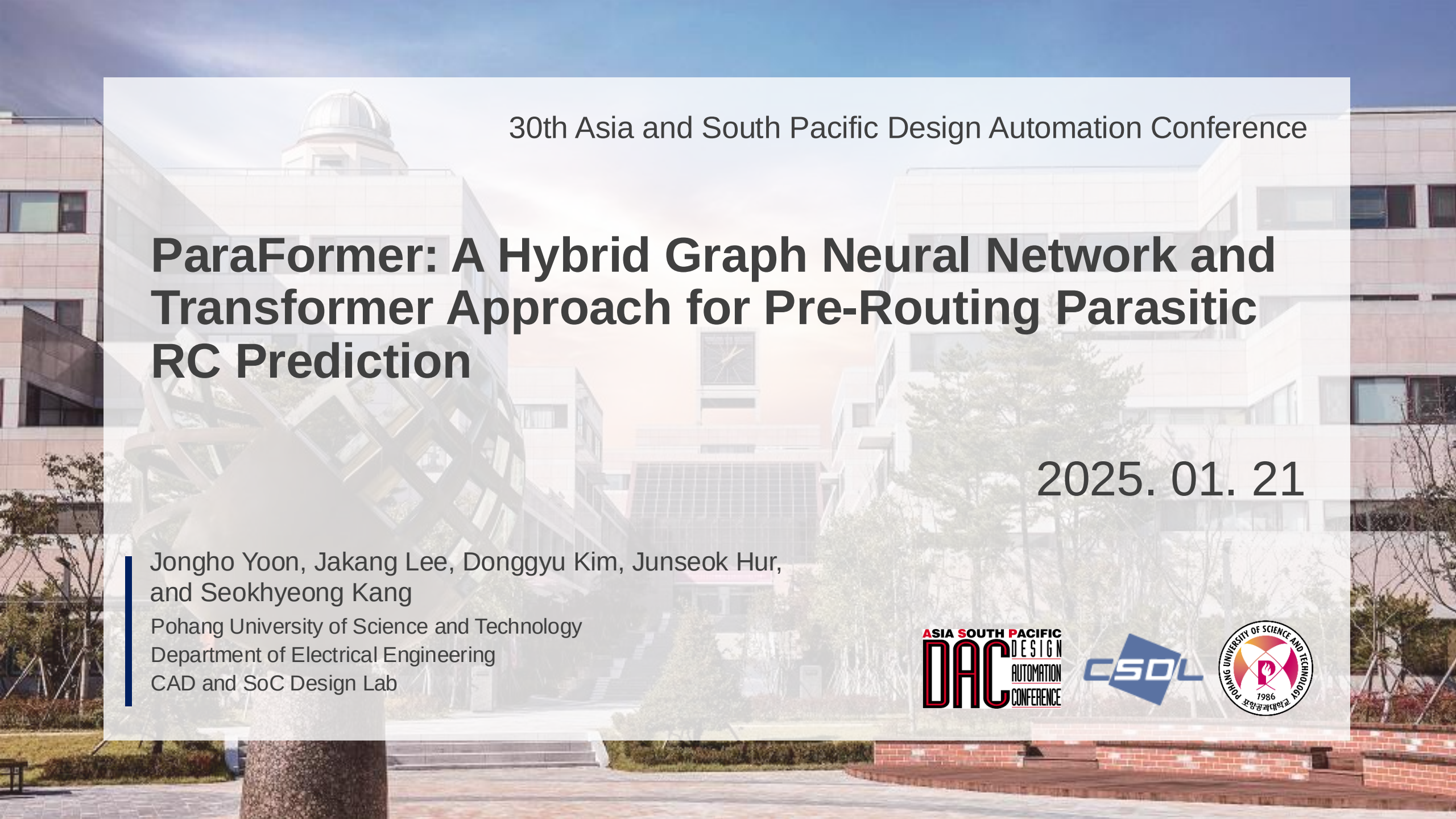


30th Asia and South Pacific Design Automation Conference

ParaFormer: A Hybrid Graph Neural Network and Transformer Approach for Pre-Routing Parasitic RC Prediction

2025. 01. 21

Jongho Yoon, Jakang Lee, Donggyu Kim, Junseok Hur, and Seokhyeong Kang

Pohang University of Science and Technology
Department of Electrical Engineering
CAD and SoC Design Lab

Outline

I. Introduction

II. Proposed Method

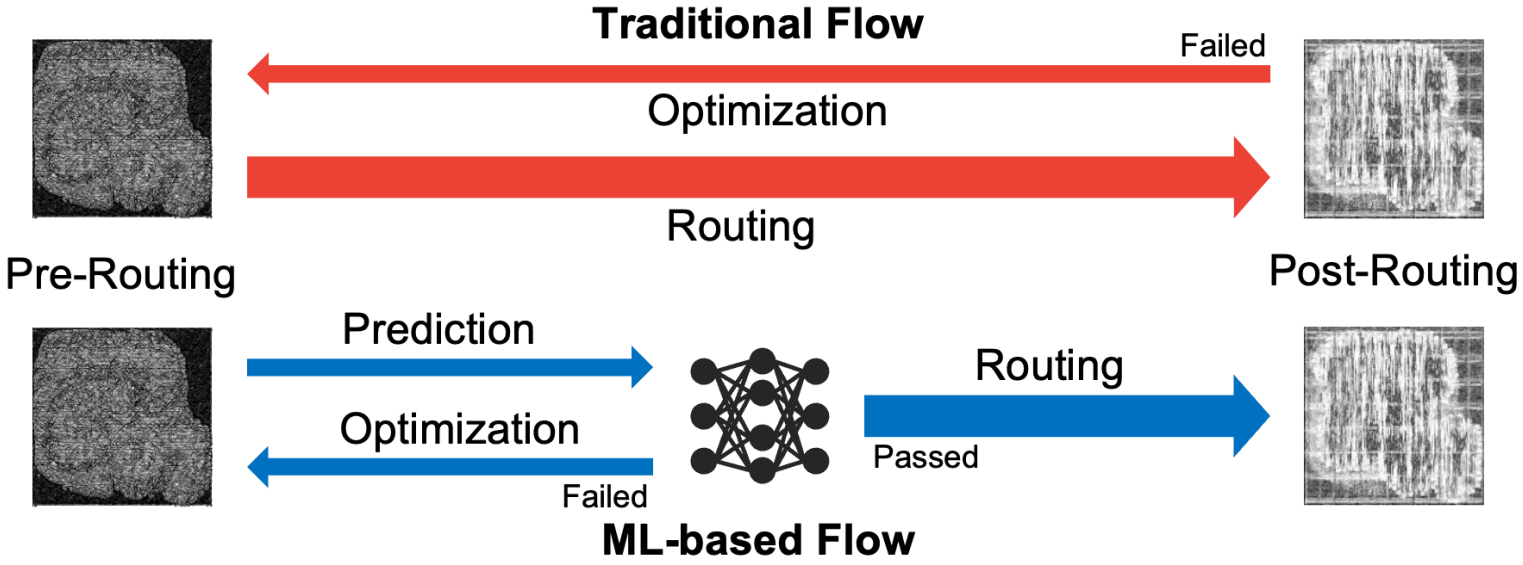
III. Experimental Setup & Result

IV. Conclusion

Need for Pre-Routing Timing Prediction

- **Why Timing Prediction is Essential:**
  - Accurate timing available only after routing
  - Routing is computationally expensive
  - Fast and accurate timing prediction reduces repetitive routing

- **Role of Machine Learning:**



Previous Works and Limitations

- **Traditional ML Approaches [1, 2]:**
  - Use only node-level features
  - Fail to capture circuit topology and geometry

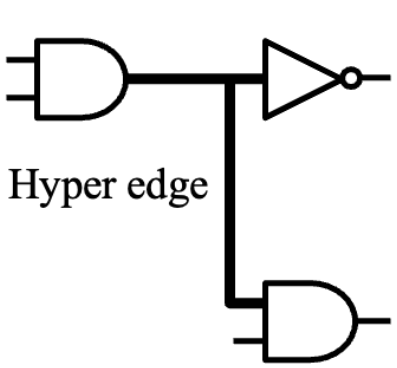
- **GNN-Based Models [3, 4]:**
  - Consider topological features using graph-based learning
  - Struggle with geometric data integration

- **Timing Prediction Models [1, 3, 4]:**
  - Errors in slew propagate across paths → Inaccuracies

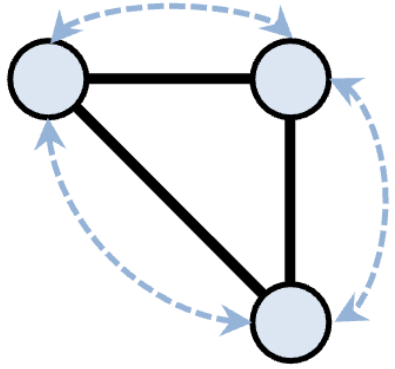
[1] E. C. Barboza, et al., “Machine Learning-Based Pre- Routing Timing Prediction with Reduced Pessimism”, DAC 2019.
[2] V. A. Chhabria, et al., “A Machine Learning Approach to Improving Timing Consistency between Global Route and Detailed Route", TODAES, 2023.
[3] Z. Guo, et al., “A Timing Engine Inspired Graph Neural Network Model for Pre-Routing Slack Prediction", DAC 2022.
[4] G. He, et al., “An Optimization-aware Pre-Routing Timing Prediction Framework Based on Heterogeneous Graph Learning", ASP-DAC 2024

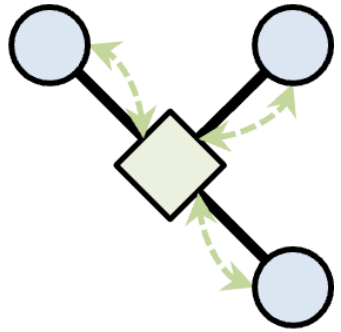
Heterogeneous Graph for Circuit Representation

- Circuit consist of diverse elements with unique roles
- **Advantages of Heterogeneous Graph:**
  - **Distinct Node Types:** Capturing unique characteristics for each elements (e.g., Pin, Net)
  - **Support Hyper-Edges:** Essential for representing netlists and aggregating information across gates
  - **Edge Diversity:** Modeling various relationships (e.g., Pin-to-Pin, Net-to-Pin)



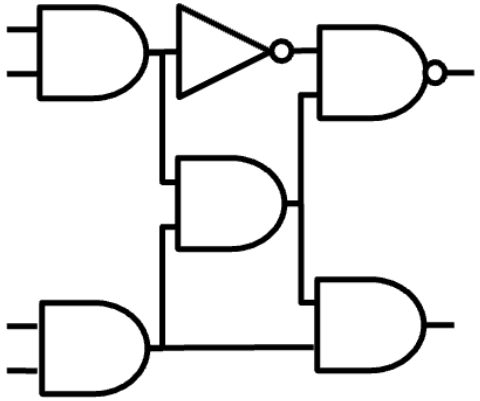
Netlist Graph


Homogeneous Graph

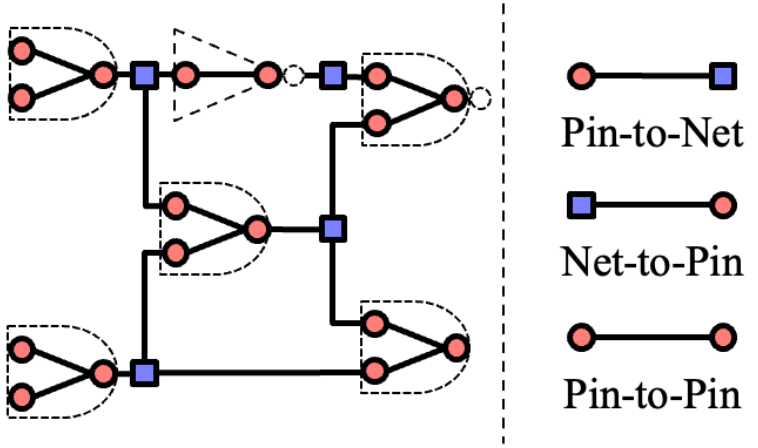

Heterogeneous Graph

Heterogeneous Circuit Graph

- **Heterogeneous Graph Components:**
  - **Nodes:**
    - **Pin:** Coordinates, Capacitance
    - **Net:** Bounding box, Area, Degree, RC values
  - **Edges:**
    - **Net-to-Pin:** Net driving multiple pins
    - **Pin-to-Pin:** Input-output connection within cell
    - **Pin-to-Net:** Pin connecting back to nets



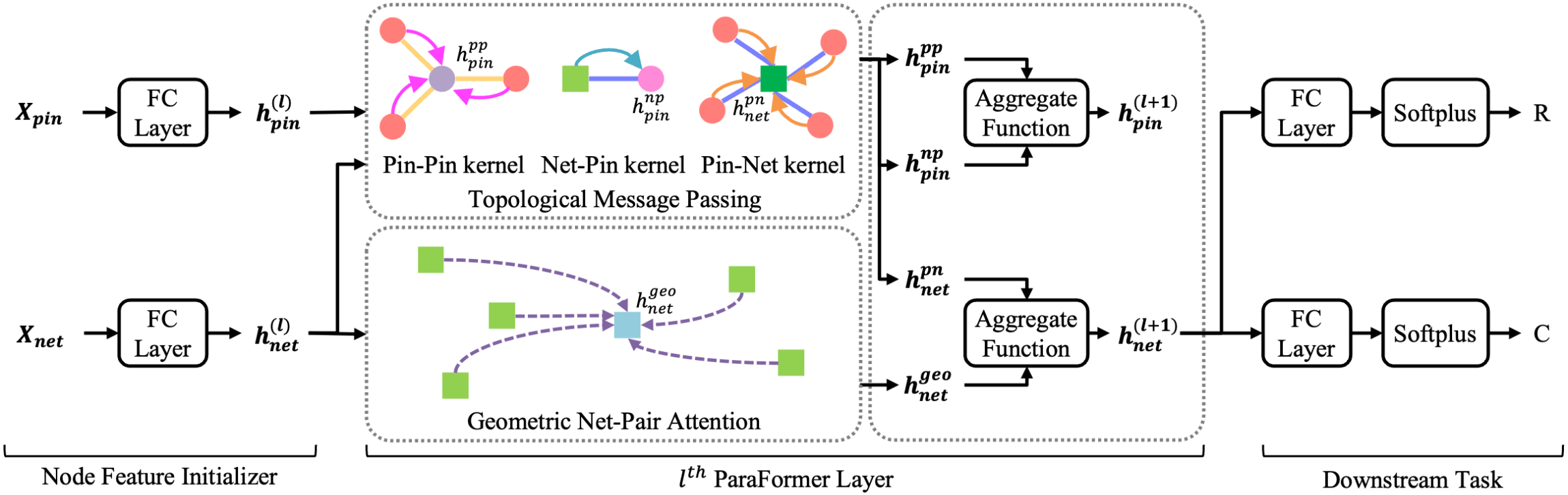
Netlist Graph



Heterogeneous Graph

ParaFormer Architecture Overview

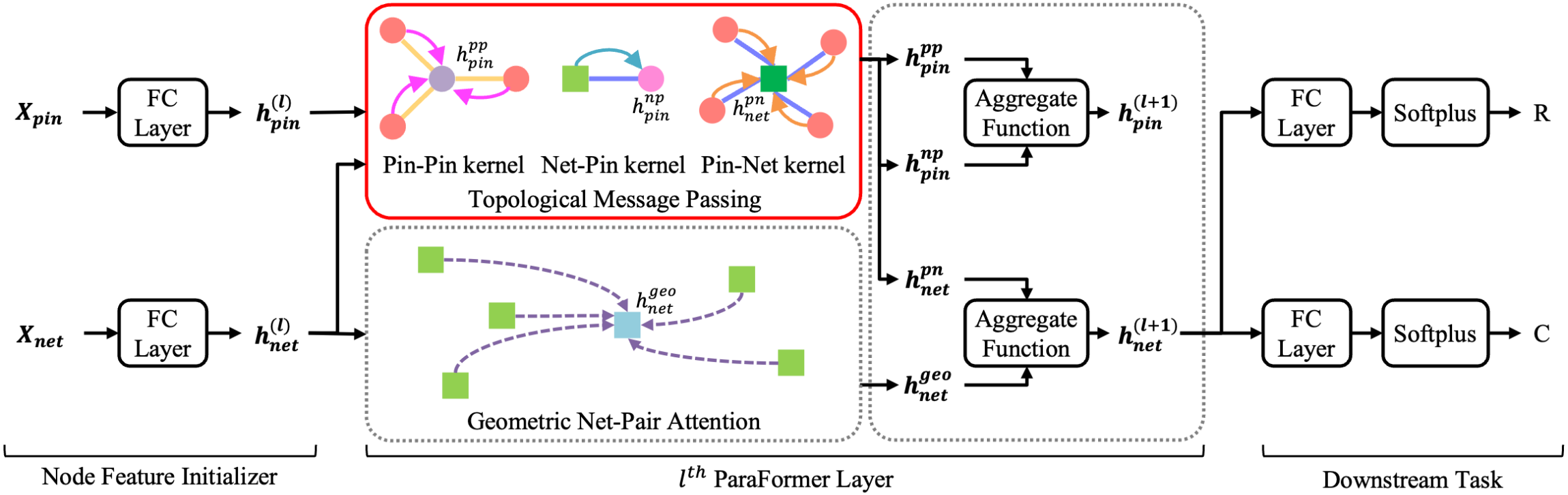
- Combine **HGNN** and **Graph Transformer** to predict parasitic RC by leveraging both **topological** and **geometric** information



Overview of ParaFormer Architecture

ParaFormer Architecture Overview

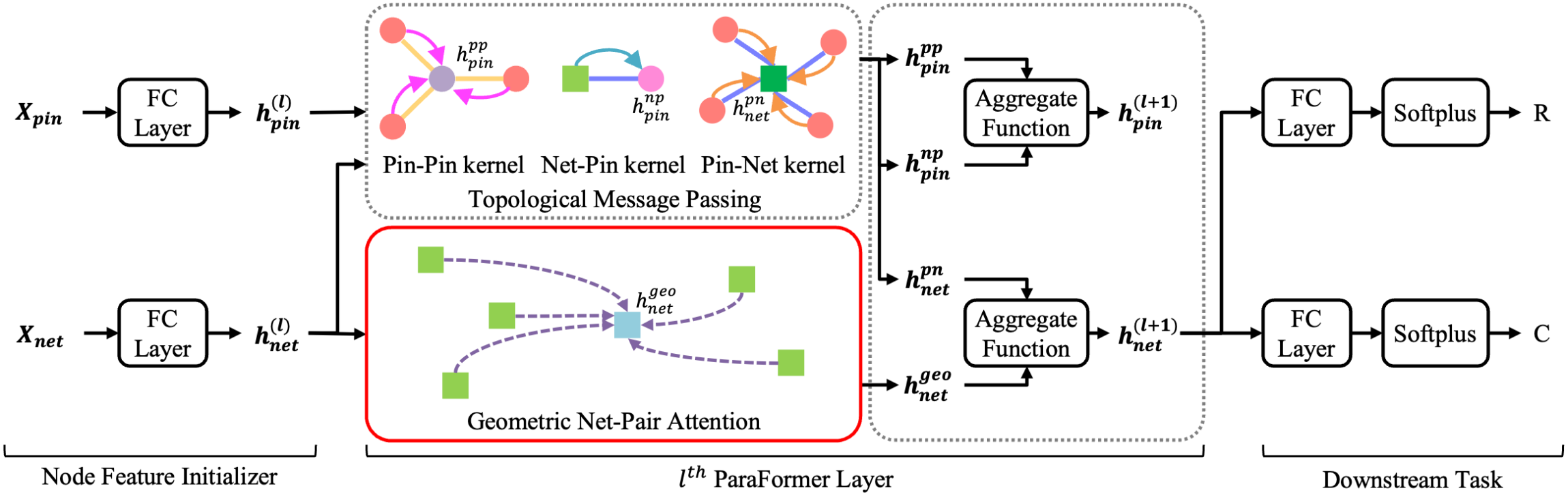
- **HGNN in ParaFormer:**
  - Extract **topological features** from heterogeneous circuit graphs using **edge-specific message passing**



Overview of ParaFormer Architecture

ParaFormer Architecture Overview

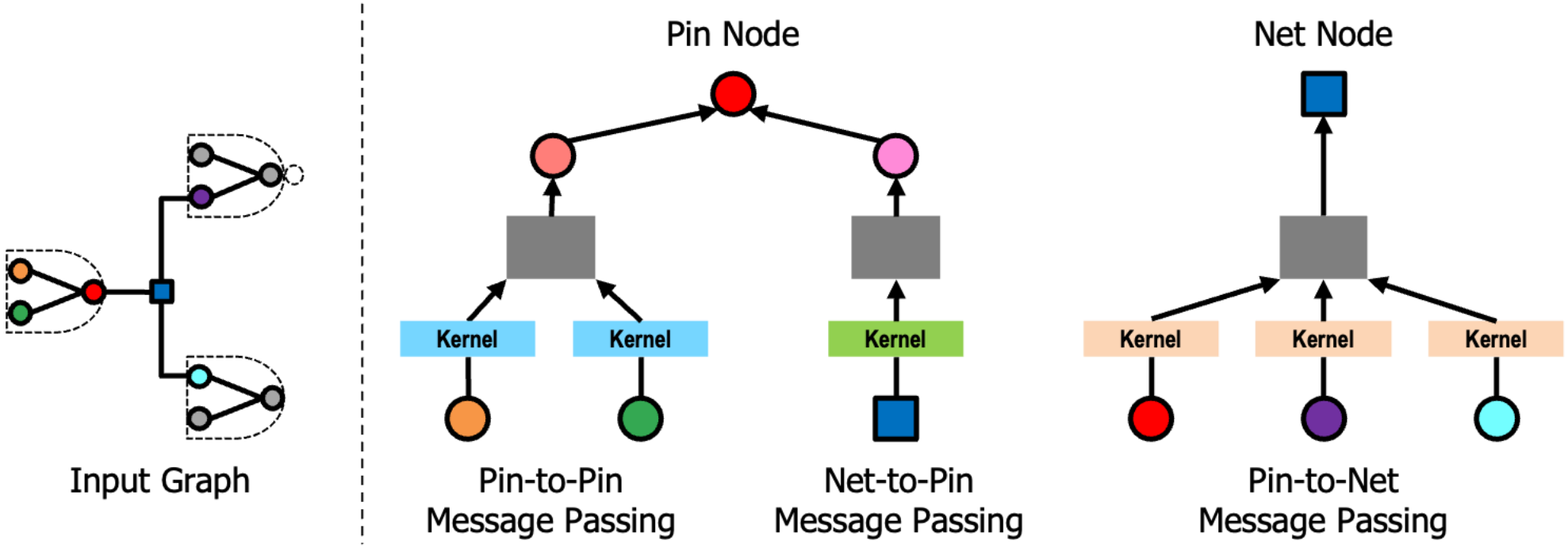
- **Transformer in ParaFormer:**
  - Capture **geometric relationships** between nets using **pairwise attention mechanism**



Overview of ParaFormer Architecture

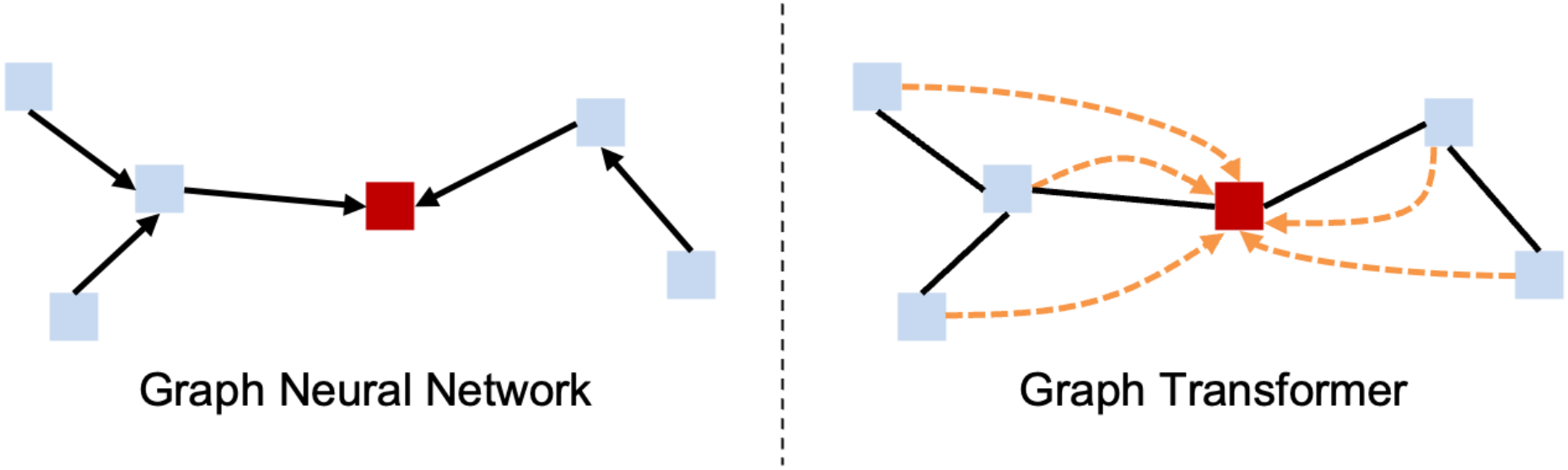
Heterogeneous Graph Neural Network

- **How Heterogeneous GNN (HGNN) Works:**
  - Separate kernels for each edge type
  - Learn specific relationships by tailored message passing
  - Aggregate node embeddings based on edge-specific relationships



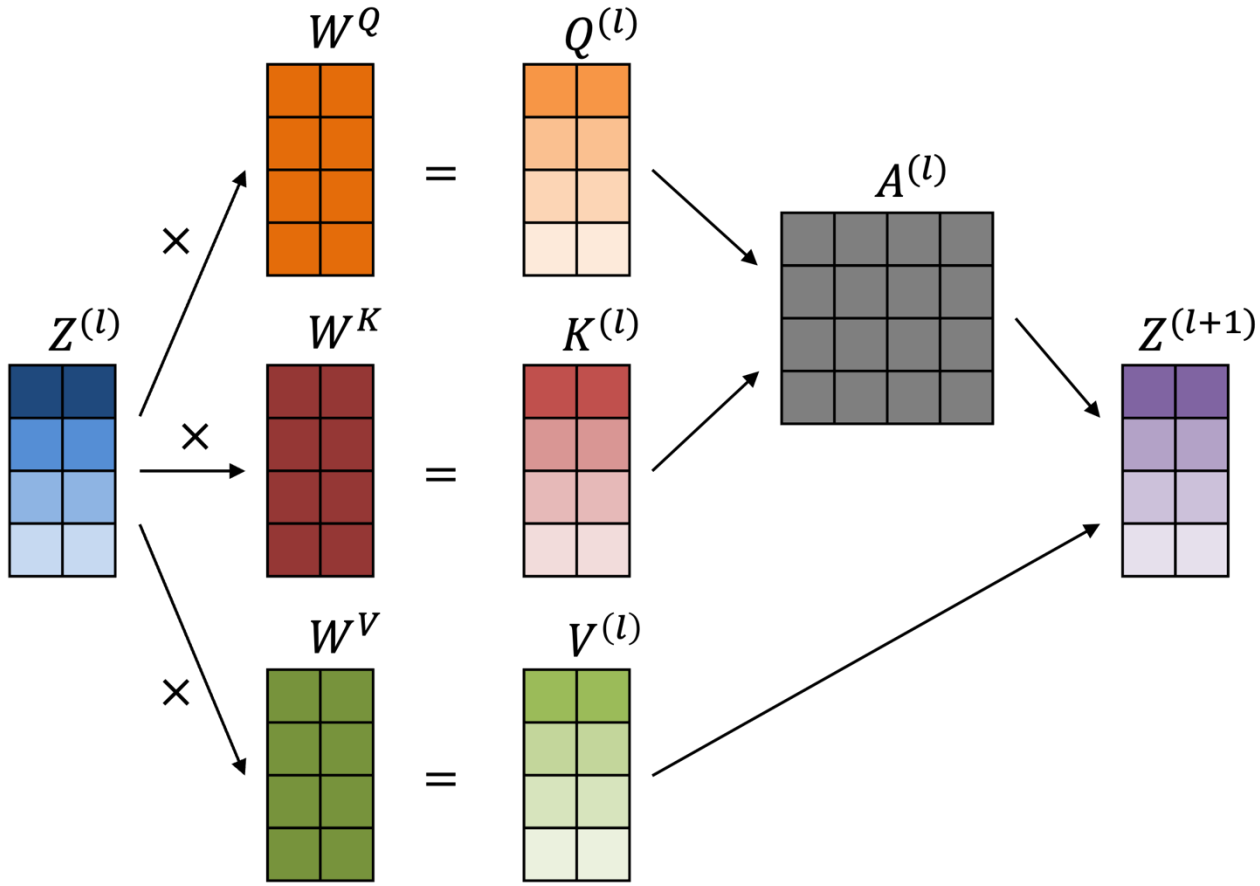
Graph Transformer for Geometric Learning

- **Why Graph Transformer?**
  - HGNN captures **topology** but not **geometry**
  - Geometric relationships affect parasitic RC significantly
  - Transformer models **pairwise relationships** between nets
- **How It Works:**



Graph Attention Mechanism

$$Z_u^{(l+1)} = \sum_{v=1}^{N} \frac{exp\left(Q_u^{(l)} \cdot K_v^{(l)}\right)}{\sum_{w=1}^{N} exp\left(Q_u^{(l)} \cdot K_w^{(l)}\right)} \cdot V_v^{(l)}$$

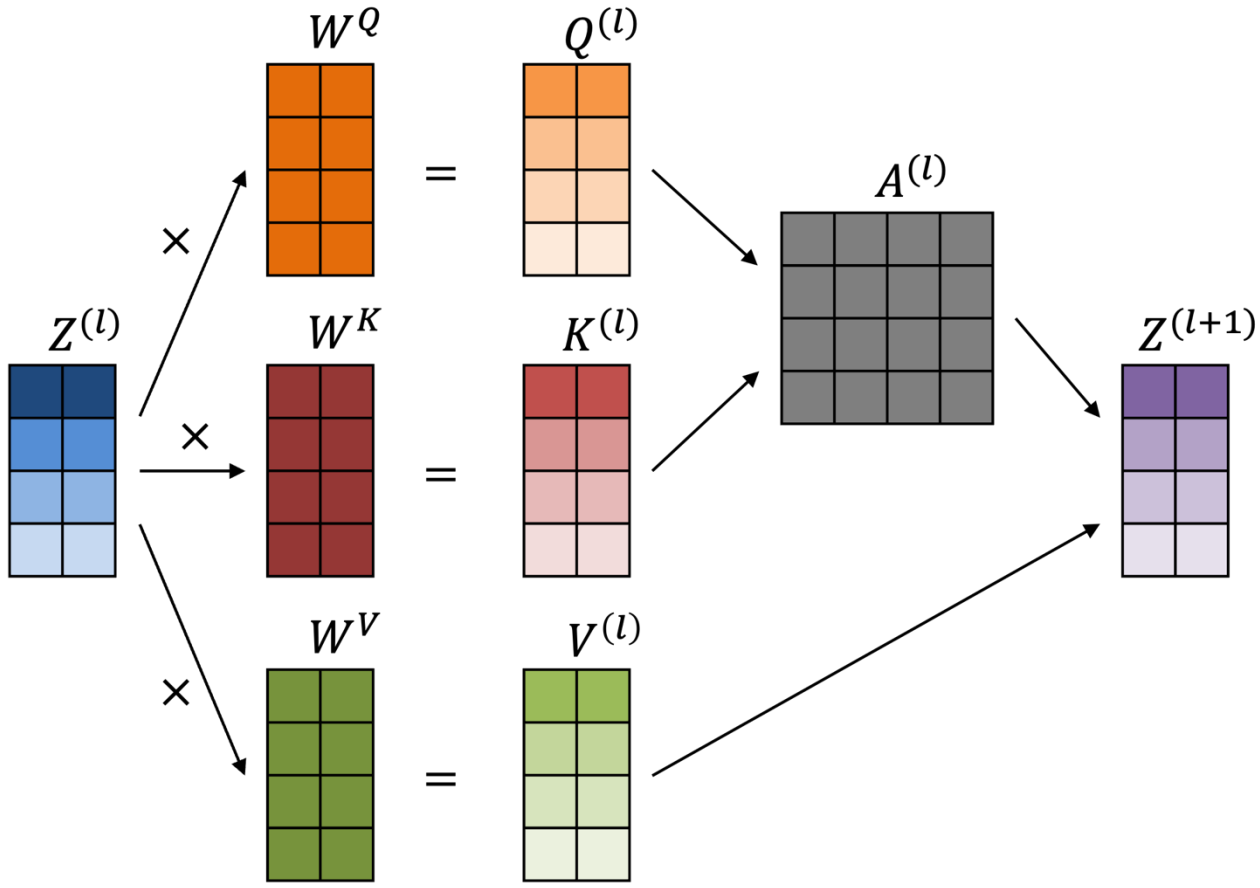


Attention Mechanism in Transformer

- Quadratic complexity $\mathcal{O}(n^2)$ limits scalability for large design

Graph Attention Mechanism

$$Z_u^{(l+1)} = \sum_{v=1}^{N} \frac{exp\left(Q_u^{(l)} \cdot K_v^{(l)}\right)}{\sum_{w=1}^{N} exp\left(Q_u^{(l)} \cdot K_w^{(l)}\right)} \cdot V_v^{(l)}$$



Attention Mechanism in Transformer

- Quadratic complexity $\mathcal{O}(n^2)$ limits scalability for large design

Scalable All-Pair Attention Message Passing

- **Kernel-Based Approximation:**
  - Replace softmax-based attention with a **kernel function** ϕ

$$Z_u^{(l+1)} = \sum_{v=1}^{N} \frac{exp\left(Q_u^{(l)} \cdot K_v^{(l)}\right)}{\sum_{w=1}^{N} exp\left(Q_u^{(l)} \cdot K_w^{(l)}\right)} \cdot V_v^{(l)}$$

$$\approx \sum_{v=1}^{N} \frac{\phi\left(Q_u^{(l)}\right) \cdot \phi\left(K_v^{(l)}\right)}{\sum_{w=1}^{N} \phi\left(Q_u^{(l)}\right) \cdot \phi\left(K_w^{(l)}\right)} \cdot V_v^{(l)}$$

Scalable All-Pair Attention Message Passing

- Kernel-based attention avoids redundant pairwise computations
- Reduce complexity from $\mathcal{O}(n^2)$ → $\mathcal{O}(n)$

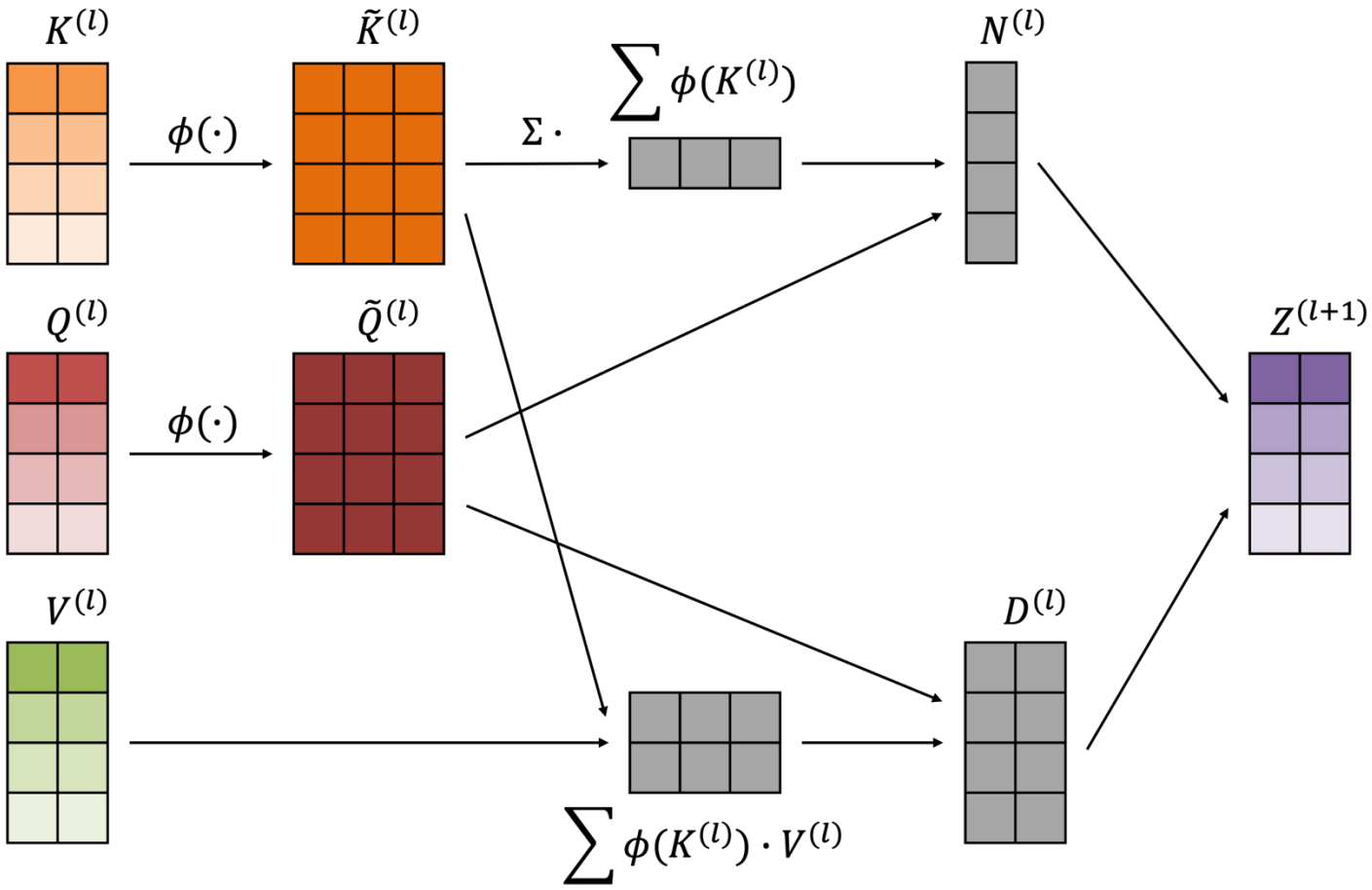
$$Z_u^{(l+1)} \approx \sum_{v=1}^{N} \frac{\phi\left(Q_u^{(l)}\right) \cdot \phi\left(K_v^{(l)}\right)}{\sum_{w=1}^{N} \phi\left(Q_u^{(l)}\right) \cdot \phi\left(K_w^{(l)}\right)} \cdot V_v^{(l)}$$

$$= \frac{\phi\left(Q_u^{(l)}\right) \cdot \sum_{v=1}^{N} \phi\left(K_v^{(l)}\right) \cdot V_v^{(l)}}{\phi\left(Q_u^{(l)}\right) \cdot \sum_{w=1}^{N} \phi\left(K_w^{(l)}\right)}$$

Scalable All-Pair Attention Message Passing

- No need for $N \times N$ attention matrix

$$Z_u^{(l+1)} \approx \frac{\phi\left(Q_u^{(l)}\right) \cdot \sum_{v=1}^{N} \phi\left(K_v^{(l)}\right) \cdot V_v^{(l)}}{\phi\left(Q_u^{(l)}\right) \cdot \sum_{w=1}^{N} \phi\left(K_w^{(l)}\right)}$$

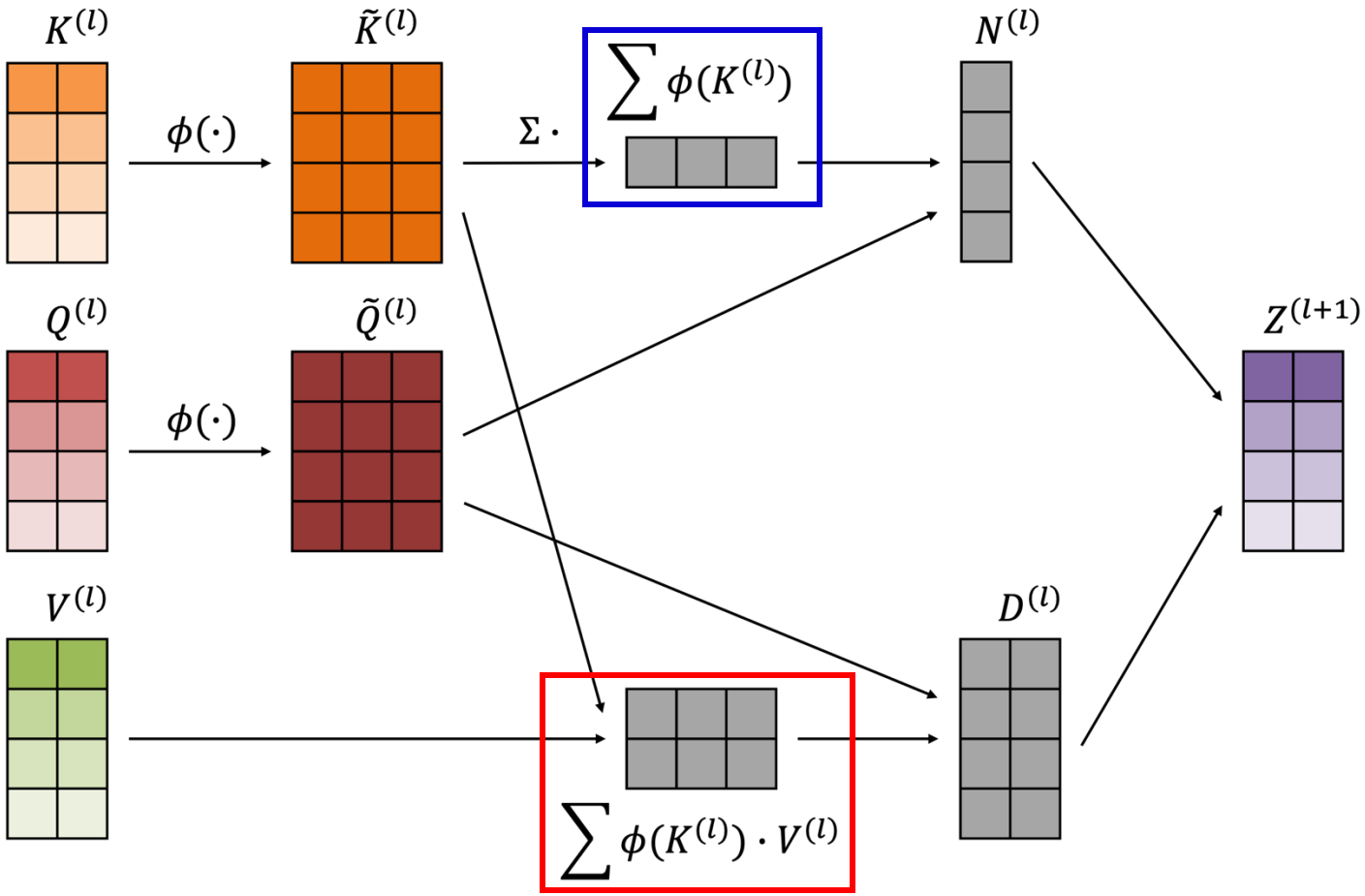


Scalable Attention Mechanism

Scalable All-Pair Attention Message Passing

- No need for $N \times N$ attention matrix

$$Z_u^{(l+1)} \approx \frac{\phi\left(Q_u^{(l)}\right) \cdot \sum_{v=1}^{N} \phi\left(K_v^{(l)}\right) \cdot V_v^{(l)}}{\phi\left(Q_u^{(l)}\right) \cdot \sum_{w=1}^{N} \phi\left(K_w^{(l)}\right)}$$

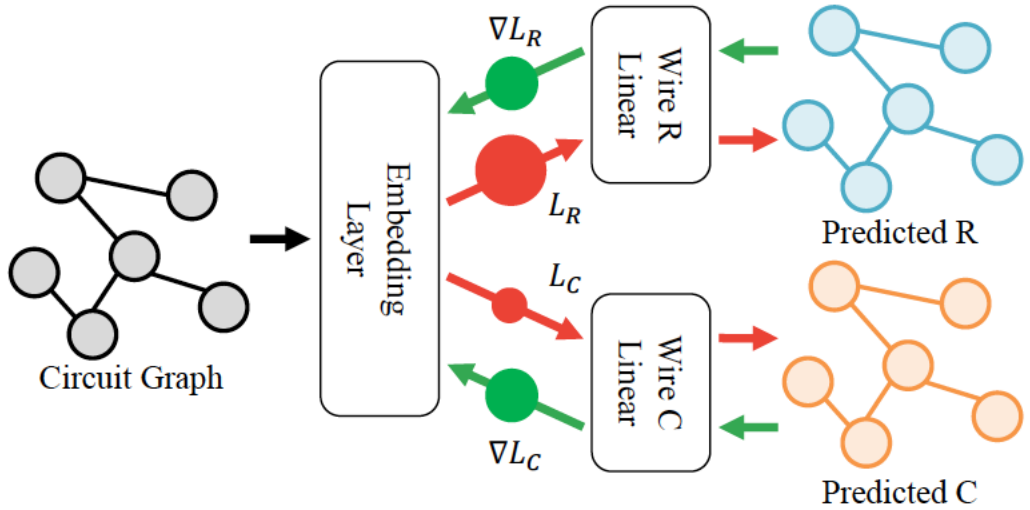


Scalable Attention Mechanism

Task Balancing with Gradient Normalization

- **Why Task Balancing?**
  - Predict **Resistance (R)** and **Capacitance (C)** simultaneously
  - Different loss scales can lead to **imbalanced learning**
- **GradNorm Technique**
  - Dynamically balances training rates
  - Adjust gradient for each task

$$w_i^{(t+1)} = w_i^{(t)} \cdot \left(\frac{\|G_i\|}{\frac{1}{T}\sum_{j=1}^{T}\|G_j\|} \right)^{\gamma}$$



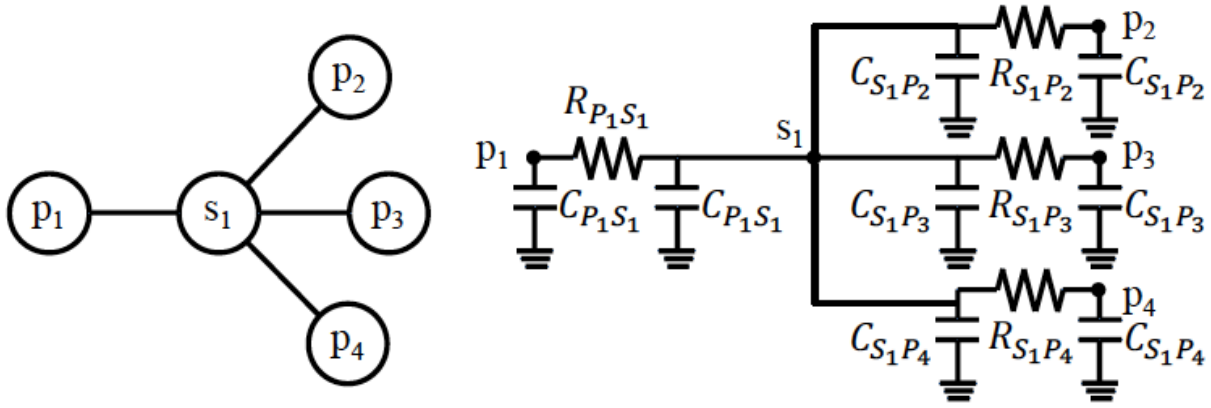
Operation of GradNorm

RC Tree Modeling for SPEF Generation

- **RC Tree Construction Steps**
  - Each net has one driver pin and multiple sink pins
  - Virtual point is placed at the center of the net bounding box
  - RC values assigned using Manhattan distance

$$R_{i,j} = \frac{|x_i - x_j| + |y_i - y_j|}{\sum_{k,l \in P} |x_k - x_l| + |y_k - y_l|} \cdot R_{pred}$$

$$C_{i,j} = \frac{|x_i - x_j| + |y_i - y_j|}{\sum_{k,l \in P} |x_k - x_l| + |y_k - y_l|} \cdot C_{pred}$$



RC Tree Construction and Modeling

Experimental Setup

- **Benchmark Dataset:**
  - Open-source designs from OpenCores [5]
  - 9 training designs, 4 test designs
  - 20 variations per design
  - Designed by *Cadence Innovus*
  - Using ASAP7 PDK [6]
- **Model Configuration:**
  - Framework: PyTorch + DGL
  - GNN Layer: GraphSAGE [7]
  - 2-layer architecture with 16 hidden units

	Design	# Cells	# Nets	# Pins
D1	*spi_top*	1829	1771	4628
D2	*usbf_top*	7764	7896	17759
D3	*aes_cipher_top*	9366	9237	26526
D4	*fpu*	17206	19362	36833
D5	*wb_commax_top*	18040	16624	46984
D6	*eth_top*	33191	33270	98527
D7	*ldpc_decoder_802_3an*	43289	41240	96465
D8	*des3*	59428	59364	128221
D9	*mpeg2_top*	376811	384545	1101167
D10	*pci_bridge32*	10661	10503	27143
D11	*keccak*	25771	25257	55027
D12	*vga_enh_top*	55145	55060	132504
D13	*tate_pairing*	167474	166309	489825

[5] OpenCores: “Open Source IP-Cores,” http://www.opencores.org.
[6] L. T. Clark, et al., “ASAP: A 7-nm finFET predictive process deisgn kit,” Microelectronics Journal, 2016.
[7] W. L. Hamilton, et al., “Inductive Representation Learning on Large Graphs,” NIPS 2017.

Parasitic RC Prediction

- **ParaFormer Results (R^2 score):**
  - R: **Ours (0.9901)** > GNN (0.9728) > HGNN (0.9717) >> ML (0.8624)
  - C: **Ours (0.9630)** > HGNN (0.9354) > GNN (0.9080) >> ML (0.8863)
- **Why ParaFormer Performed Better:**
  - ML: Lacked structural information →Limited generalization
  - GNN/HGNN: Missed geometric data → Reduced accuracy

Design	Wire Parasitic R				Wire Parasitic C			
	ML	GNN	HGNN	Ours	ML	GNN	HGNN	Ours
D10	0.9794	0.9869	0.9797	**0.9904**	0.9420	0.9282	0.9722	**0.9871**
D11	0.9431	0.9597	0.9521	**0.9812**	0.9278	0.8227	0.8384	**0.9318**
D12	0.9214	0.9677	0.9669	**0.9954**	0.9618	0.9455	**0.9707**	0.9677
D13	0.6056	0.9769	0.9880	**0.9934**	0.7137	0.9355	0.9601	**0.9656**
Train avg.	0.9872	0.9737	0.9721	0.9854	0.9861	0.9269	0.9532	0.9701
Test avg.	0.8624	0.9728	0.9717	**0.9901**	0.8863	0.9080	0.9354	**0.9630**

Timing & Power Analysis using Commercial Tool

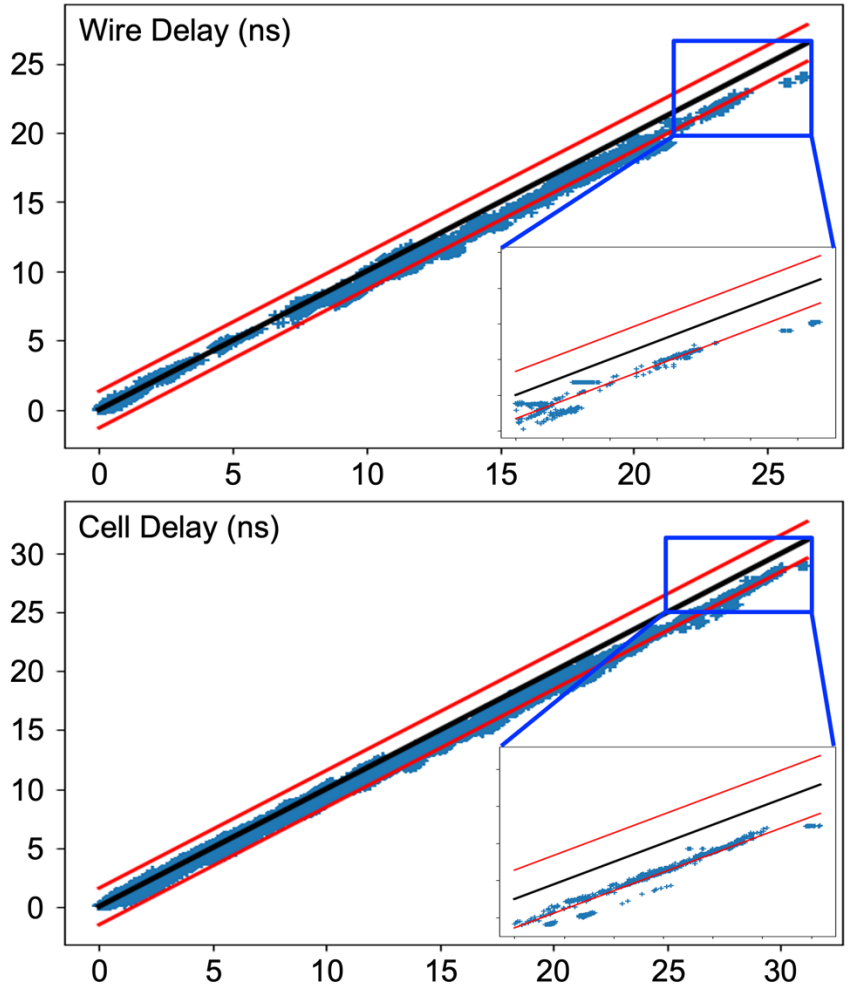
- Analyze timing and power using *Synopsys PrimeTime*
- Extract SPEF file using predicted results for each method
- **ParaFormer Results:**
  - Delay: $R^2 = 0.9073$ (wire) / $R^2 = 0.9346$ (cell)
  - Power: $MAPE = 3.31\ \%$

Design	Wire Delay				Cell Delay				Power			
	ML	GNN	HGNN	Ours	ML	GNN	HGNN	Ours	ML	GNN	HGNN	Ours
D10	0.8790	0.7239	**0.9084**	0.8657	0.9693	0.9492	0.9740	**0.9801**	**0.290**	0.690	0.800	0.430
D11	0.8741	0.9189	0.9243	**0.9660**	0.8609	0.9104	0.9179	**0.9637**	11.300	8.640	9.680	**5.620**
D12	0.9273	0.8427	0.9124	**0.9305**	**0.9553**	0.8649	0.9290	0.9401	5.130	4.330	**0.880**	3.210
D13	0.7246	0.8184	0.8503	**0.8670**	0.7242	0.8130	0.8374	**0.8543**	**1.590**	4.120	2.540	4.070
Train avg.	0.9262	0.7988	0.8866	0.9229	0.9843	0.9396	0.9546	0.9739	2.236	6.380	5.108	6.268
Test avg.	0.8513	0.8260	0.8989	**0.9073**	0.8774	0.8844	0.9146	**0.9346**	4.578	4.445	3.475	**3.310**

Impact of GradNorm on Downstream Tasks

- Timing & Power Analysis depend on both R and C
- **Results with GradNorm (R^2 score):**
  - R: 0.9901 → 0.9785 (w/ GradNorm)
  - C: 0.9630 → 0.9917 (w/ GradNorm)

Design	Wire Delay		Cell Delay		Power	
	w/o G	w/ G	w/o G	w/ G	w/o G	w/ G
D10	0.8657	**0.9587**	0.9801	**0.9874**	**0.340**	0.430
D11	0.9660	**0.9915**	0.9637	**0.9899**	5.620	**2.500**
D12	0.9305	**0.9714**	0.9401	**0.9928**	3.210	**0.970**
D13	0.8670	**0.9780**	0.8543	**0.9803**	4.070	**1.910**
Train avg.	0.9229	0.9493	0.9739	0.9876	6.268	2.561
Test avg.	0.9073	**0.9749**	0.9346	**0.9876**	3.310	**1.453**



Conclusion

- **ParaFormer** combines HGNN and Transformer to capture both **topological** and **geometric** features
- **GradNorm** ensures balanced R and C learning, improving results of downstream tasks
- **ParaFormer** allows for faster and efficient design optimization, reducing the need for iterative adjustments

Thank you

jh.yoon@postech.ac.kr