Lay-Net: Grafting Netlist Knowledge on Layout-Based Congestion Prediction

Su Zheng[*1], Lancheng Zou[*2], Peng Xu[1], Siting Liu[1], Bei Yu[1], Martin Wong[1]

[1] The Chinese University of Hong Kong
[2] Wuhan University

Oct. 30, 2023

Outline

1. Introduction
2. Algorithm
3. Experiments

Introduction

Placement and Congestion Modeling

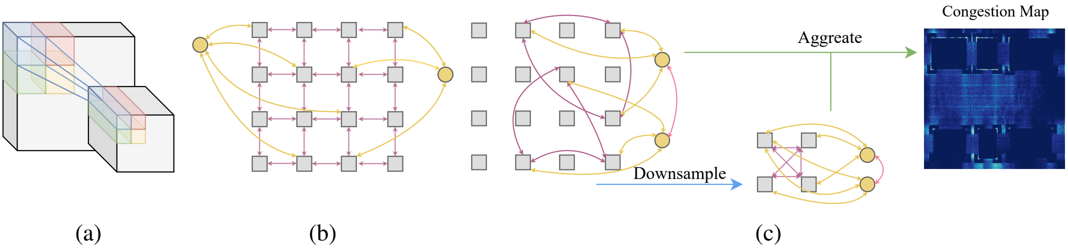
- Placement is crucial but time-consuming
- Congestion modeling
  - Fully Convolutional Networks (FCN)[1]
  - Graph Neural Networks (GNN)[2]
- Accurate congestion prediction enables better optimization!

[1] Zhiyao Xie et al. (2018). "RouteNet: Routability Prediction for Mixed-size Designs Using Convolutional Neural Network". In: *Proc. ICCAD*, 80:1–80:8.

[2] Bowen Wang et al. (2022). "LHNN: Lattice Hypergraph Neural Network for VLSI Congestion Prediction". In: *Proc. DAC*, pp. 1297–1302.

Placement and Congestion Modeling

- Existing methods
  - Image-based: local perception without global view
  - Graph-based: insufficient modeling of physical information
- What do we need? Netlist + layout!
  - Multi-modality $\rightarrow$ global view + sufficient information

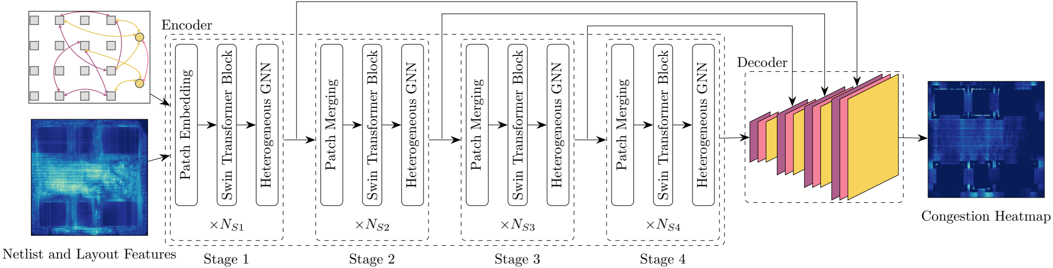


Algorithm

Overview of the Proposed Method

- Netlist + layout → congestion heatmap
  - $\mathcal{G}_H$: connestion information from the netlist
  - $\mathbf{X}$, $\mathbf{Y}$: geometric information from the layout

$$L_H(\mathcal{G}_H, \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{NM} \|\boldsymbol{f}_H(\mathcal{G}_H, \boldsymbol{X}) - \boldsymbol{Y}\|_2^2. \tag{1}$$



How to Extract Layout Information?

- Layout Features
  - RUDY:

$$\mathbf{RUDY}_e(\boldsymbol{x},\boldsymbol{y}) = (\frac{1}{x_e^h - x_e^l} + \frac{1}{y_e^h - y_e^l}), x \in [x_e^l, x_e^h], y \in [y_e^l, y_e^h]. \tag{2}$$

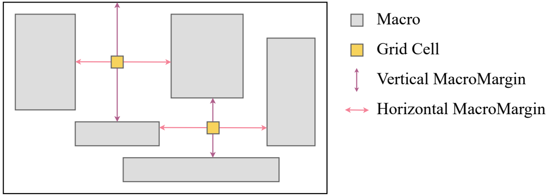
  - PinRUDY:

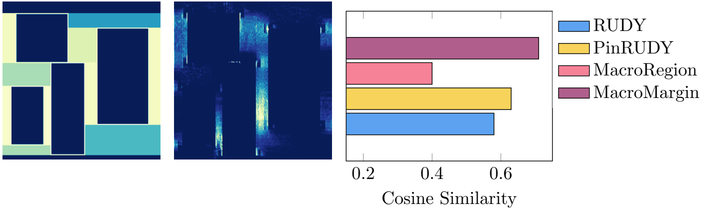
$$\mathbf{PinRUDY}_{p_e}(k,l) = (\frac{1}{x_e^h - x_e^l} + \frac{1}{y_e^h - y_e^l}), (x_{p_e}, y_{p_e}) \in b_{k,l}. \tag{3}$$

  - MacroRegion:

$$\mathbf{MacroRegion}(k,l) = \begin{cases} 1, & \text{if } b_{k,l} \text{ is in a macro cell,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

- The novel MacroMargin feature

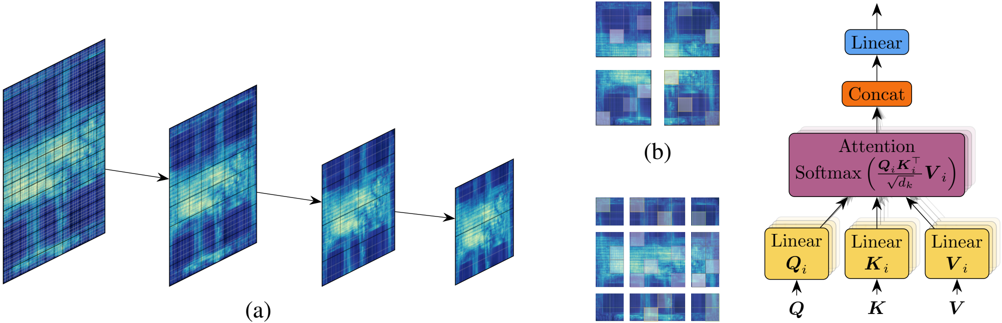




(a) (b) (c)

How to Extract Layout Information?

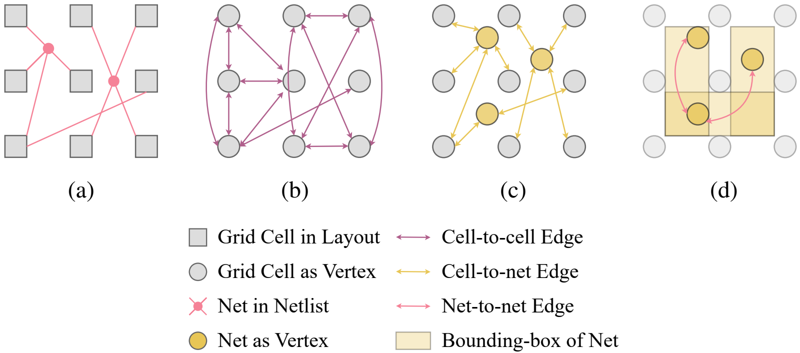
- Network Architecture
  - Multi-scale feature extraction → global view
  - Shifted-window self-attention → local perception
  - Based on Swin Transformer → good feature extractor



(a) (b) (c) (d)

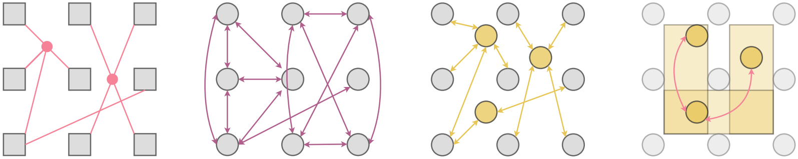
How to Extract Netlist Information?

- Graft the netlist knowledge on the layout-based features!



How to Extract Netlist Information?

- Heterogeneous Message Passing
  - Cell-to-cell Connections
  - Cell-to-net Connections
  - Net-to-net Connections

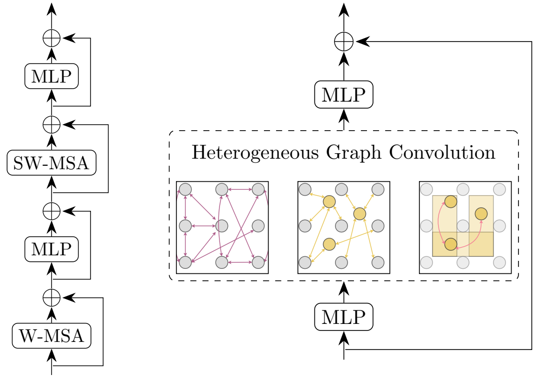


(a) (b) (c) (d)

- Note that the "cell" here refers to a grid-cell on the layout
- A grid-cell can contain multiple cells from the netlist

Graft the Netlist Knowledge on the Layout

- Swin transformer block[3] + heterogeneous message passing

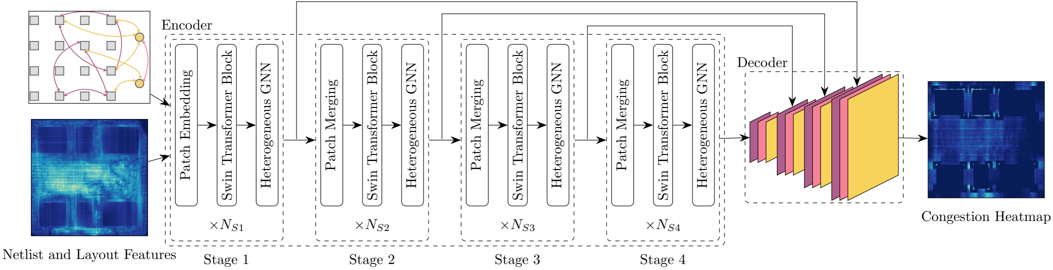


(a) (b)

[3]Ze Liu et al. (2021). “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proc. CVPR*, pp. 10012–10022.

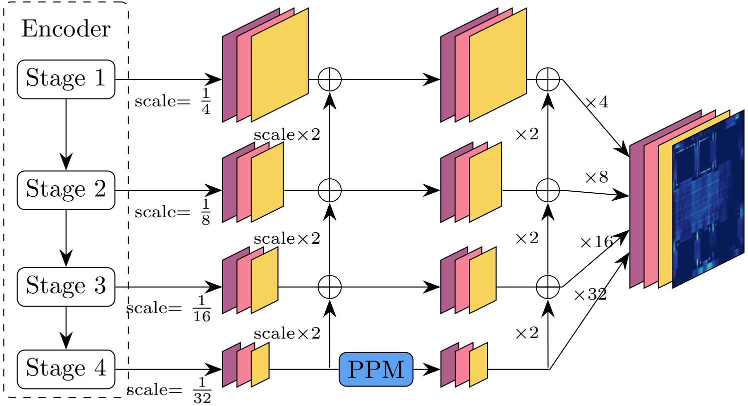
Graft the Netlist Knowledge on the Layout

- Swin transformer block + heterogeneous message passing



Graft the Netlist Knowledge on the Layout

- The Decoder: UPerNet[4]
  - Utilizing the multi-scale features



[4]Tete Xiao et al. (2018). "Unified perceptual parsing for scene understanding". In: *Proc. ECCV*, pp. 418–434.

Comparison Between Ours and Previous Methods

- RouteNet[5], GAN[6], NAS[7], Cross-Graph[8], LHNN[9], PGNN[10], CircuitGNN[11]

[5] Zhiyao Xie et al. (2018). "RouteNet: Routability Prediction for Mixed-size Designs Using Convolutional Neural Network". In: *Proc. ICCAD*, 80:1–80:8.

[6] Cunxi Yu and Zhiru Zhang (2019). "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets". In: *Proc. DAC*.

[7] Chen-Chia Chang et al. (2021). "Automatic Routability Predictor Development Using Neural Architecture Search". In: *Proc. ICCAD*.

[8] Amur Ghose et al. (2021). "Generalizable Cross-Graph Embedding for GNN-based Congestion Prediction". In: *Proc. ICCAD*.

[9] Bowen Wang et al. (2022). "LHNN: Lattice Hypergraph Neural Network for VLSI Congestion Prediction". In: *Proc. DAC*, pp. 1297–1302.

[10] Kyeonghyeon Baek et al. (2022). "Pin Accessibility and Routing Congestion Aware DRC Hotspot Prediction using Graph Neural Network and U-Net". In: *Proc. ICCAD*.

[11] Zhihao Yang et al. (2022). "Versatile Multi-stage Graph Neural Network for Circuit Representation". In: *Proc. NeurIPS* 35, pp. 20313–20324.

Comparison Between Ours and Previous Methods

- Comparing the features of different methods

Table: Comparison Between Prediction Methods

Characteristic	RUDY	Macro	No Routing	Global Info.	Cell-to-cell	Cell-to-net	Net-to-net	Multi-scale Graphs
RouteNet	✓	✓	✗	✗	✗	✗	✗	✗
GAN	✓	✓	✓	✗	✗	✗	✗	✗
NAS	✓	✓	✓	✗	✗	✗	✗	✗
Cross-Graph	✗	✗	✓	✗	✓	✗	✗	✗
LHNN	✓	✗	✓	✗	✓	✓	✗	✗
PGNN	✓	✗	✓	✗	✓	✗	✗	✗
CircuitGNN	✓	✗	✓	✗	✓	✓	✗	✗
Lay-Net	✓	✓	✓	✓	✓	✓	✓	✓

Experiments

Experimental Setup

- Dataset: ISPD 2015, half for training, half for testing
- Structural similarity (SSIM) ↑:

$$\mathrm{SSIM}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{(2\mu_{\boldsymbol{Y}}\mu_{\overline{\boldsymbol{Y}}} + C_1)(2\sigma_{\boldsymbol{Y},\overline{\boldsymbol{Y}}} + C_2)}{(\mu_{\boldsymbol{Y}}^2 + \mu_{\overline{\boldsymbol{Y}}}^2 + C_1)(\sigma_{\boldsymbol{Y}}^2 + \sigma_{\overline{\boldsymbol{Y}}}^2 + C_2)}. \tag{5}$$

- Normalized root mean square error (NRMS) ↓:

$$\mathrm{NRMS}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{\|\overline{\boldsymbol{Y}} - \boldsymbol{Y}\|_2}{(Y_{\max} - Y_{\min})\sqrt{N_Y}}, \tag{6}$$

- Score ↑:

$$\mathrm{Score}(\overline{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{\mathrm{SSIM}(\overline{\boldsymbol{Y}}, \boldsymbol{Y})}{\mathrm{NRMS}(\overline{\boldsymbol{Y}}, \boldsymbol{Y})}. \tag{7}$$

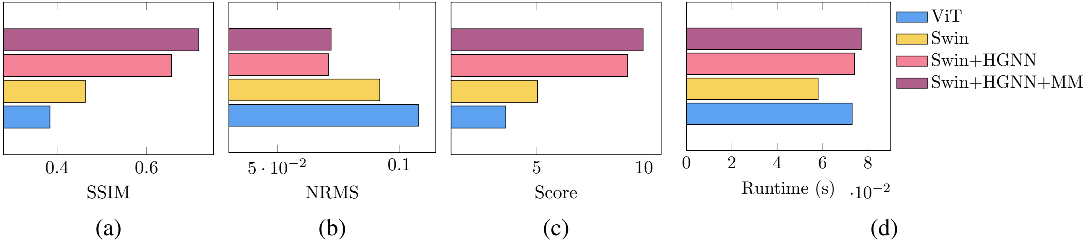
Comparison Between Ours and Previous Methods

Table: Comparison Between Lay-Net and Previous Methods on ISPD 2015 Benchmark

Benchmark	#Cells	#Nets	Part	RouteNet			GAN			LHNN			Lay-Net		
				SSIM	NRMS	Score	SSIM	NRMS	Score	SSIM	NRMS	Score	SSIM	NRMS	Score
des_perf_1	113k	113k	B	0.364	0.087	4.183	0.442	0.076	5.815	0.716	0.100	7.159	0.721	0.068	10.60
des_perf_a	109k	110k	A	0.499	0.072	6.930	0.542	0.081	6.691	0.789	0.079	9.987	0.778	0.061	12.75
des_perf_b	113k	113k	A	0.499	0.069	7.231	0.531	0.085	6.247	0.863	0.064	13.48	0.851	0.053	16.05
edit_dist_a	130k	131k	A	0.464	0.091	5.098	0.491	0.109	4.504	0.777	0.089	8.730	0.772	0.068	11.35
fft_1	35k	33k	A	0.432	0.087	4.965	0.482	0.102	4.725	0.753	0.079	9.531	0.755	0.060	12.58
fft_2	35k	33k	A	0.465	0.083	5.602	0.494	0.100	4.939	0.775	0.085	9.117	0.771	0.063	12.23
fft_a	34k	32k	A	0.470	0.105	4.476	0.489	0.114	4.289	0.651	0.113	5.761	0.826	0.094	8.787
fft_b	34k	32k	B	0.337	0.096	3.510	0.494	0.085	5.811	0.814	0.074	11.00	0.801	0.059	13.57
matrix_mult_1	160k	159k	B	0.325	0.091	3.571	0.383	0.088	4.352	0.526	0.112	4.696	0.530	0.092	5.760
matrix_mult_2	160k	159k	B	0.375	0.083	4.518	0.435	0.077	5.649	0.669	0.105	6.371	0.676	0.070	9.657
matrix_mult_a	154k	154k	B	0.391	0.089	4.393	0.451	0.085	5.305	0.599	0.092	6.510	0.603	0.088	6.852
matrix_mult_b	151k	152k	B	0.422	0.092	4.586	0.493	0.081	6.086	0.708	0.173	4.092	0.715	0.070	10.21
matrix_mult_c	151k	152k	B	0.366	0.090	4.066	0.443	0.081	5.469	0.660	0.112	5.892	0.664	0.079	8.405
pci_bridge32_a	30k	30k	B	0.301	0.102	2.950	0.356	0.095	3.747	0.675	0.115	5.869	0.530	0.092	5.760
pci_bridge32_b	29k	29k	A	0.425	0.093	4.569	0.471	0.102	4.617	0.730	0.101	7.227	0.734	0.077	9.532
superblue11_a	954k	936k	B	0.445	0.074	6.013	0.521	0.070	7.442	0.675	0.115	5.869	0.740	0.066	11.21
superblue12	1.3m	1.3m	B	0.323	0.111	2.909	0.392	0.096	4.083	0.638	0.093	6.860	0.641	0.084	7.630
superblue14	634k	620k	A	0.476	0.083	5.734	0.498	0.099	5.030	0.793	0.083	9.554	0.783	0.063	12.42
superblue16_a	698k	697k	A	0.385	0.095	4.052	0.458	0.084	5.452	0.653	0.108	6.046	0.661	0.068	9.720
superblue19	522k	512k	A	0.454	0.116	3.913	0.488	0.105	4.647	0.800	0.078	10.25	0.783	0.064	12.23
Average	-	-	-	0.411	0.090	4.566	0.468	0.091	5.142	0.713	0.099	7.202	**0.717**	**0.072**	**9.958**
Ratio	-	-	-	0.57	1.25	0.46	0.65	1.26	0.52	0.99	1.38	0.72	**1.00**	**1.00**	**1.00**

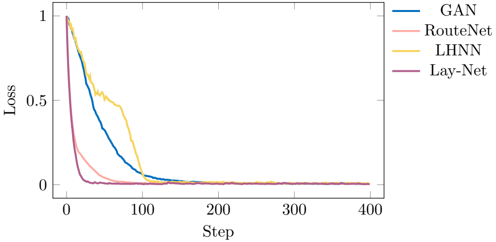
Ablation Study

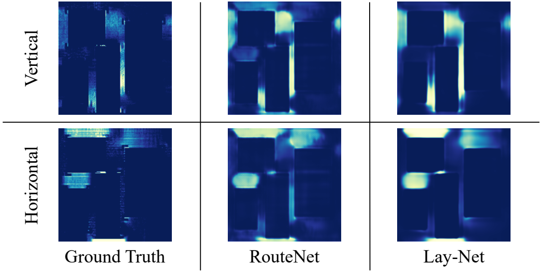
- ViT: vanilla vision transformer backbone
- Swin: Swin transformer backbone
- Swin+HGNN: heterogeneous fusion, without MacroMargin
- Swin+HGNN+MM: heterogeneous fusion, with MacroMargin



Training Curve & Examples

- Lay-Net converges faster than others
- Lay-Net does not misclassify like the previous method





Conclusion

- Lay-Net enables the multi-modal fusion of layout and netlist information
- Lay-Net achieves up to 38.9% improvement over existing methods
- The proposed MacroMargin feature is effective
- Layout-netlist information fusion works!

THANK YOU!