International Symposium on Physical Design

GraphCAD: Leveraging Graph Neural Networks for Accuracy Prediction Handling Crosstalk-affected Delays

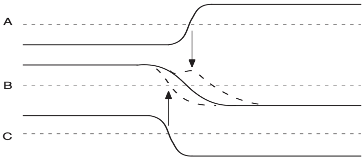
Fangzhou Liu[1], Guannan Guo[2], Yuyang Ye[1], Ziyi Wang[1], Wenjie Fu[3], Weihua Sheng[2], Bei Yu[1]

[1]The Chinese University of Hong Kong, Hong Kong SAR
[2]Huawei Design Automation Lab, Hong Kong SAR
[3]HiSilicon Technologies Co., Shanghai

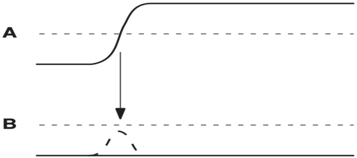
Outline

Crosstalk challenges:

- Scaling: Wire length-to-width adjustments → increased coupling capacitance.



(a) Transition slowdown or speedup



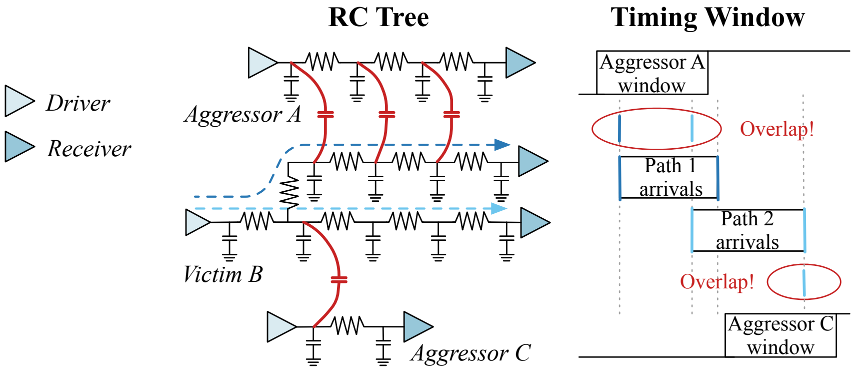
(b) Noise bump

Existing Methods:

- **Traditional simulations**: Challenges in multi-input transfer functions & holding resistance; Issues with logic correlation; Computationally intensive
- **Previous learning-based works**: Focus on only RC paratistics;[1] Lack of coupling features or timing features;[2] Limited methodology.

[1] Andrew B Kahng, Mulong Luo, and Siddhartha Nath (2015). "SI for free: machine learning of interconnect coupling delay and transition effects". In: *Proc. SLIP*, pp. 1–8.

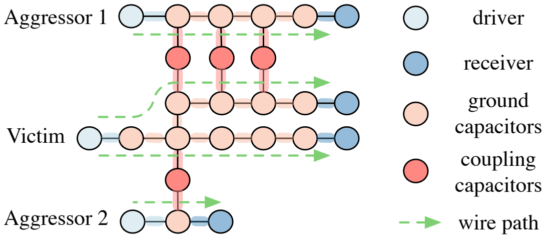
[2] Yuyang Ye et al. (2023). "Fast and accurate wire timing estimation based on graph learning". In: *Proc. DATE*. IEEE, pp. 1–6.



Primary causes of crosstalk noise:

1. Coupling effect;
2. Timing window overlapping between nets.

Algorithm



Description of node and path features.

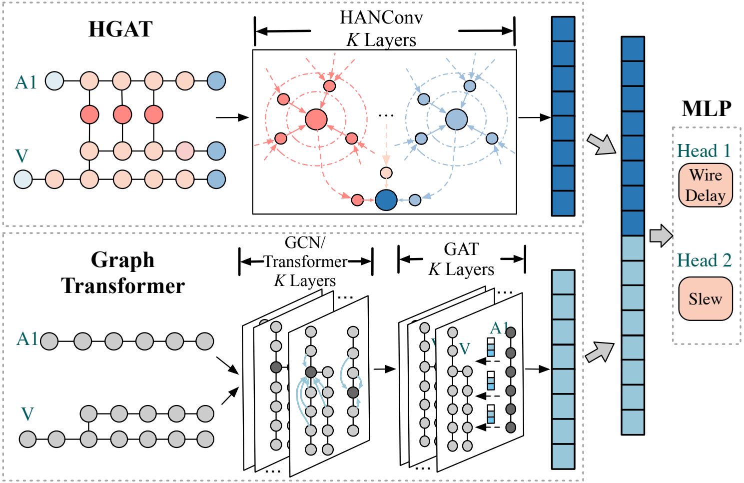
Feature	Description
f_{n1}	Capacitance values
f_{n2}	Number of input nodes
f_{n3}	Number of output nodes
f_{n4}	Total input capacitance
f_{n5}	Total output capacitance
f_{n6}	Number of connected resistors
f_{n7}	Total input resistance
f_{n8}	Total output resistance
f_{n9}	Ratio of coupling-to-total capacitance
f_{n10}	Indicates if it is a victim net
f_{n11}	List of corresponding aggressors
f_{p1}	Incremental delay for each wire path
f_{p2}	Minimum transition time for driver/receiver
f_{p3}	Maximum transition time for driver/receiver
f_{p4}	Minimum arrival time for driver/receiver
f_{p5}	Maximum arrival time for driver/receiver

Graph construction:

- **Nodes**: drivers, receivers, and capacitances; **Edges**: resistances.
- **HGAT input**: a heterogeneous graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$;
$\boldsymbol{X}_{\text{dict}} : \{\text{"coup"} : \boldsymbol{x}_{\text{coup}}, \text{"cap"} : \boldsymbol{x}_{\text{cap}} \ldots\}$; $\boldsymbol{E}_{\text{dict}}:\{(\text{"cap"}, \text{"coup"}) : \boldsymbol{e}_{\text{cap, coup}} \ldots\}$.
- **Graph transformer input**: $\mathcal{G} = (\mathcal{E}, \mathcal{V}, \mathcal{P})$, where $\mathcal{P}$ denotes wire paths; a node feature matrix $\boldsymbol{X}$, a path feature matrix $\boldsymbol{P}$ and a weighted adjacency matrix $\boldsymbol{A} = [a_{i,j}]$.

Predition objective:

- $t_{\text{wire}} = g(\boldsymbol{f}_n, \boldsymbol{f}_p; \theta_g)$; $\quad t_{\text{trans}} = h(\boldsymbol{f}_n, \boldsymbol{f}_p; \theta_h)$



Overview of GraphCAD.

HGAT Model: Coupling Effect Analysis

Intra-relation information encoding:

- Learn the weight among neighboring nodes of the same type:

$$e_{u,v}^{\Theta} = \sigma(a_{\Theta}^{\top} \cdot [h_i || h_j]). \tag{1}$$

- Normalize:

$$\alpha_{u,v}^{\Theta} = \mathrm{softmax}(e_{u,v}^{\Theta}) = \frac{\exp(e_{u,v}^{\Theta})}{\sum_{k \in \mathcal{N}^{\Theta}(u)} \exp(e_{u,k})}. \tag{2}$$

- The relation-based embedding of node u:

$$z_u^{\Theta} = \sigma(\sum_{v \in \mathcal{N}^{\Theta}(u)} \alpha_{u,v}^{\Theta} h_v). \tag{3}$$

HGAT Model: Coupling Effect Analysis

Aggregation of relation-level information:

- Average the importance of all the relation-level node embeddings:

$$e_{\Theta_i} = \frac{1}{|\mathcal{V}_{\Theta_i}|} \sum_{u \in \mathcal{V}_{\Theta_i}} \boldsymbol{q}^\top \cdot tanh(W \times z_u^{\Theta_i} + b). \tag{4}$$

- Normalize:

$$\alpha_{\Theta_i} = \mathrm{softmax}(e_{\Theta_i}) = \frac{\exp(e_{\Theta_i})}{\sum_{j=1}^{K} \exp(e_{\Theta_j})}. \tag{5}$$

- Fuse the relation-level node embeddings to generate the final embeddings:

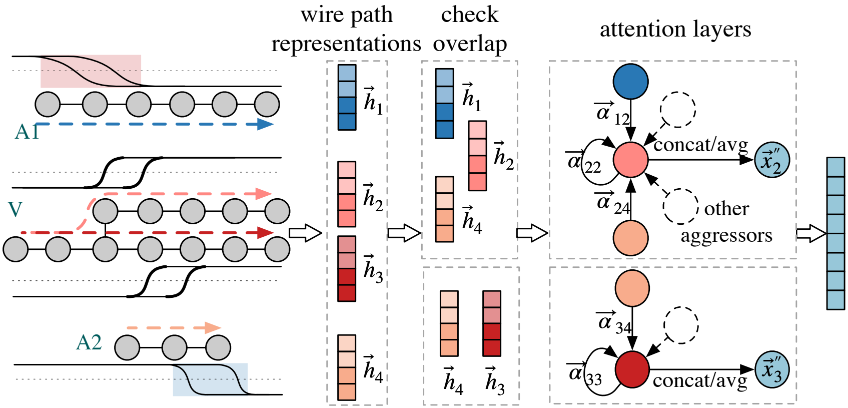
$$Z = \sum_{i=1}^{K} \alpha_{\Theta_i} \times Z_{\Theta_i}. \tag{6}$$

- Global pooling and output:

$$\begin{aligned} \boldsymbol{y}_{\mathrm{HGAT}} &= GlobalPool(Z) \\ &= [(\frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} z_u)||(\frac{1}{|\mathcal{V}^{coup}|} \sum_{v \in \mathcal{V}^{coup}} f(x_v))]. \end{aligned} \tag{7}$$

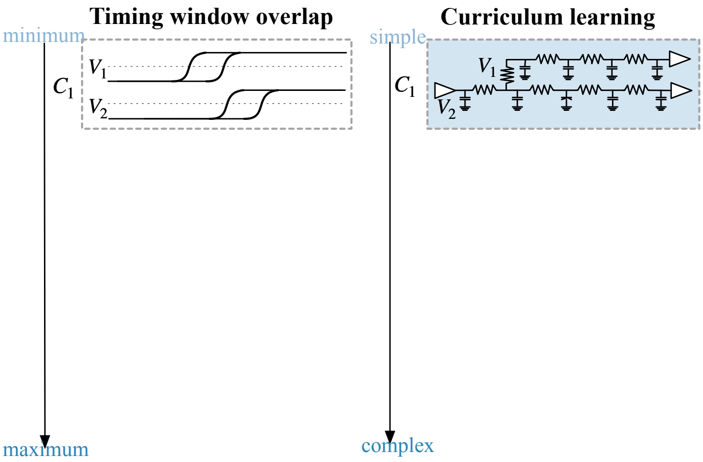
Graph transformer model: Overlapping Net Analysis

- GraphSAGE: $\boldsymbol{x}_v^{(l+1)} = \mathrm{ReLU}\left(\mathrm{Norm}\left(\boldsymbol{W}^{(l)} \cdot \mathrm{MEAN}\left(\{\boldsymbol{x}_v^{(l)}\} \cup \{\boldsymbol{x}_u^{(l)} : u \in \mathcal{N}(v)\}\right)\right)\right)$
- Transformer: $\boldsymbol{x}'_v = \mathrm{TransformerEncoder}(\boldsymbol{x}_v^{(L_1)}, L_2)$
- GAT layers: $\boldsymbol{x}''_v = \sum_{u \in \mathcal{N}_{\mathcal{C}}(v)} \alpha_{vu} \boldsymbol{W} \boldsymbol{x}_u$



Graph Transformer model incorporating overlapping net information.

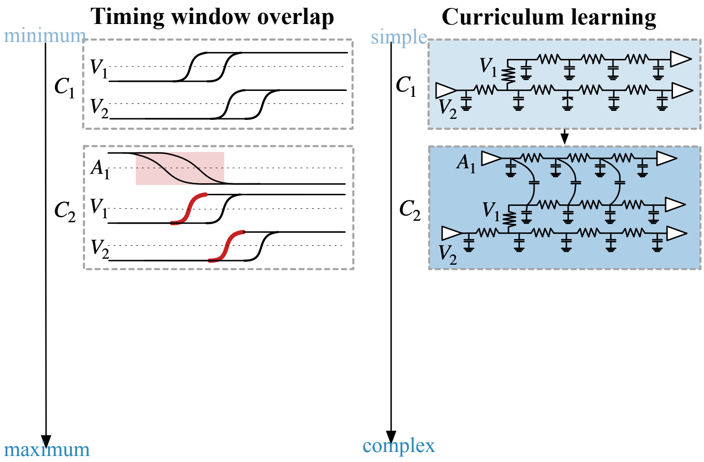
Curriculum Learning Mechanism



- Customized loss function:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} C \cdot \mathcal{L}(y_i, f(x_i; \theta)), \tag{8}$$

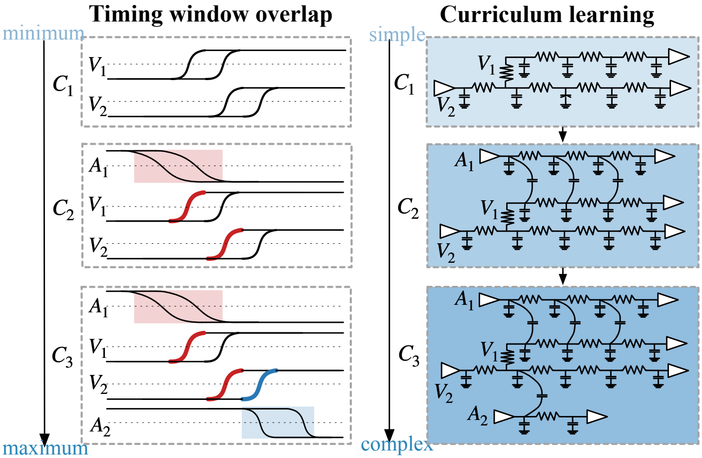
Curriculum Learning Mechanism



- Customized loss function:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} C \cdot \mathcal{L}(y_i, f(x_i;\theta)), \tag{8}$$

Curriculum Learning Mechanism



- Customized loss function:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} C \cdot \mathcal{L}(y_i, f(x_i; \theta)), \tag{8}$$

Results

Experimental setting

Dataset preparation:

- Technology library: Open-source ASAP7,
- Synthesis & PnR: Design Compiler $\rightarrow$ Innovus,
- Feature extraction: PrimeTime non-SI mode,
- Ground truth: HSPICE.

Configurations:

- GraphCAD: pyg and pytorch, spef-parser(cpp),
- dynamic adjusted learning rate from 0.01 to 0.006,
- batch size: 128, 150 epochs,
- training: 20 hours on a single GPU.

Overall Performance

Table: Comparison of estimation errors against HSPICE results.

Benchmarks	RC-VA	Predicted Wire Delay				Predicted Slew at Receiver			
		PrimeTime	NetTiming[3]	MLP[4]	GraphCAD	PrimeTime	NetTiming	MLP	GraphCAD
wdsp	19	**14.89%**	31.32%	37.37%	25.66%	47.36%	30.04%	80.80%	**26.03%**
ae18	11	**13.27%**	23.39%	24.34%	16.42%	35.95%	35.71%	72.96%	**34.82%**
wb2axip	22	14.45%	24.37%	22.14%	**12.51%**	17.26%	22.54%	33.75%	**10.17%**
usb_device	99	**14.91%**	22.38%	31.21%	21.18%	30.46%	28.37%	53.25%	**27.93%**
fpu	183	**10.19%**	23.00%	29.05%	20.25%	21.82%	34.64%	39.79%	**20.84%**
LSU	31	**7.93%**	22.83%	41.00%	18.31%	39.30%	40.03%	80.16%	**36.81%**
vga_lcd	24	**15.82%**	32.18%	65.24%	19.59%	43.70%	24.33%	97.15%	**20.45%**
SmallQuadBoom	511	**6.62%**	26.43%	27.71%	24.07%	**16.72%**	36.18%	29.63%	22.28%
SmallBoom	402	**8.44%**	25.64%	34.83%	21.92%	**20.26%**	35.08%	29.75%	23.52%
BoomCore	326	**11.41%**	31.92%	33.50%	24.91%	**16.84%**	45.77%	25.57%	24.05%
or1200	2	12.36%	14.15%	76.56%	**9.42%**	**15.86%**	47.58%	42.14%	21.79%
sparc	174	**10.30%**	26.30%	29.00%	24.93%	**19.47%**	37.44%	28.72%	23.48%
Average	-	**11.72%**	25.33%	37.66%	19.93%	27.08%	34.81%	51.14%	**24.35%**
Delta	-	**-8.21%**	5.40%	17.73%	0	2.74%	10.46%	26.79%	**0**

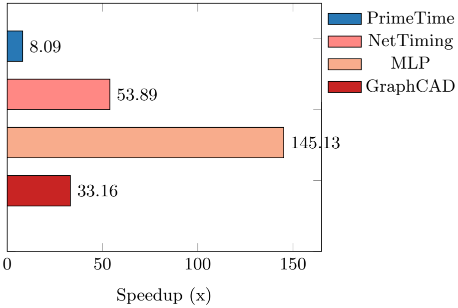
[3] Yuyang Ye et al. (2023). "Fast and accurate wire timing estimation based on graph learning". In: *Proc. DATE*. IEEE, pp. 1–6.

[4] Leilei Jin et al. (2024). "A Crosstalk-Aware Timing Prediction Method in Routing". In: *arXiv preprint arXiv:2403.04145*.

Runtime Comparison

Benchmarks	Runtime (s)				
	HSPICE	PrimeTime	NetTiming [7]	MLP	GraphCAD
wdsp	34.222	12.363	3.004	2.641	2.126
ae18	18.687	11.780	3.252	2.945	1.559
wb2axip	38.667	15.045	4.297	1.180	2.188
usb_device	169.281	16.026	4.838	1.225	5.905
fpu	321.501	21.878	5.862	1.199	9.246
LSU	55.532	24.155	5.296	1.117	2.371
vga_lcd	46.886	16.284	5.896	1.131	2.222
SmallQuadBoom	903.420	43.565	7.397	2.886	25.861
SmallBoom	709.360	44.576	7.516	1.242	20.769
BoomCore	590.145	67.807	7.204	2.812	20.486
or1200	3.649	37.710	2.639	2.554	0.906
sparc	535.247	112.374	6.390	2.687	9.716
Average	285.550	35.297	5.299	1.968	8.613
Ratio	33.154	4.098	0.615	0.229	1.000

(a) Runtime comparison.



(b) Illustration of runtime speedup.

Conclusion

- We propose GraphCAD, an end-to-end GNN framework to predict crosstalk-affected delays by jointly modeling coupling effects and overlapping nets.
- We combine heterogeneous graph learning and transformers to map aggressor-victim interactions and analyze their overlapping timing windows.
- A curriculum learning strategy is implemented to handle complex multi-aggressor scenarios progressively.
- Experimental studies validate the framework through tests on 7nm technology open-source designs, demonstrating improved accuracy and efficiency.