WHERE **INNOVATION** BEGINS

DESIGN AUTOMATION CONFERENCE

JULY 9–13, 2023

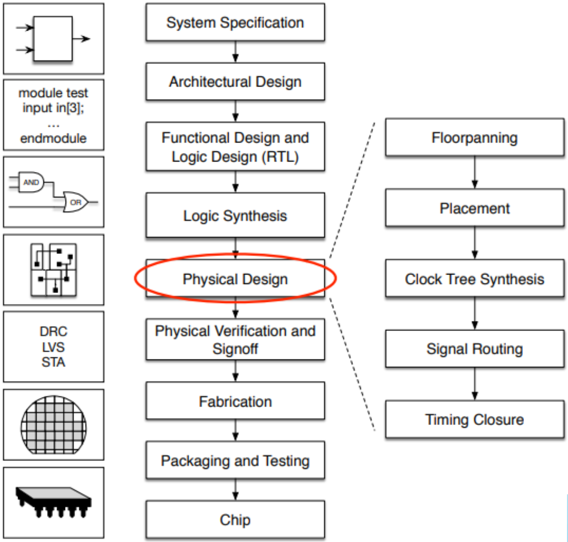
MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Restructure-Tolerant Timing Prediction via Multimodal Fusion

Ziyi Wang[1], Siting Liu[1], Yuan Pu[1], Song Chen[2], Tsung-Yi Ho[1], Bei Yu[1]

[1]Chinese University of Hong Kong
[2]University of Science and Technology of China

Background: Pre-routing Timing Prediction



- repetitive *P&R* to guarantee timing closure is **costly**
- Timing evaluation in early stages is necessary
- raise the demand for **pre-routing timing prediction**

Previous Methods

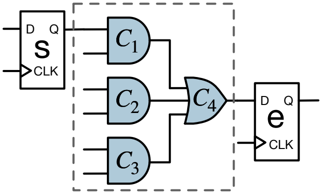
- **Traditional** method, e.g., Elmore's model [Rub+83], is imprecise due to inaccurate wire estimation without actual routing information.
- **ML-driven** timing prediction works can be divided into 2 classes:
  1. **two-stage** [Bar+19] [He+22]: first predict local net/cell delays and then apply graph traversals to evaluate global timing metric.
  2. **end-to-end** [Guo+22]: directly predict global timing metrics, but still relies on local net/cell delay prediction as auxiliary tasks.

Drawback of Previous ML-assisted Works

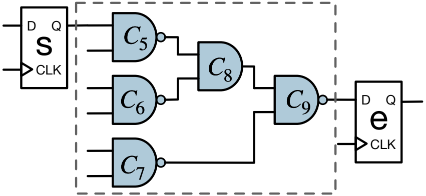
- follow a **local-view** fashion: only focus on **local** graph information

  can not deal with real-world scenarios where timing optimization is taken into account!

Timing optimization: Destructed topology



(a) Input circuit



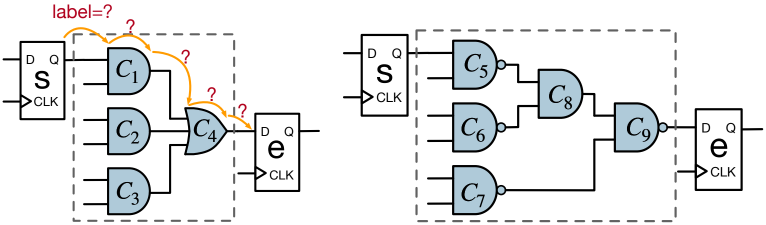
(b) Optimized circuit

Example of circuit reconstruction after timing optimization.

graph topology is **destructed** after timing optimization!

- timing prediction based on only graph information is unreliable!

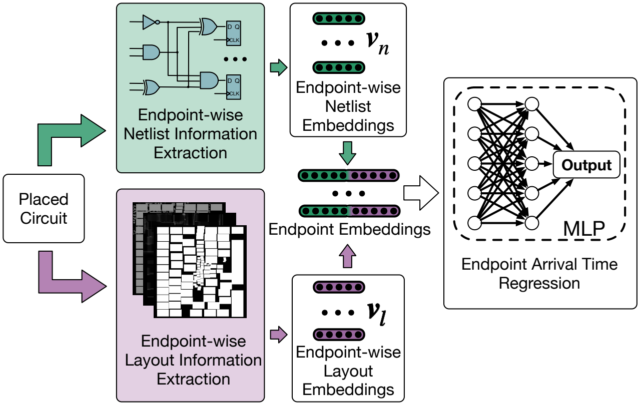
Impact of Topology Restructuring



(a) Input netlist (b) Optimized netlist

1. prohibits labeling the net/cell delays inside the box.
   - previous local-view method can only work in **semi-supervised** way
2. leads to a **mismatch** between input features and ground-truth features.
   - **inconsistency** between local delay supervision and global timing metrics prediction.

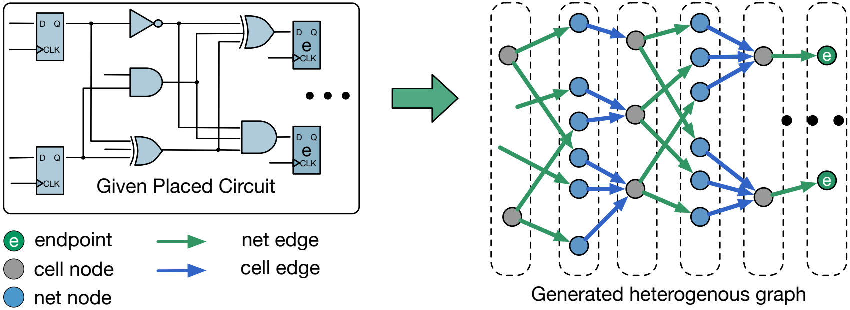
Overview



- Global endpoint-wise views from both netlist and layout
- Customized GNN model to extract endpoint-wise netlist information.
- CNN model with masking to extract endpoint-wise layout information.

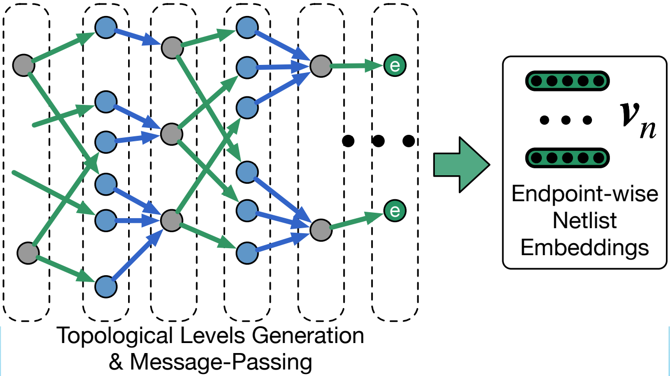
Netlist Embedding: Data Representation

- each pin as a node
- **heterogeneous** graph with two edge types: cell edge and net edge
- transformed to **DAG** by removing cell edges of registers

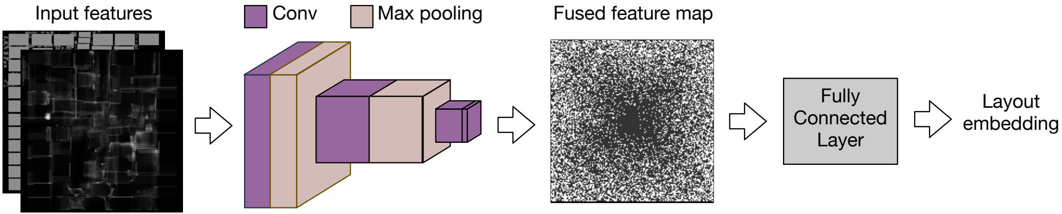


Netlist Embedding: Message Passing Scheme

- Motivated by delay propagation
- flows in the **topological order** and aggregated at endpoints
- **different** aggregators $\mathcal{A}_c$ and $\mathcal{A}_n$ for cell nodes and net nodes, respectively.
- use **maximum** operator to gather predecessor messages for cell nodes



Layout Embedding: Naive Flow



Problem: identical layout embedding for all endpoints.

- does not make sense: timing optimization's impact **varies greatly** for different endpoints.

We should extract **unique** layout information for each endpoint!

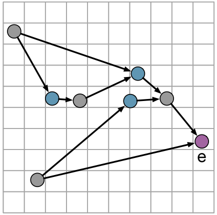
Endpoint-wise masking

- We propose a critical **region-based** method to extract unique **endpoint-wise** layout information.
- We derive the critical region from a critical path.

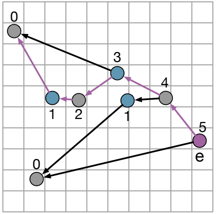
Critical path for e is defined as the longest path from PIs to e.
Arrival time at e is closely related to the critical path.

Critical path finding

- Reverse the graph and conduct a DFS starting from each endpoint e
- Always move to the successor with topological level -1 in the next step.
- Stopped when reaching PIs.



(c) Input graph



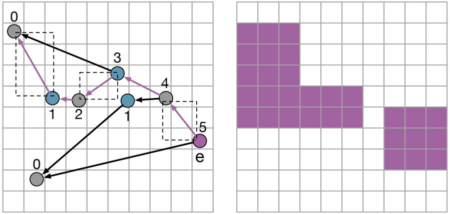
(d) Path-finding

Purple, blue, and gray represent the endpoint, net nodes and cell nodes, respectively. The number next to each node in (b) indicates its topological level, and the purple lines depict the longest path P_e for e.

Mask Generation

- The critical region $\mathbf{R}_e$ for an endpoint e is constructed by taking the union region covered by the bounding boxes of the two-pin net edges along the critical path P_e:

$$\mathbf{R}_e = \bigcup_{\{d,s\} \in E^n(P_e)} \mathbf{B}_{d,s}, \tag{1}$$

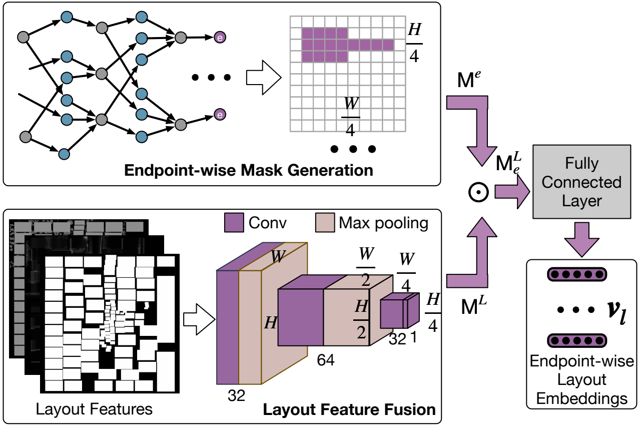


(a) Found critical paths

(b) Output mask

The dotted boxes illustrate the critical region $\mathbf{R}_e$, which consists of net edge bounding boxes along P_e. Only the regions covered by net edges are considered.

Layout Embedding Generation Flow



Our endpoint-wise layout embedding generation flow with a CNN model and a novel endpoint-wise masking technique.

Experimental Setting: Dataset Preparation

Table: Statistics of the dataset. edp stands for endpoint, e_n and e_c denote net edge and cell edge, respectively.

Benchmark		#pin	#edp	$\#e_n$	$\#e_c$
train	jpeg	932842	40801	650878	607795
	rocket	698347	52731	490499	432068
	smallboom	694441	61764	488052	423344
	steelcore	26598	1662	19439	17732
	xgate	20842	684	14653	13010
test	arm9	44469	2500	33065	29287
	chacha	35687	1986	25117	23083
	hwacha	1357798	61313	985057	922085
	or1200	1165114	172401	844443	658961
	sha3	794720	60323	552021	485596
Avg	train	474614	31528	332704	298790
	test	679558	59705	487941	423802

- 10 open-source designs from chipyard and Github.
- Cadence Genus advanced 7-nm ASAP7 PDK [Cla+16] for synthesis, and Cadence Innovus for placement, timing optimization, and routing.

Comparison with SOTA timing prediction works

Benchmark	baselines' net/cell delay prediction (R^2 score)			Endpoint arrival time prediction (R^2 score)					
	DAC19 [Bar+19]	DAC22-he [He+22]	DAC22-guo [Guo+22]	DAC19	DAC22-he	DAC22-guo	our CNN-only	our GNN-only	our full
arm9	0.0101	-0.5187	-0.2960 / -1.8234	0.6655	0.7304	0.8279	-0.0011	0.8405	**0.8852**
chacha	-0.1389	-0.1008	-0.0813 / -0.2737	0.4406	0.6146	-0.0253	-0.1152	0.7346	**0.9027**
hwacha	0.0519	-0.0323	-0.8003 / -0.8630	0.2752	0.5186	0.7090	-0.0173	0.8022	**0.8623**
or1200	-0.0395	-0.3051	-3.5679 / -0.0924	0.3226	0.4484	0.6776	-0.0019	0.7381	**0.8081**
sha3	0.3941	0.5554	-0.3713 / 0.1230	0.7784	0.7917	0.8464	-0.0058	0.8635	**0.9035**
avg	0.0555	-0.0803	-1.0234 / -0.5859	0.4965	0.6207	0.6071	-0.0283	0.7958	**0.8724**

- Our framework vastly **outperforms** all the baseline approaches
- Hard to model timing optimization's impact locally with pre-routing information.
- Prediction on local delay is inconsistent with that on global timing metrics.
- **Layout alone is useless** but works well when combined with netlist.

Runtime Analysis

Table: Runtime (s) comparison with an industry-leading commercial tool.

design	commercial (20 threads)				ours			
	opt	route	sta	total	pre	infer	total	speedup
jpeg	7863	624922	227	633012	20.63	5.56	26.19	**24170**×
rocket	16239	19161	167	35567	18.53	2.02	20.55	**1731**×
smallboom	9051	53942	152	63145	19.72	4.81	24.53	**2574**×
steelcore	1294	747	20	2061	0.39	1.12	1.51	**1365**×
xgate	338	630	17	985	0.34	0.48	0.82	**1201**×
arm9	305	1825	16	2146	0.88	1.78	2.66	**807**×
chacha	1621	1794	23	3438	0.82	1.20	2.02	**1702**×
hwacha	43883	136946	241	181070	23.89	5.77	29.66	**6105**×
or1200	28641	40291	339	69271	112.20	6.52	118.72	**583**×
sha3	18785	16870	185	35840	24.95	2.58	27.53	**1302**×
avg.	12802	89713	139	102654	22.23	3.184	25.42	**4154**×

Summary

- Fast and accurate pre-routing timing prediction is critical in reducing design cycles.
- Previous ML-assisted works following a local-view fashion did not consider the impact of timing optimization, leading to performance degradation in real-world applications.
- A novel endpoint embedding framework is presented with multimodal fusion by utilizing both GNN and CNN to extract netlist and layout information.
- We should keep a close eye on multimodal fusion in the VLSI design flow for more thorough information mining.

Reference I

[1] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in rc tree networks", *tcad*, pp. 202–211, 1983.

[2] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism",, 2019.

[3] X. He, Z. Fu, Y. Wang, C. Chang Liu, and Y. Guo, "Accurate timing prediction at placement stage with look-ahead rc network",, 2022.

[4] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction",, 2022.

[5] L. T. Clark *et al.*, "Asap7: A 7-nm finfet predictive process design kit", *Microelectronics Journal*, 2016.

THANK YOU!