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

Introduction

- Repetitive placement and routing to guarantee timing closure is **costly**.
- Pre-routing Timing evaluation is necessary to save abundant routing running time and provide quick feedback to optimize timing early.

Background

Timing Closure

- Slack: $s_e = r_e - a_e$ for a timing path endpoint e, where r_e and a_e denote e's required time and arrival time.
- Worst Negative Slack (WNS): $w(\cdot) = \min_e s_e$.
- Total Negative Slack (TNS): $t(\cdot) = \sum_e \{\min\{0, s_e\}\}$

Timing awareness has been extended to most phases of the physical design flow for the timing closure.

Related works

- **Traditional** method, e.g., Elmore's model [1], is imprecise due to inaccurate wire estimation without actual routing information.
- **2** classes of ML-driven methods:
  1. two-stage [2] [3]: first predict local net/cell delays and then apply graph traversals to evaluate the global timing metric.
  2. end-to-end [4]: directly predict global timing metrics, but still relies on local net/cell delay prediction as auxiliary tasks.

Highlights

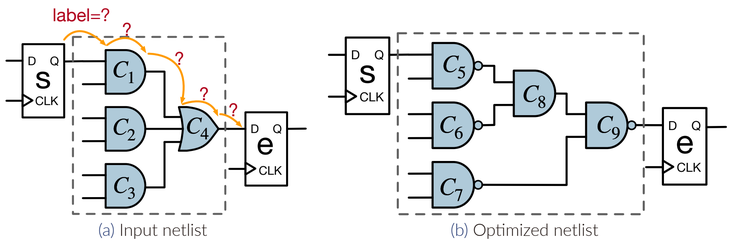


Figure 1. Example of circuit reconstruction after timing optimization

- Previous methods follow a local-view fashion that only focuses on **local** graph information, which is destructed after timing optimization (TO).
- Graph Restructuring leads to inconsistency between local delay supervision and global timing metrics prediction.

Restructure-tolerant Pre-routing Timing Prediction Given the pre-routing layout and netlist of a design, our goal is to make an accurate and efficient estimation of the sign-off global timing metrics, i.e., endpoint arrival time, with the impact of timing optimization taken into account.

Overall Flow

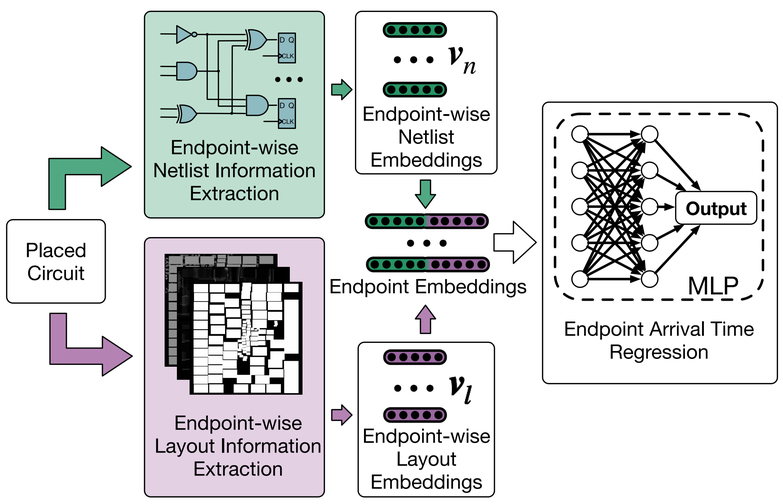


Figure 2. Overview of our pre-routing timing prediction framework, which first generates endpoint-wise embedding and then conducts sign-off global timing prediction.

Highlights

- We develop a novel endpoint embedding framework that fuses layout-netlist information considering the impact of timing optimization.
- A customized graph neural network is presented to extract and aggregate endpoint-wise netlist information.
- A convolutional neural network with an endpoint-wise masking technique is developed to efficiently extract the unique layout information for each timing endpoint.

Netlist Embedding Model

Data Representation

- Heterogeneous graph taking each pin as a node, and with two edge types: cell edge and net edge.
- Directed acyclic graph (DAG) by removing cell edges of registers.

Customized Message Passing Scheme

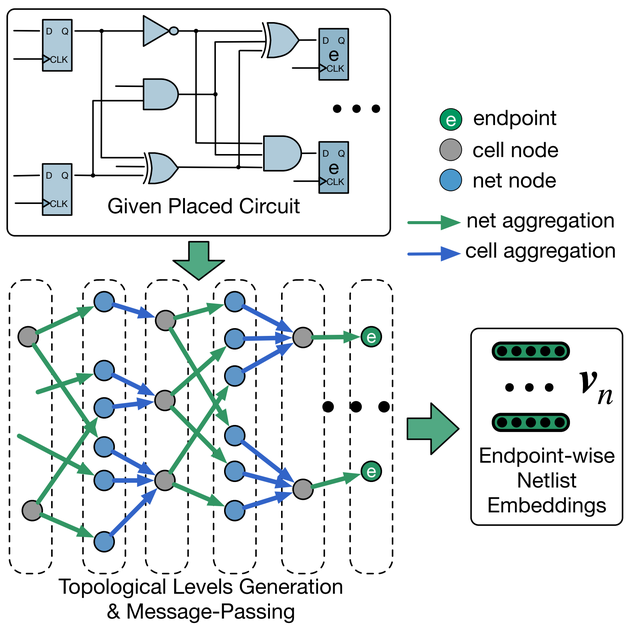


Figure 3. Our netlist embedding flow.

$$\vec{h}_v = \begin{cases} \sigma(\, f_{c1}^{\text{MLP}}(\max\{\vec{h}_u : u \in \mathcal{N}(v)\}) + f_{c2}^{\text{MLP}}(\vec{h}_v^c)\,) & v \in V_c \\ \sigma(\, \vec{h}_d + f_n^{\text{MLP}}(\vec{h}_v^n)\,) & v \in V_n \end{cases} \quad (1)$$

Highlights

- Motivated by delay propagation
- Flows in the topological order and aggregated at endpoints
- Different aggregators for cell nodes and net nodes, respectively.

Layout Embedding Model

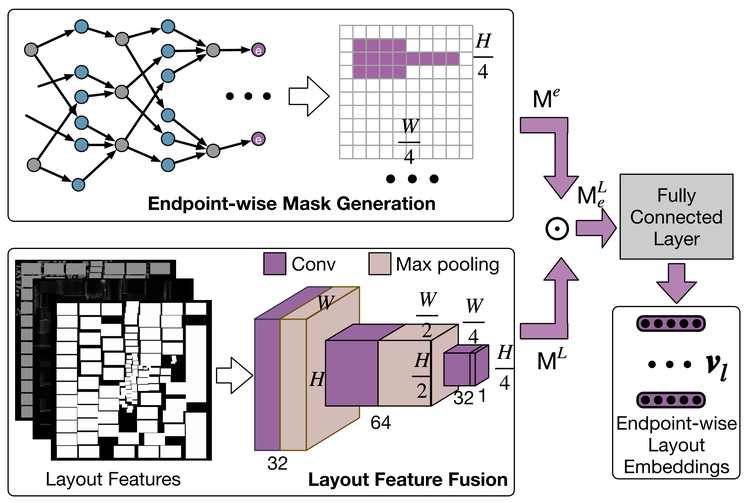
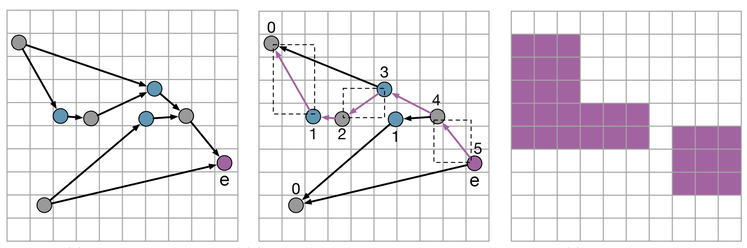


Figure 4. Our endpoint-wise layout embedding generation flow with a CNN model and a novel endpoint-wise masking technique.

Highlights

- We propose a critical region-based method to extract unique endpoint-wise layout information.
- We derive the critical region from a critical path, based on the observation that the impact of TO on e is closely related to this path.

Critical Mask Generation



(a) Input graph (b) Path-finding and mask generation (c) Output mask

Figure 5. Example of mask generation. The number next to each node indicates its topological level, and the purple lines depict the longest path P_e for e. The dotted boxes illustrate the critical region consisting of net edge bounding boxes along P_e.

Results

Benchmark		Input information				Impact of timing optimization on sign-off metrics					
						slack variation		net variation		cell variation	
		#pin	#edp	$\#e_n$	$\#e_c$	Δwns	Δtns	#replaced	Δdelay	#replaced	Δdelay
train	jpeg	932842	40801	650878	607795	98.8%	99.8%	32.5%	50.8%	35.4%	40.6%
	rocket	698347	52731	490499	432068	94.0%	94.8%	28.5%	70.0%	8.0%	24.2%
	smallboom	694441	61764	488052	423344	87.4%	99.1%	40.9%	53.1%	15.6%	39.8%
	steelcore	26598	1662	19439	17732	89.6%	98.3%	49.8%	51.8%	18.4%	29.4%
	xgate	20842	684	14653	13010	94.8%	99.1%	31.3%	50.7%	16.9%	20.9%
test	arm9	44469	2500	33065	29287	82.5%	88.8%	46.7%	47.8%	24.0%	42.9%
	chacha	35687	1986	25117	23083	86.5%	88.0%	47.1%	62.5%	38.8%	40.2%
	hwacha	1357798	61313	985057	922085	91.7%	92.5%	45.1%	70.4%	22.0%	37.1%
	or1200	1165114	172401	844443	658961	95.4%	97.3%	49.1%	61.4%	20.8%	28.6%
	sha3	794720	60323	552021	485596	96.0%	97.4%	30.3%	77.6%	8.3%	28.8%
Avg	train	474614	31528	332704	298790	92.9%	98.2%	36.6%	55.3%	18.9%	31.0%
	test	679558	59705	487941	423802	90.4%	92.8%	43.7%	63.9%	22.8%	35.5%

Table 1. Dataset statistic. We use Cadence Genus with advanced 7-nm ASAP7 PDK [5] for synthesis, and Cadence Innovus for placement, timing optimization, and routing.

Benchmark	baselines' net/cell delay prediction (R^2 score)			Endpoint arrival time prediction (R^2 score)					
	DAC19 [?]	DAC22-he [?]	DAC22-guo [?]	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**

Table 2. Overall Comparison on the test benchmarks.

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**×

Table 3. Runtime (s) Analysis where pre is for preprocessing consisting of graph construction, topological level and critical mask generation.

Highlights

- Our proposed global-view fashion outperforms prior local-view fashion when timing optimization is taken into account.
- Previous methods perform badly on local net/cell delay prediction, demonstrating that it is hard to model the influence of TO locally with only pre-routing information.
- Prediction performance on local net/cell delay is inconsistent with that on global timing metrics.
- Layout information alone is useless but works well when combined with netlist information.
- Our proposed framework is highly efficient for obtaining an accurate estimation of sigh-off timing performance.

Conclusion & Future Directions

- This study has raised the importance of timing-optimization-aware pre-routing timing prediction and provides a novel framework for timing endpoint embedding generation.
- We should keep a close eye on multimodal fusion in the VLSI design flow for more thorough information mining.
- This work can be extended by feedbacking the prediction to guide timing optimization in the placement stage.

References

[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," in *Proc. DAC*, 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," in *Proc. DAC*, 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," in *Proc. DAC*, 2022.

[5] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, 2016.

DESIGN AUTOMATION CONFERENCE 60
FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW