WHERE **INNOVATION** BEGINS

DESIGN AUTOMATION CONFERENCE

JULY 9–13, 2023

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

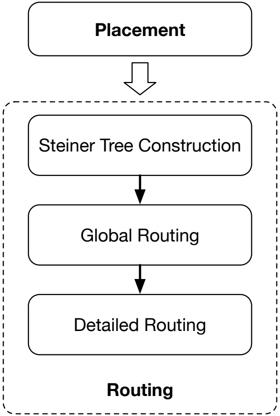
Concurrent Sign-off Timing Optimization via Deep Steiner Points Refinement

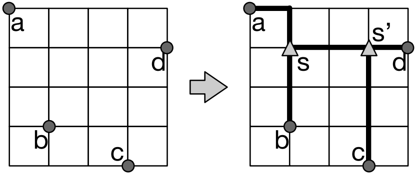
Siting Liu[1,2†], Ziyi Wang[1†], Fangzhou Liu[1],
Yibo Lin[2], Bei Yu[1], Martin Wong[1]

[1]Chinese University of Hong Kong
[2]Peking University

Outline

Physical Design





Steiner tree construction

- Modern routing stage includes three stages.
- Steiner tree decomposes the multi-pin net into a set of two-pin net.

Early Stage Timing Optimization

- Placement [1] [2] : Pre-routing timing metrics.
- Routing [3] [4] [5] [6] : Path lengths, early timing metrics.

[1] P. Liao, and et al., “DREAMPlace 4.0: timing-driven global placement with momentum-based net weighting," DATE 2022

[2] Z. Guo and Y. Lin, “Differentiable-Timing-Driven Global Placement,” DAC 2022

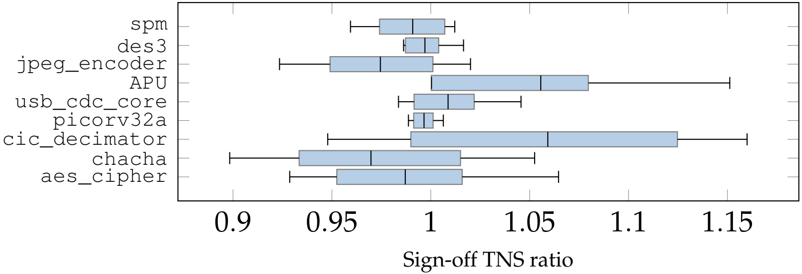
[3] C. J. Alpert, and et al., “Timing-driven Steiner trees are (practically) free," DAC 2006

[4] C. J. Alpert, and et al., "Prim-Dijkstra revisited: Achieving superior timing-driven routing trees," ISPD 2018

[5] S. Held, and et al., "Global routing with timing constraints," TCAD 2017

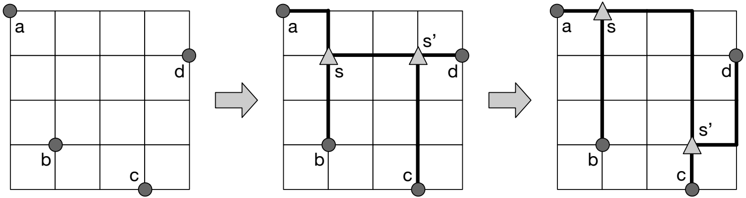
[6] D. Wu, and et al., "Timing driven track routing considering coupling capacitance," ASPDAC 2005

Random Steiner Point Disturbance



- The sign-off timing performance could be significantly affected even by a random disturbance on Steiner point position.
- The impact of random moving is considerately unstable, and its average performance is slight.

Steiner Point Refinement



Steiner point refinement

Timing-driven Steiner Point Refinement

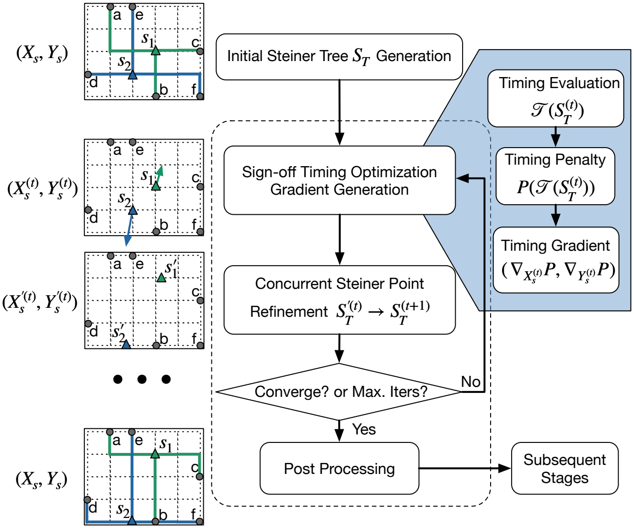
Given an initial Steiner tree set $S_T = \{T^1, T^2, \cdots, T^n\}$, $T^i = (V_c^i, V_s^i, E^i)$, where V_c^i is the set of cell nodes, V_s^i is the set of Steiner nodes and E^i means the edges connecting V_c^i and V_s^i of the i^{th} Steiner tree, our task is to refine the position (X_s, Y_s) of $V_s = \{V_s^i, 1 \leq i \leq n\}$ in the pre-routing stage to obtain better **sign-off** timing performance.

Machine Learning Opportunities [7]

ML enables fast and accurate sign-off timing evaluation.

[7] Z. Guo, and et al., "A timing engine inspired graph neural network model for pre-routing slack prediction," DAC 2022

Overall flow - TSteiner

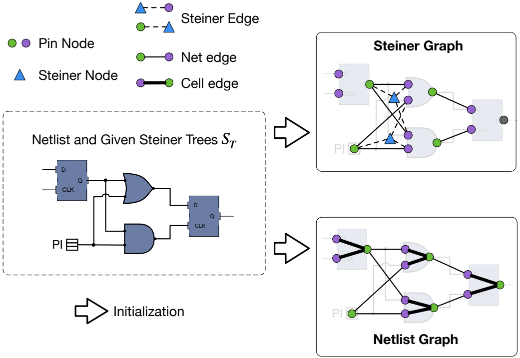


Sign-off Timing Optimization Gradients

Sign-off Timing Evaluation

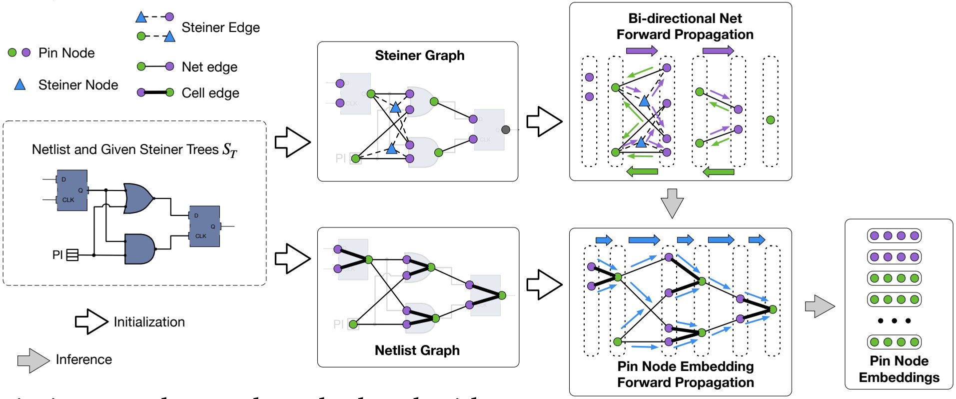
Given a Steiner tree solution S_T, timing evaluation is to find an estimator $\mathcal{T}$ to evaluate the sign-off timing metrics $\mathcal{T}(S_T)$, i.e., arrival time at each pin.

Sign-off Timing Optimization Gradients - Initialization



- **Steiner Graph**: pin node and Steiner node; net edge and Steiner edge.
- **Netlist Graph**: pin node; cell edge and net edge.

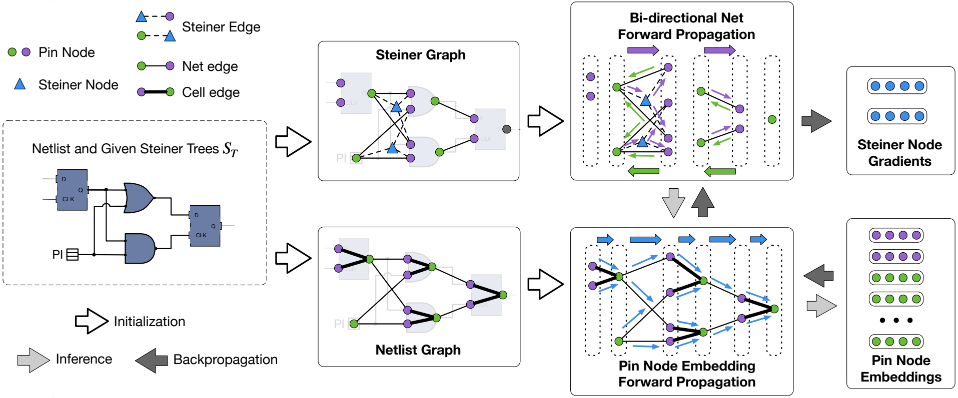
Sign-off Timing Optimization Gradients - Inference



The timing penalty can be calculated with,

$$P(\mathcal{T}(S_T)) = \lambda_w w(\mathcal{T}(S_T)) + \lambda_t t(\mathcal{T}(S_T)). \quad (1)$$

Sign-off Timing Optimization Gradients - Backpropagation



- The timing optimization gradients w.r.t. Steiner points positions $(\nabla_{X_s} P, \nabla_{Y_s} P)$ can be computed automatically via backward propagation.

Concurrent timing-driven Steiner Point Refinement

Adam Stochastic Optimizer

$$
\begin{aligned}
m_x^{(t)} &= (1-\beta_1) \cdot \nabla_{X_s^{(t)}} P, \;\; v_x^{(t)} = (1-\beta_2) \cdot (\nabla_{X_s^{(t)}} P \odot \nabla_{X_s^{(t)}} P), \\
X_s'^{(t)} &= X_s^{(t)} - \theta \cdot \frac{m_x^{(t)}}{\sqrt{v_x^{(t)}} + \epsilon},
\end{aligned} \tag{2}
$$

where θ is the stepsize to optimize Steiner point positions; β_1, β_2, and ϵ are the hyper-parameters.

Adaptive Stepsize Scheme - *Adaptive_Theta*

1. Obtain the initial timing gradient $(\nabla_{X_s} P, \nabla_{Y_s} P)$ w.r.t. the given Steiner point positions (X_s, Y_s).
2. Apply a small move:

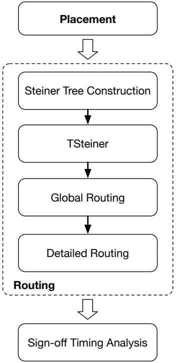
$$\begin{aligned} X_s' &= X_s + \alpha \nabla_{X_s} P, \\ Y_s' &= Y_s + \alpha \nabla_{Y_s} P, \end{aligned} \tag{3}$$

   where α is a hyper-parameter to control the scale of θ.
3. Obtain the updated timing gradient $(\nabla_{X_s'} P, \nabla_{Y_s'} P)$.

The adaptive stepsize is then calculated as:

$$\theta = \frac{|(X_s, Y_s) - (X_s', Y_s')|_2}{|(\nabla_{X_s} P, \nabla_{Y_s} P) - (\nabla_{X_s'} P, \nabla_{Y_s'} P)|_2}. \tag{4}$$

Setting



Cadence Innovus

FLUTE + edge shifting

CUGR

TritonRoute

Cadence Innovus

Table: Benchmark statistics.

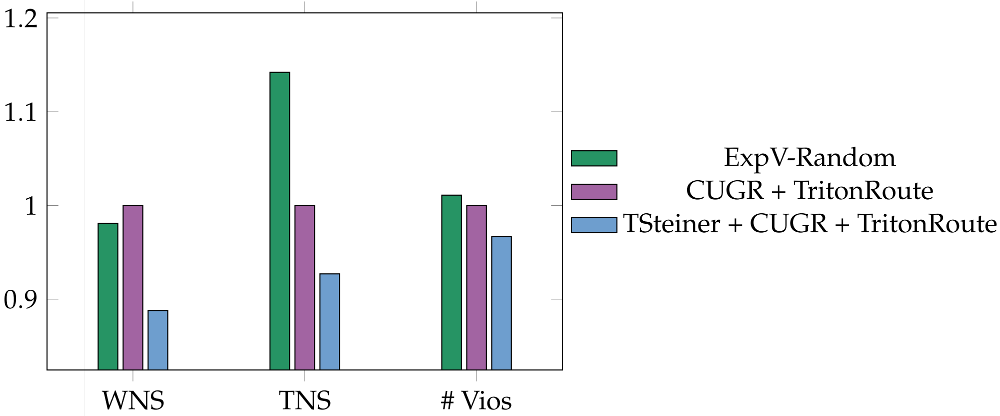
Benchmark	# Nodes Cell	# Nodes Steiner	# Edges Net	# Edges Cell	# Endpoints
chacha	15700	5398	44468	41204	1972
cic_decimator	781	196	2112	1982	130
APU	2897	1154	8373	7918	427
des	14652	5487	43065	40432	2048
jpeg_encoder	55264	15982	170520	161743	4420
spm	238	63	645	516	129
aes_cipher	11532	7323	37085	35825	659
picorv32a	13622	4542	41030	38191	1879
usb_cdc_core	1642	625	4632	3999	626
des3	47410	20004	136257	125093	8872
Total Train	89532	28280	269183	253795	9126
Total Test	74206	32494	219004	203108	12036

Results

Table: Experimental results on real-world open-source designs compared to the routing flow without integrating TSteiner.

Benchmark	CUGR + TritonRoute						TSteiner + CUGR + TritonRoute					
	WNS (ns)	TNS (ns)	# Vios	WL($\times 10^6$)	# Vias	# DRV	WNS (ns)	TNS (ns)	# Vios	WL($\times 10^6$)	# Vias	# DRV
aes_cipher	-11.246	-1516.9	512	984.971	109574	5	**-8.38**	**-1434.2**	**504**	984.527	109443	**3**
chacha	-48.538	-26259.1	1378	1,257.427	126600	2	**-46.68**	**-25375.7**	**1372**	1,258.011	126898	2
cic_decimator	-2.834	-169.981	72	16.466	5586	3	**-2.724**	**-161.436**	72	16.413	5593	3
picorv32a	-17.762	-441.607	67	727.216	109293	38	**-17.686**	**-434.443**	**56**	727.472	109311	**37**
usb_cdc_core	-5.914	-1365.2	347	49.351	12396	0	**-5.823**	**-1343.1**	**346**	49.117	12407	0
APU	-2.265	-33.713	25	101.179	23031	3	**-2.221**	**-33.598**	25	101.454	23101	3
des	-7.352	-405.427	341	682.828	115698	5	**-3.987**	**-227.331**	**285**	682.788	115599	5
jpeg_encoder	-74.342	-64909.2	**1967**	2,969.654	439126	1	**-70.629**	**-60789.1**	2007	2,973.304	439561	1
des3	-7.048	-1890	1512	2,680.848	372583	**48**	**-5.668**	**-1879.6**	**1509**	2,684.367	372768	49
spm	-0.817	-65.866	126	4.394	1553	2	**-0.782**	**-63.846**	126	4.399	1544	2
Average	1.000	1.000	1.000	1.0000	**1.0000**	1.0000	**0.888**	**0.929**	**0.967**	**0.9999**	1.0001	**0.9549**

Results



1.2
1.1
1
0.9
WNS
TNS
Vios
ExpV-Random
CUGR + TritonRoute
TSteiner + CUGR + TritonRoute

Conclusion

- We propose a deep learning-assist concurrent early-stage sign-off timing optimization framework, TSteiner.
- This study has raised the importance of Steiner point refinement for timing closure and provides a novel solution for early-stage timing optimization.
- TSteiner can be extended to more physical design stages since Steiner points exist not only in the pre-routing stage but also in routing solutions.
- The connections and the number of Steiner points may limit the optimization performance. => Search for new opportunities to support the flexibility.

THANK YOU!