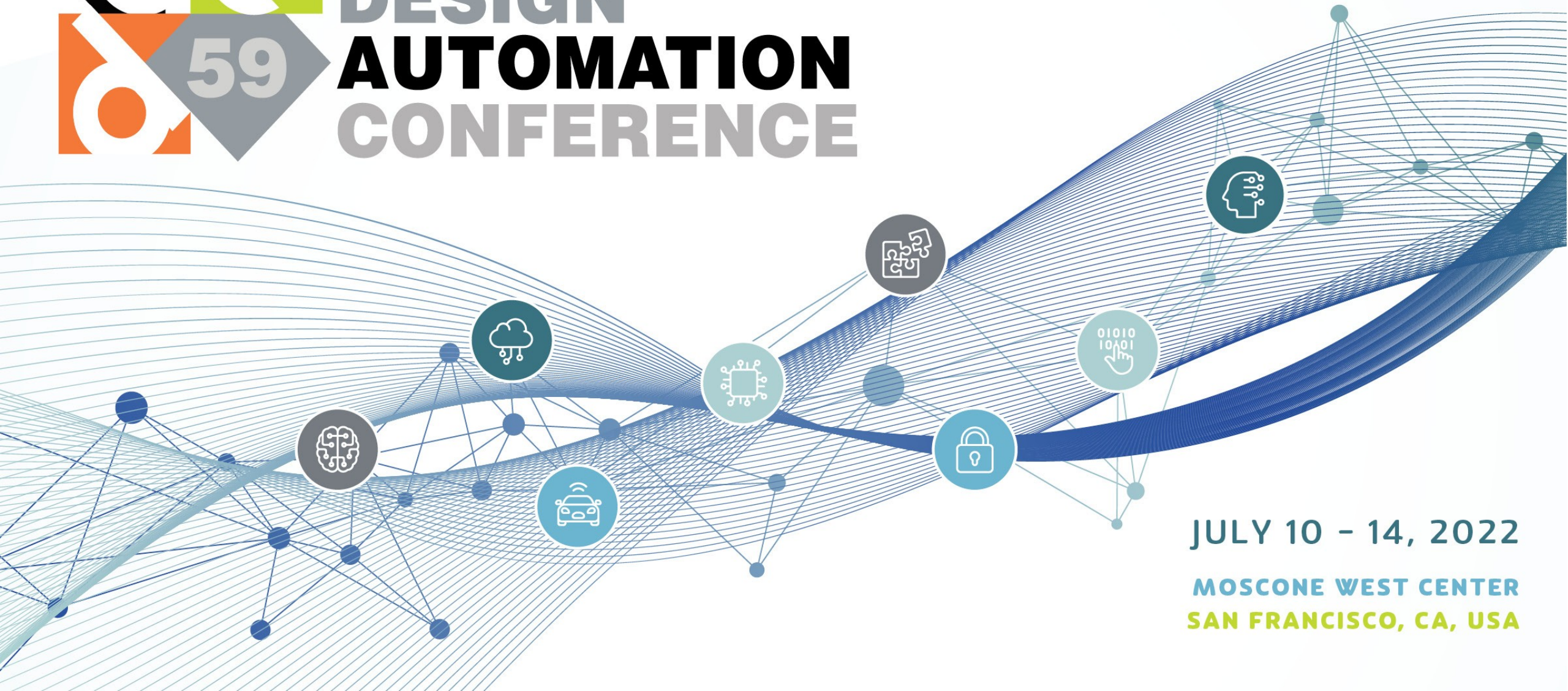
59 DESIGN AUTOMATION CONFERENCE



JULY 10 - 14, 2022

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

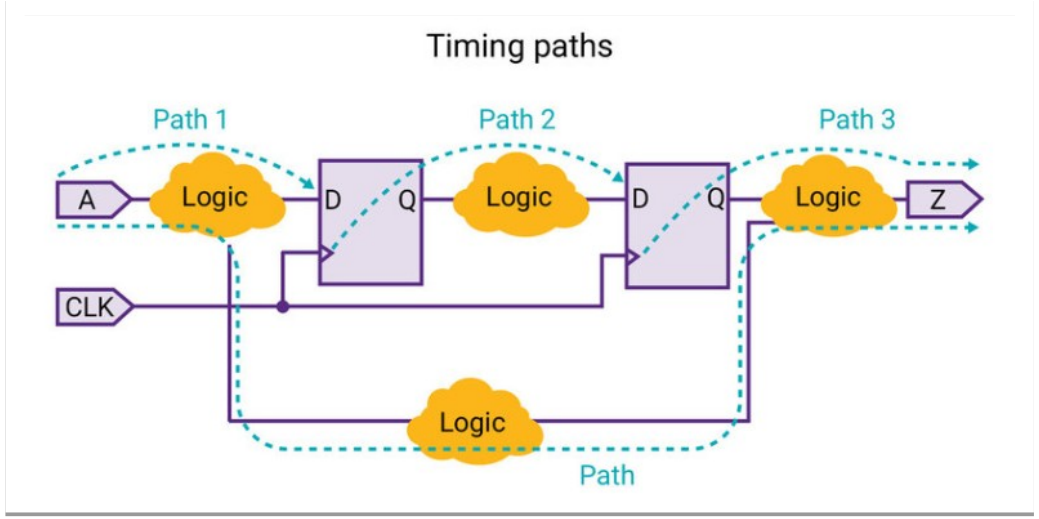
A Timing Engine Inspired Graph Neural Network Model for Pre-Routing Slack Prediction

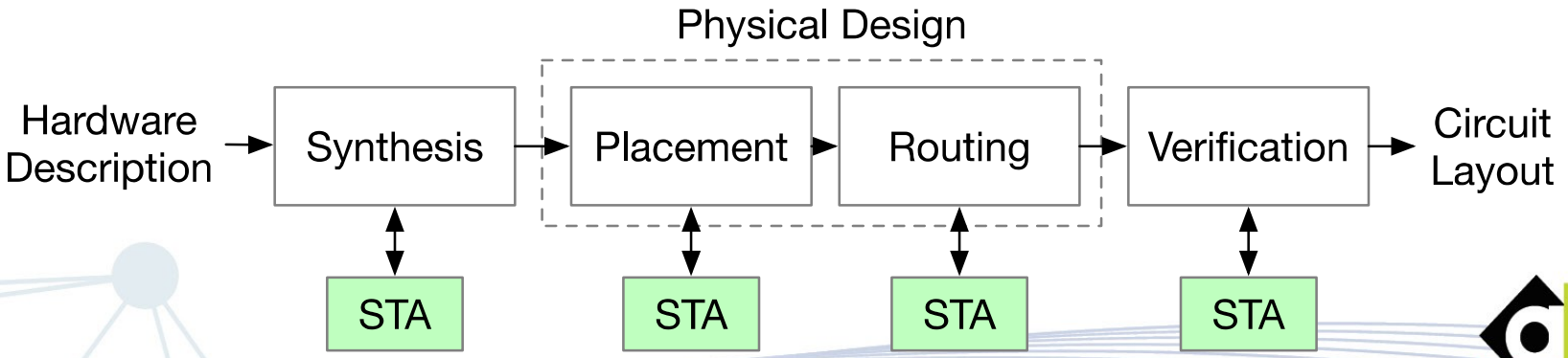
Zizheng Guo*, **Mingjie Liu***, Jiaqi Gu, Shuhan Zhang, David Z. Pan, Yibo Lin

Peking University, UT Austin

Static Timing Analysis (STA) in EDA

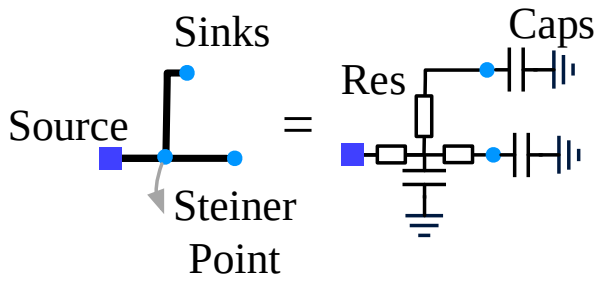
- Analyses circuit signal transition patterns and performs constraint checks.
- Determines circuit correctness & performance.
- Called during different design stages.

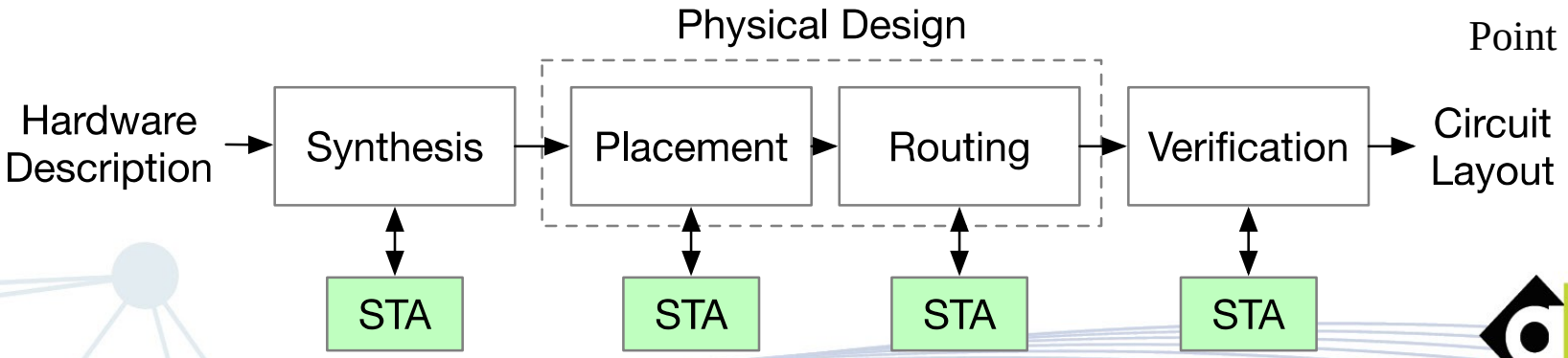




Predict Timing at Pre-Routing Stages

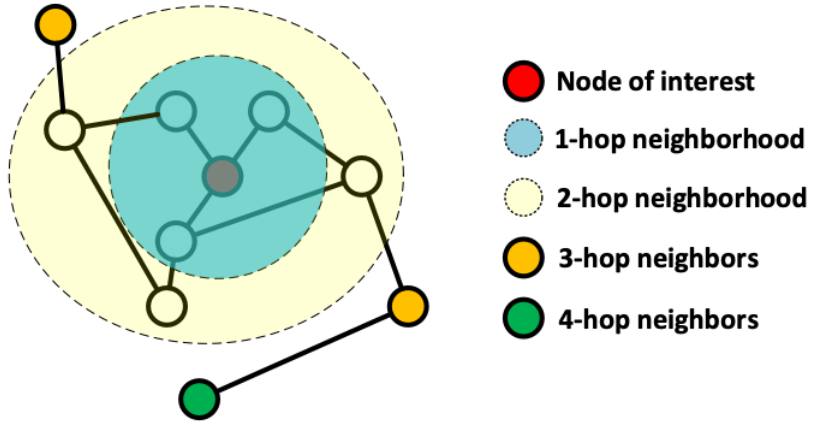
- With a complete design, STA engine performs exact analysis (verification).
- Without a complete design, STA engine gives **estimations**.
- E.g.: no wire parasitics available before routing. Thus, the wire delay need to be estimated during placement.





Challenges of Timing Prediction

- STA is complex (physical effects, iterations, …)
  - Cannot directly represent.
- Data-driven STA estimation: Machine Learning (ML).
  - Previous works limited to single-stage delay estimation.
  - There lacks global timing estimation (e.g. slacks).
- Graph neural networks (GNNs) can capture global information, but..
  - They suffer from "over-smoothing"
  - Limited receptive field.

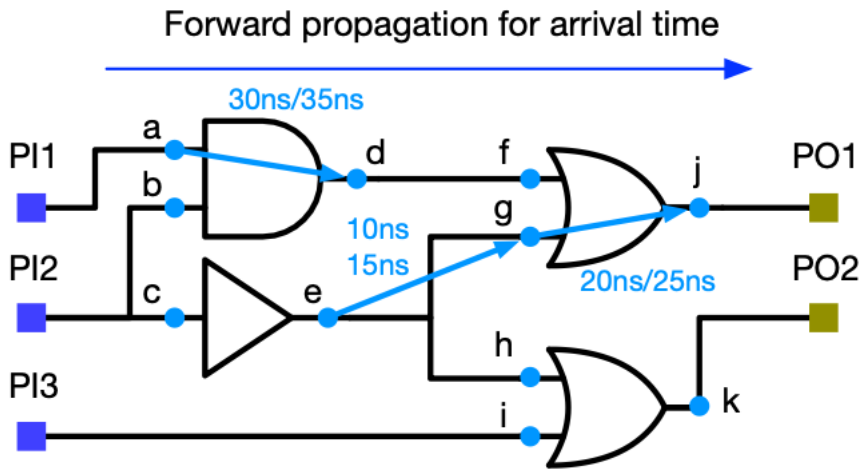


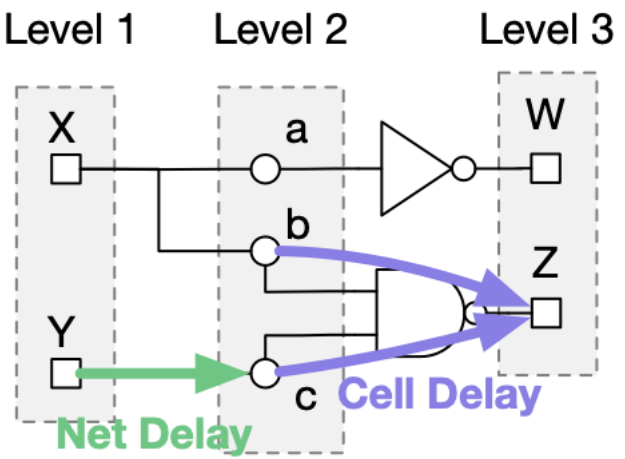
Contribution

- First ***end-to-end*** timing prediction framework, based on graph learning.
- Provide **direct signal arrival time estimation**.
- Customized GNN architecture with large receptive field, **inspired from real STA engines**.
- Auxiliary tasks to improve performance.
- Significantly better prediction accuracy on both single-stage delay and global timing.
- ALL **benchmarks, code, and models open-sourced**.

Timing-Engine-Inspired

- Two stages of a modern STA engine:
  - delay computation (local), and arrival time propagation (global, in-order)
  - => Two stage models with gradient connections.
- Propagation on directed graph
  - => ordered node embedding updates on **levelized** node lists.





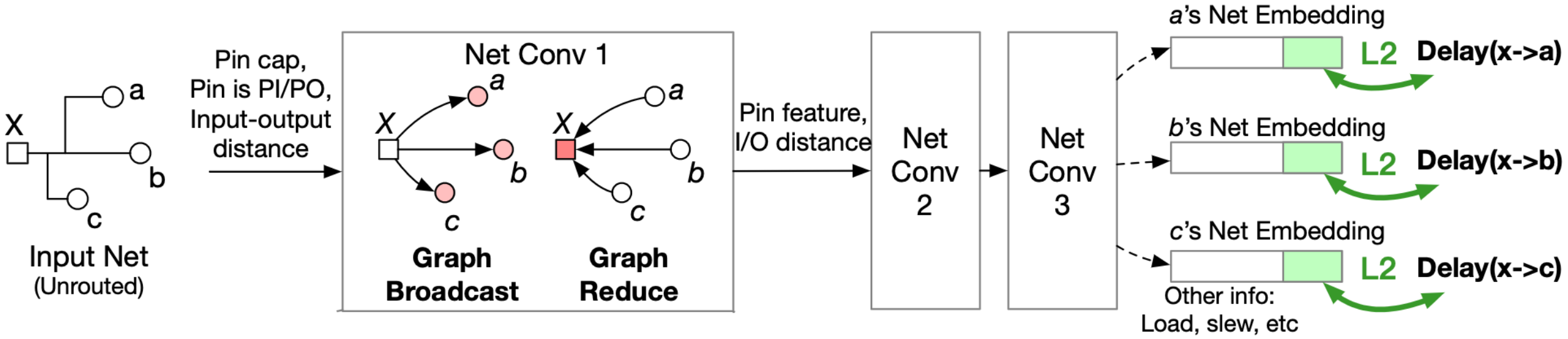
Data Representation

- Features/tasks for edges and nodes.
- Cell libraries included.

Type	Name	Size
Features	(Net) distances along x/y direction	2
	(Cell) LUT is valid or not	8
	(Cell) LUT indices	$8 \times (7 + 7)$
	(Cell) LUT value matrices	$8 \times (7 \times 7)$
Tasks	(Cell) edge delay	4 (EL/RF)
Total (Net)		2
Total (Cell)		516

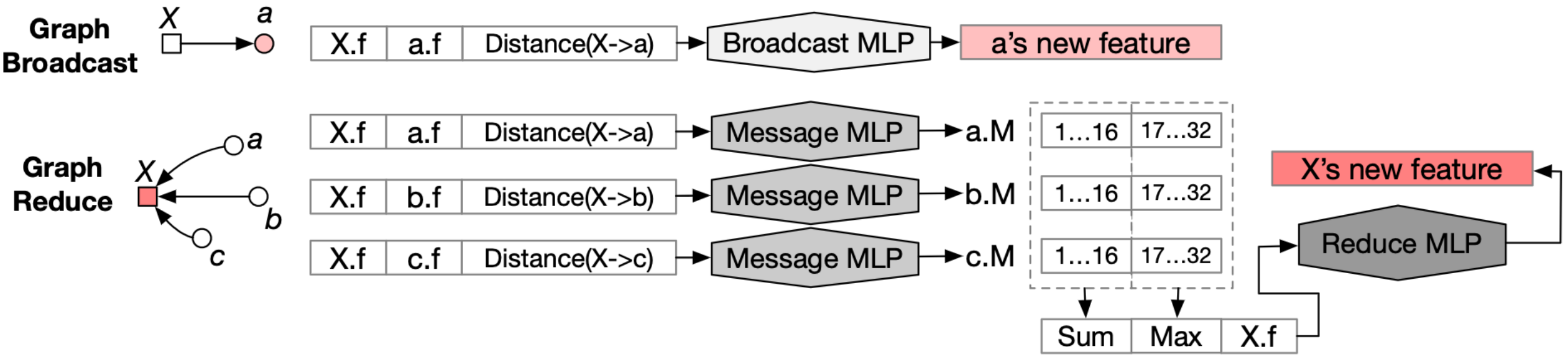
Type	Name	Size
Features	is primary I/O pin or not	1
	is fanin or fanout	1
	distance to the 4 die area boundaries	4
	pin capacitance	4 (EL/RF)
Tasks	net delay to root pin	4 (EL/RF)
	arrival time	4 (EL/RF)
	slew	4 (EL/RF)
	is timing endpoint or not	1
	required arrival time for endpoints	4 (EL/RF)
Total		27

Net Embedding Model



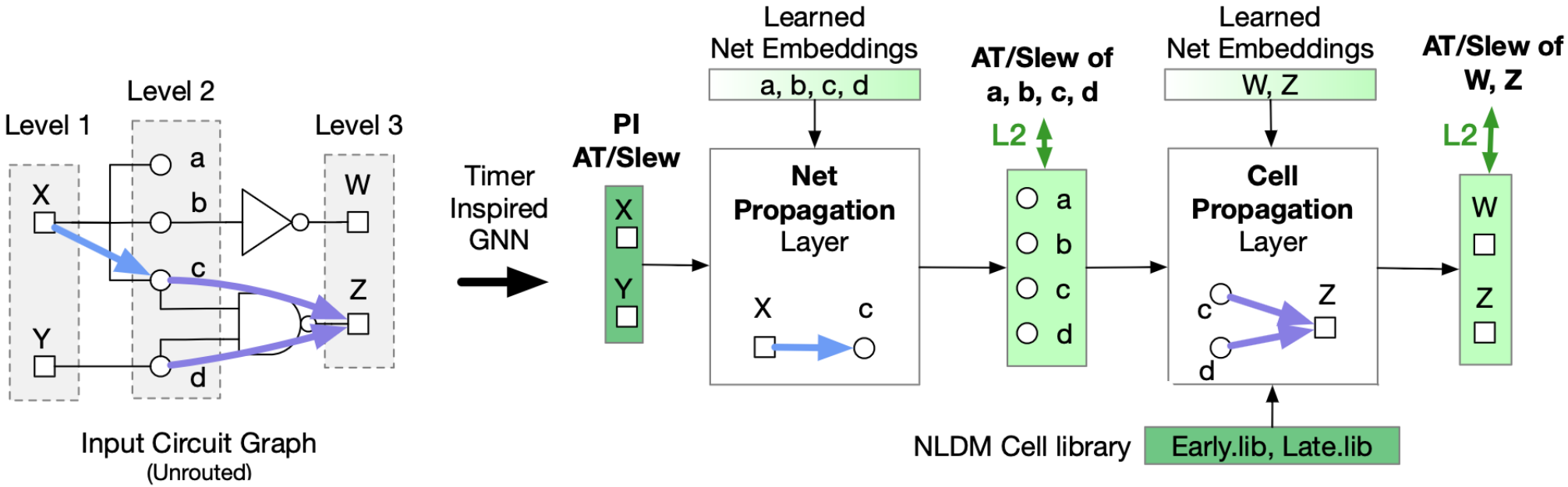
- 3 stacked net conv layers, with broadcast and reduce primitives.
- Half-supervised resulting-embeddings.
  - Delay; slew, load

Net Embedding Model



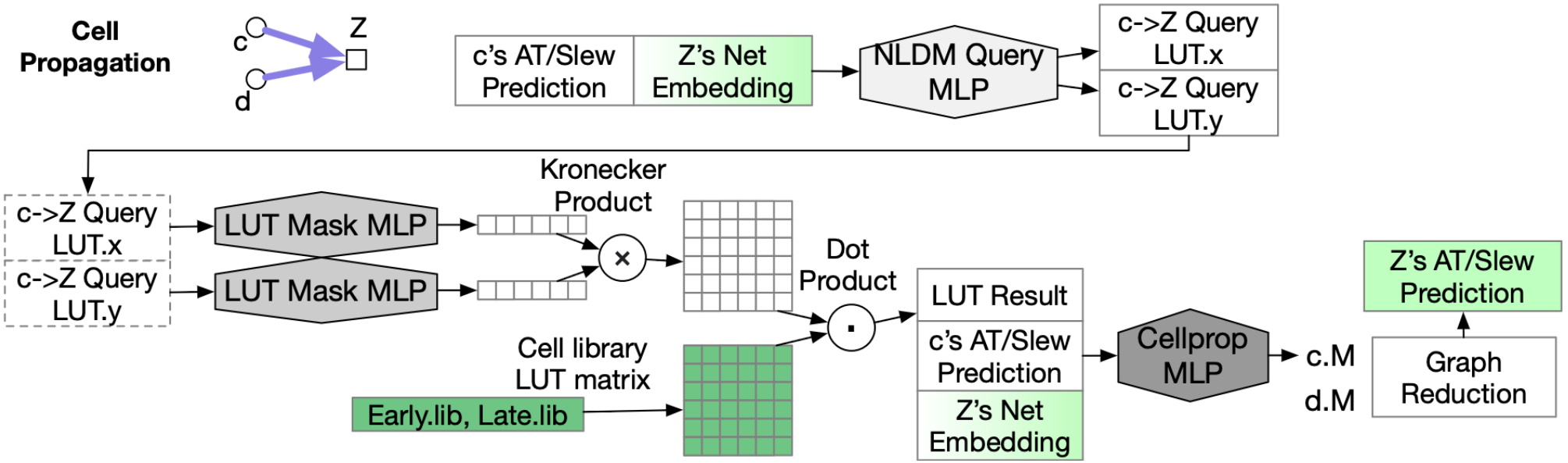
- Broadcast send information downstream.
- Reduce collect information upstream, and digest them through dual-channel sum/max ops.

Delay Propagation Model



- Graph is levelized. Then, net prop/cell prop layers alternate.
- Signal embeddings propagated given net delay and cell library as side inputs.

Delay Propagation Model



- Highly efficient cell library look-up table (LUT) query-layer.
- Dimension reduced through Kronecker product.

Auxiliary Task Supervision

- L2 of pin arrival time and slew.

$$\mathcal{L}_{atslew}(\theta, \phi | G, AS) = \frac{1}{N} \sum_{\text{node } p \in G} \left\| M^{\phi}_{prop\_atslew}(M^{\theta}_{net}(G)) - AS \right\|_2$$

- Auxiliary(1): L2 of cell delay.

$$\mathcal{L}_{celld}(\theta, \phi | G, CD) = \frac{1}{E_{cell}} \sum_{\text{cell arc } e \in G} \left\| M^{\phi}_{prop\_celld}(M^{\theta}_{net}(G)) - CD \right\|_2$$

- Auxiliary(2): L2 of net delay.

$$\mathcal{L}_{netd}(\theta | G, ND) = \frac{1}{N} \sum_{\text{fan-in node } p \in G} \left\| M^{\theta}_{net\_d}(G) - ND \right\|_2$$

Open Source

- https://github.com/TimingPredict/TimingPredict
- Fully open-source benchmarks:
  - From open source skywater130 PDK
  - From open source OpenROAD design flow, OpenSTA timing engine as baseline
  - Available to download w/o NDA restrictions.
- Models and code also available.

Results: Net Delay Prediction

- Our net embedding GNN layers compared with [Barboza, DAC 2019]
  - Random forest and MLP models on net statistics.
- We generalizes better.

	Statistics-based		Our GNN
	RF	MLP	
Avg. Train	**0.9944**	0.9550	0.9870
Avg. Test	0.9418	0.9357	**0.9552**

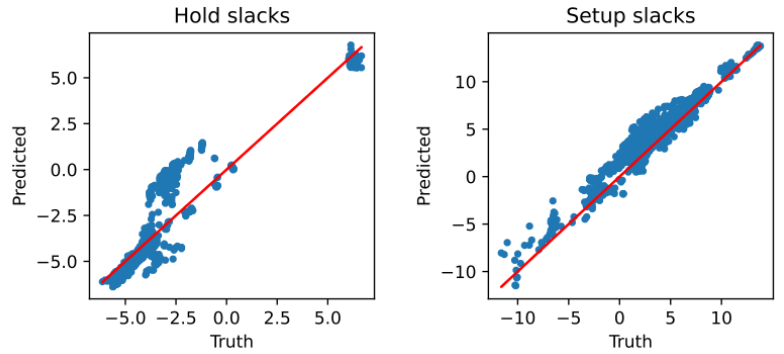
- Runtime compared with OpenROAD routing + OpenSTA:
  - **2664x** speed-up on average.

Results: Arrival Time Prediction

- Vanilla deep graph convolutional networks (GCNs), cannot learn useful knowledge even with a recent work (GCNII).
- Our timing-engine-inspired GNN architecture can achieve R2 > 0.89.
- Strongly-correlated slack prediction results.

Arrival Time / Slack Prediction (R^2 score)

	OpenROAD	Vanilla Deep GCNII			Our Timer-inspired GNN		
	Flow	4 layers	8 layers	16 layers	Full	w/ Cell	w/ Net
Avg. Train	1.0000	0.5710	0.3586	0.6810	**0.9493**	0.8215	0.9374
Avg. Test	1.0000	-0.8446	-0.7766	-1.5101	**0.8957**	0.8150	0.8513



Conclusion

- End-to-end graph learning framework for circuit timing estimation.
- Customized GNN architecture inspired from real STA engines.
- Both accurate and fast.
- All benchmarks, code, and models open-sourced.

- Future work:
  - Generalizability across different design processes
  - More complex timing effects: variation, aging, …

Thank You!

Questions are welcome