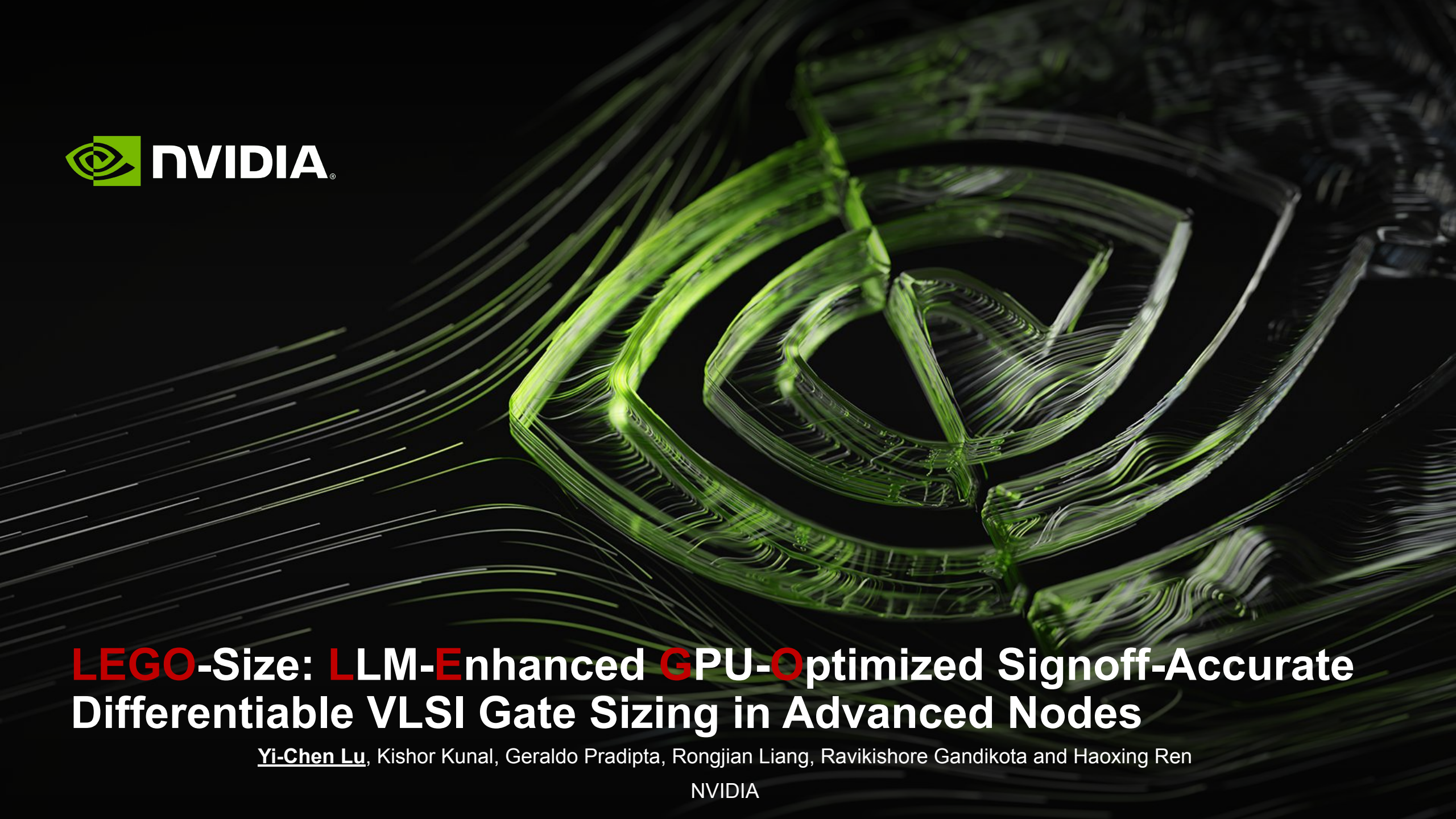


LEGO-Size: LLM-Enhanced GPU-Optimized Signoff-Accurate Differentiable VLSI Gate Sizing in Advanced Nodes

Yi-Chen Lu, Kishor Kunal, Geraldo Pradipta, Rongjian Liang, Ravikishore Gandikota and Haoxing Ren

NVIDIA

VLSI Gate Sizing with Large Language Models

Motivation & Key Idea

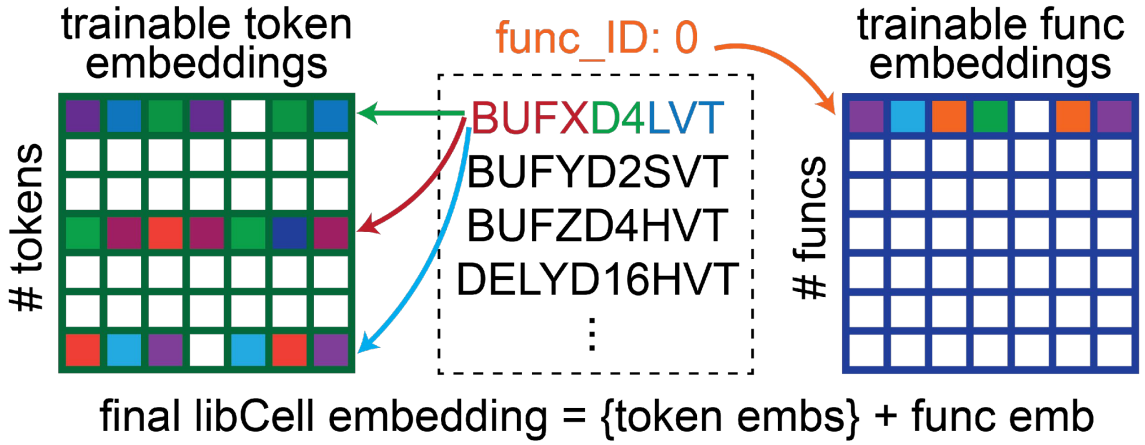
- Gate Sizing: A forever-important transform in Physical Design:
  - Gate sizing for PPA optimization is a fundamental transform used repetitively
  - Existing gate sizing approaches are time-consuming (especially in advanced nodes)
    - Commercial 3nm node offers more than 10 thousands libCells
    - Sensitivity-based approaches do not scale

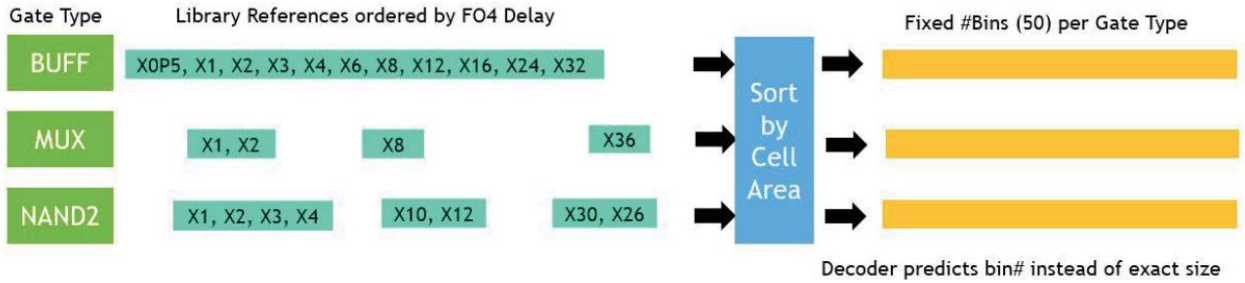
- Why LLMs can Help?
  - LLMs have proven super-human capabilities
    - No one cares about the Turing test anymore!
  - One key lesson has been taught:
    - With enough **data** and **compute**, we can do magic
    - □ path data are abundant in PD

- Key Idea of Gate Sizing as Language Modeling:
  - Each timing path is considered as a “sentence”
  - Library cells (libCells) along a path can be thought as “words”
  - Path-based sizing transform can be thought as sentence-to-sentence ”translation”

Philosophy of Customizing LLMs for PD Tasks

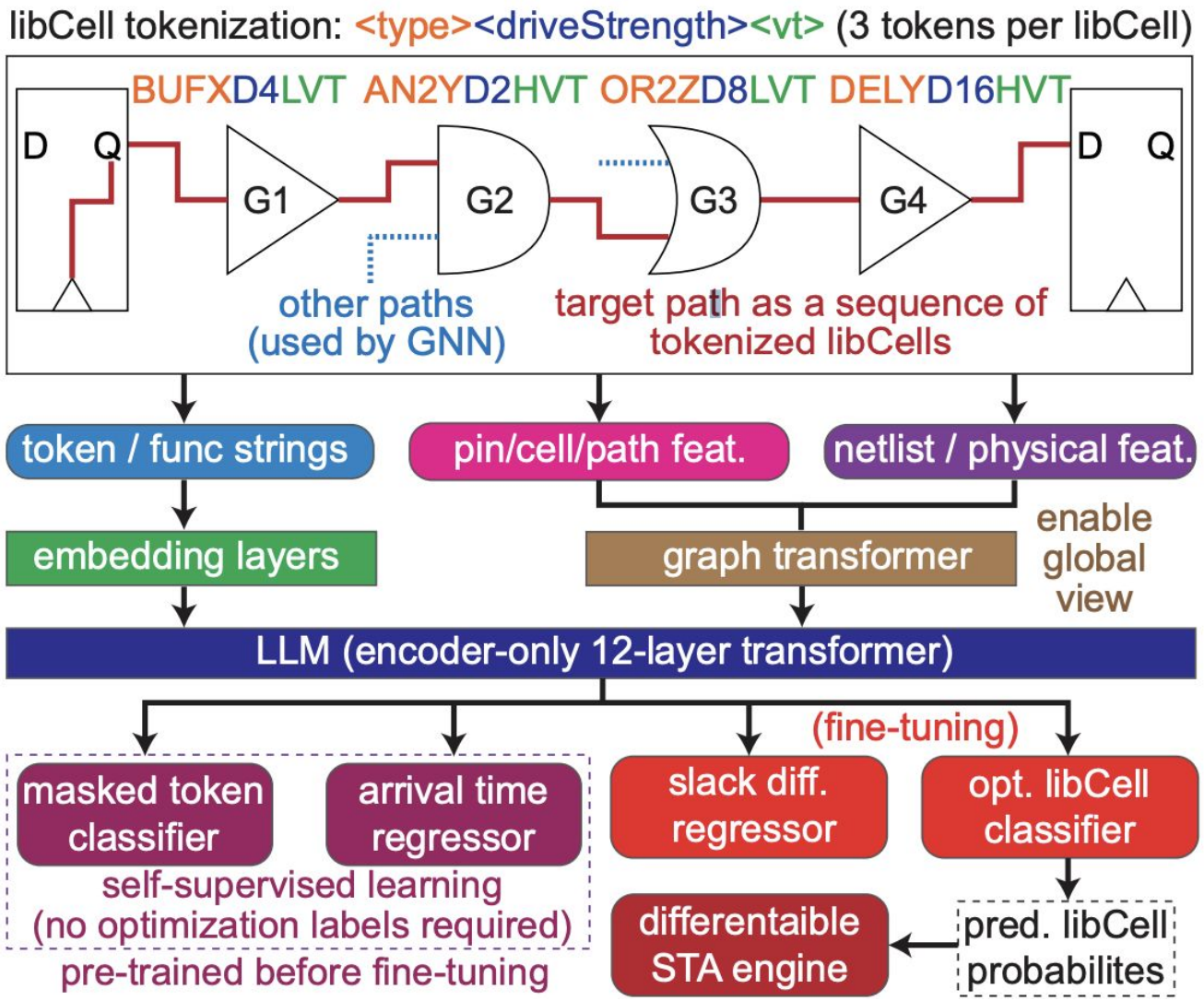
- Language modeling is powerful
  - But we need a “PD language” for true PPA gain
  - We need well-defined, explainable tokens
- TransSizer’s Limitation: Non-Exact Predictions
  - Gates are categorized to bins
  - Decoder predicts ‘bin ID’ and searches through final gate
  - We need exact language modeling
- Auto-regressive nature is not always good
  - Sampling drifts
    - Beam search can help but not much
  - Computationally inefficient
    - Decoding process is not parallelizable
  - We aim to explore encoder-only models
- Warm-up for unseen designs
  - We need self-supervised pre-training





tranSizer’s binning strategy:
same number of bins for different libCells

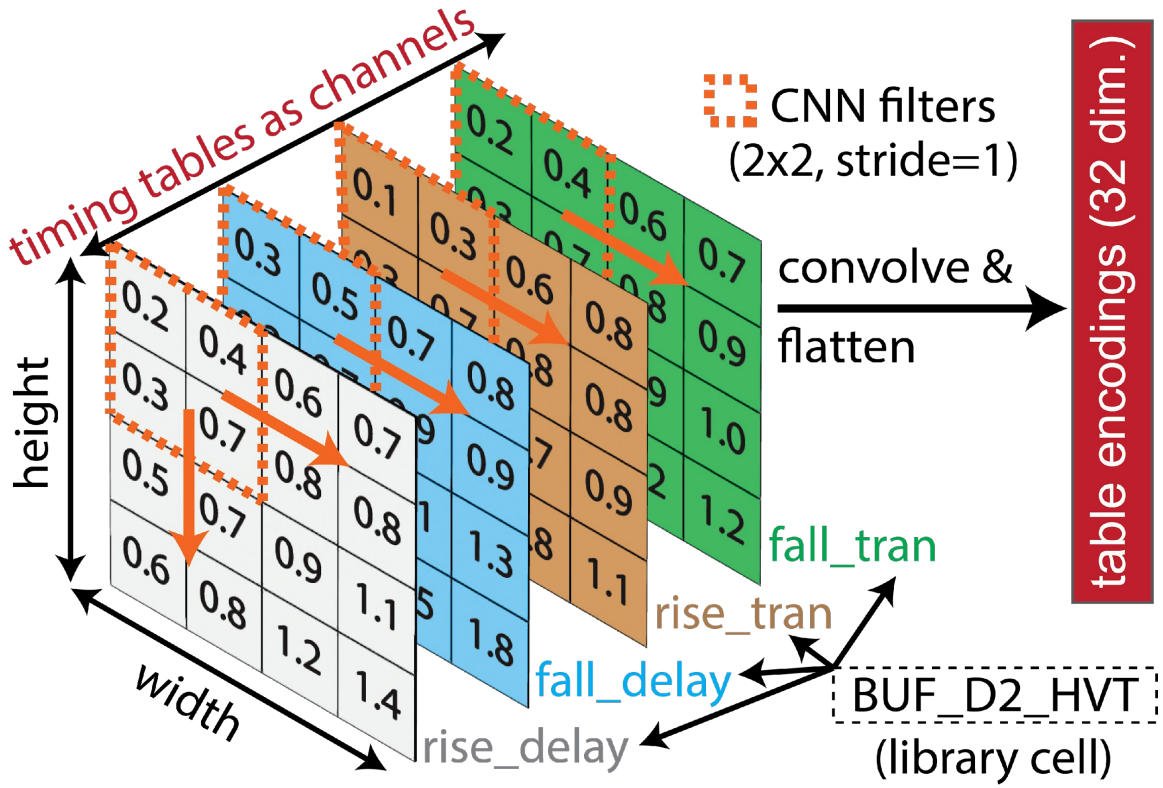
LEGO-Size Framework Overview



- A multi-stage framework that performs:
  - Self-Supervised Learning
    - Requires no optimization labels
  - Supervised Fine-Tuning (SFT)
    - Optimization labels from PrimeTime
  - Differentiable STA for probability refinement
    - Our in-house GPU-accelerated STA engine (INSTA)
- Self-Supervised Learning Tasks:
  - Masked Token Prediction
  - Arrival Time Prediction
- Supervised Fine-Tuning Tasks:
  - Slack improvement prediction
  - Final gate type prediction
- Why Tokenization?
  - Drastically helps modeling generalization.
  - Example:
    - There are 10k cells in our 3nm library
    - But can be represented by ~300 tokens

$$\#\ \text{all libCells} = \{\text{type tokens}\} \otimes \{\text{drv tokens}\} \otimes \{V_{th}\ \text{tokens}\}$$

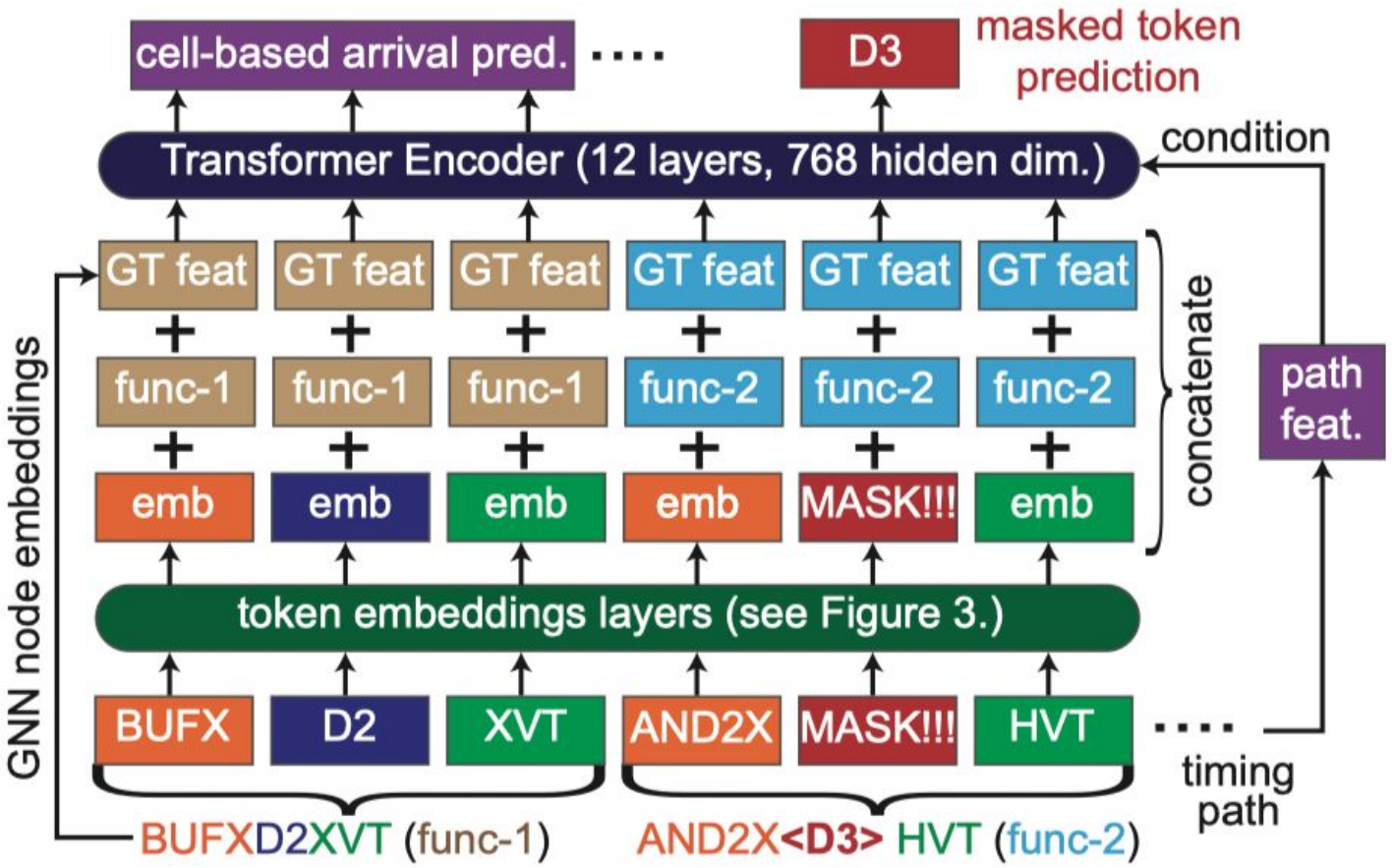
Library Tables Encoding using CNN



- Goal:
  - Construct generic representation for lib tables
- Idea:
  - Each table can be viewed as ‘channel’
  - Similar to RGB channels in image

Self-Supervised Pre-Training

construct PD-specific token embeddings

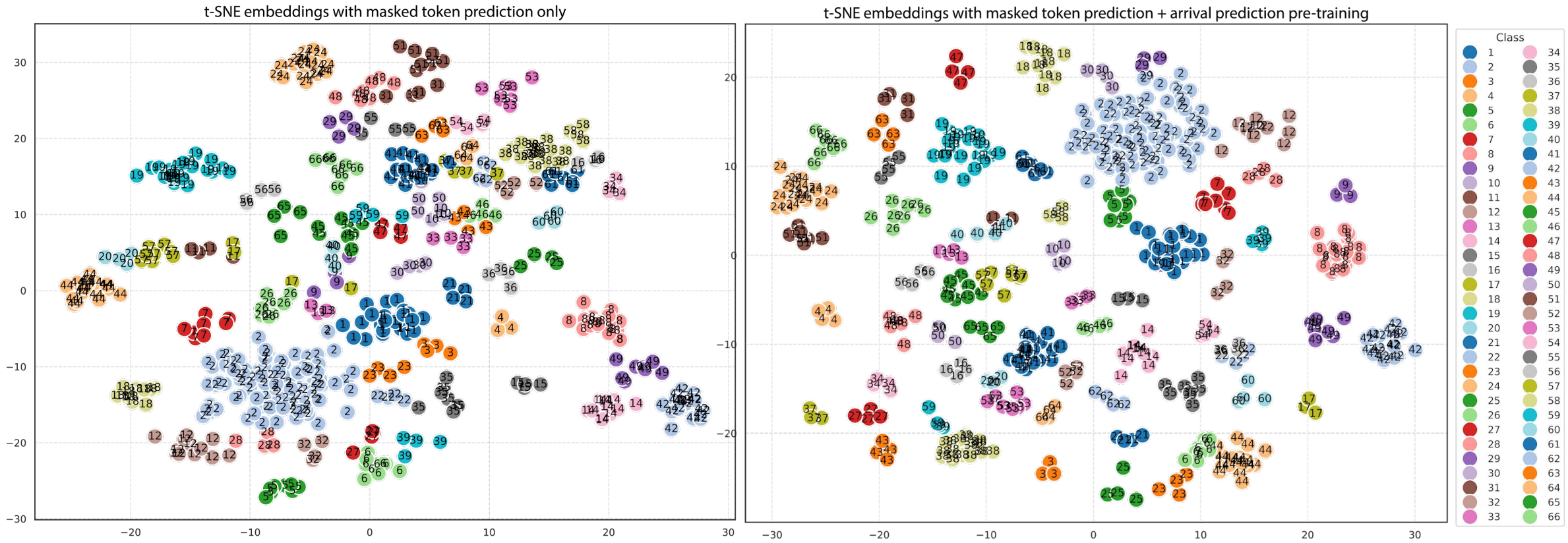


- Masked Token Prediction (termed as MLM)
- Stage-based Arrival Increment Prediction
- Key Idea:
  - randomly masked out some tokens (libCells) along a path
  - given the RC features, reconstruct the masked tokens

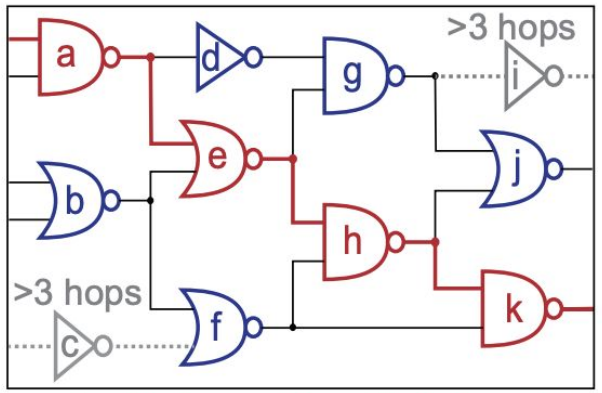
Ablation Study: Self-Supervised Pre-Training on LibCell Embeddings

Left figure: pre-trained with masked token prediction only
Right figure: pre-trained with masked token prediction and arrival prediction

each point is a “lib type” (e.g, BUFFSK, BUFCS) and each class denotes a unique functionality



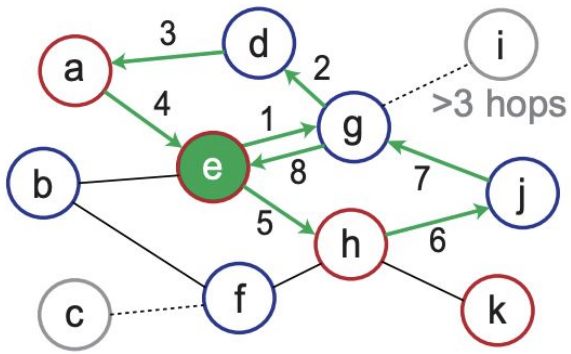
Improving Path-Based Optimization with Global View



target cells (on criti. paths): {a, e, h, k}
support cells (<= 3hops): {b, d, f, g, j}
dont care cells (>3 hops from criti.): {c, i}

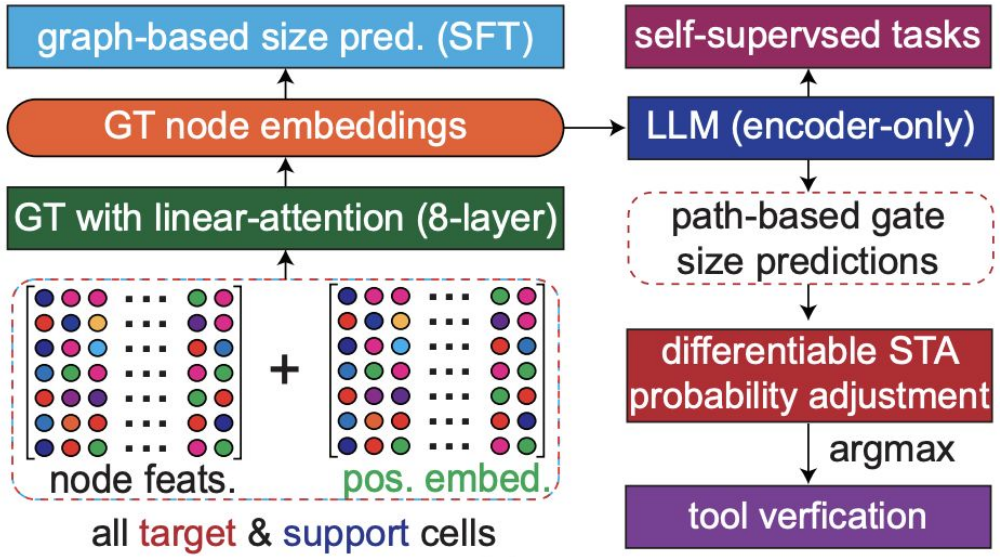
(a)

(arc-based graph construction)



build positional embeddings (PE)
from fixed-length random walk (RW)
(RW generates probability vectors)

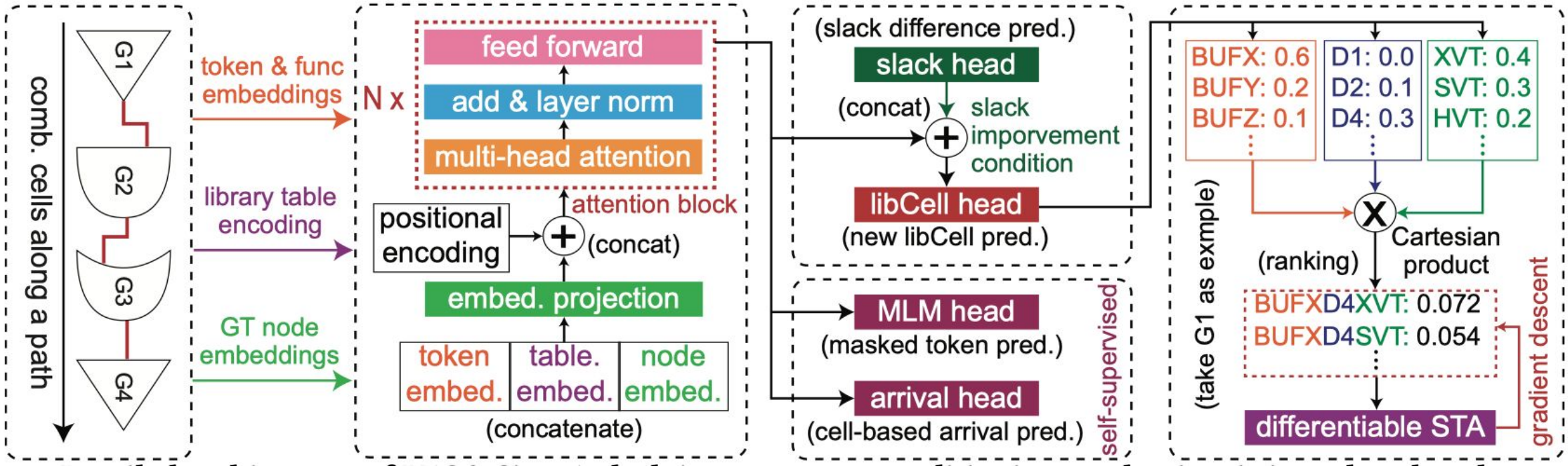
(b)



(c)

- Key Idea:
  - Incorporate neighborhood information in path-based optimization
  - Improve GNN with a linear-attention mechanism to consider more nodes jointly
- Details about random walk PE:
  - Assume $A \in N\ x\ N$ and $D \in N\ x\ 1$. → Define $RW = AD^{-1} \in N\ x\ N$ → basically normalize each row by self degree
  - PE of a node v: $PE_v = [RW_{v,v}, RW^2_{v,v}, RW^3_{v,v}, \ldots, RW^{10}_{v,v}]$. → the probability of returning to each node in different length of walks. We use k=12 in this work.

Detailed Architecture of LEGO-Size

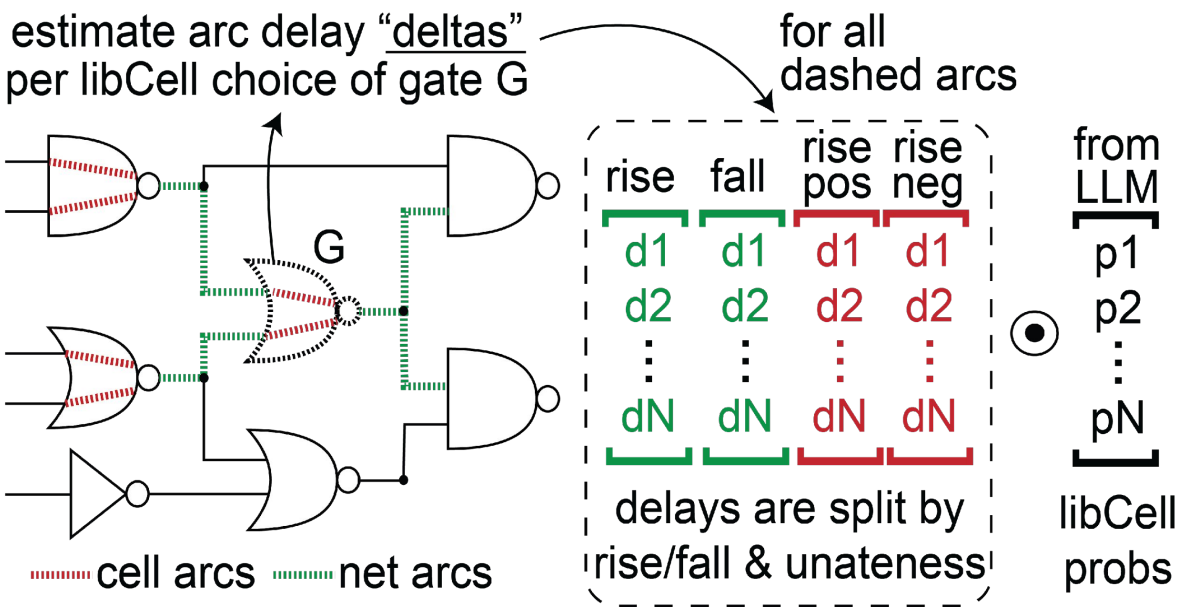


- Slack improvement conditioning
  - Idea: we let the model know how much slack to improve in advance
  - trained with groundtruth labels (teacher-forcing)

- A differentiable, GPU-accelerated STA engine is used to adjust probabilities

Estimate ECO for Sub-Graph Reannotation

Extremely fast with parallel execution
1m+ (cell, libCell) pairs take a few seconds

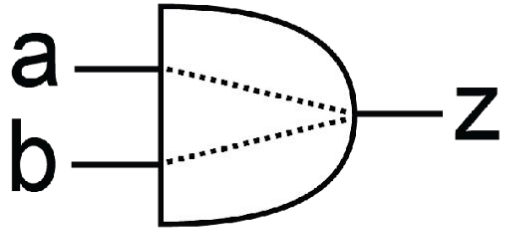



```
cell name       : wishbone/bd_ram/U9/U6200
current lib cell: slow/CLKINVX4

new lib cell: slow/INVX20
max_rise                                  current          estimate
-------------------------------------------------------------------
area                                       13.310            63.200
input net delay                             0.005             0.005
stage delay                                 2.727             1.062
  cell delay                                2.609             0.957
  output net delay                          0.118             0.105
  delta delay                               0.000             0.000
```

$$AT_Z = \tau * \log(\exp\{\frac{AT_a + d_{az}}{\tau}\} + \exp\{\frac{AT_b + d_{bz}}{\tau}\})$$

$$\frac{\partial AT_z}{\partial AT_a} = \frac{\partial AT_z}{\partial d_{az}} = \frac{\exp\{\frac{AT_a+d_{az}}{\tau}\}}{\exp\{\frac{AT_a+d_{az}}{\tau}\} + \exp\{\frac{AT_b+d_{bz}}{\tau}\}}$$

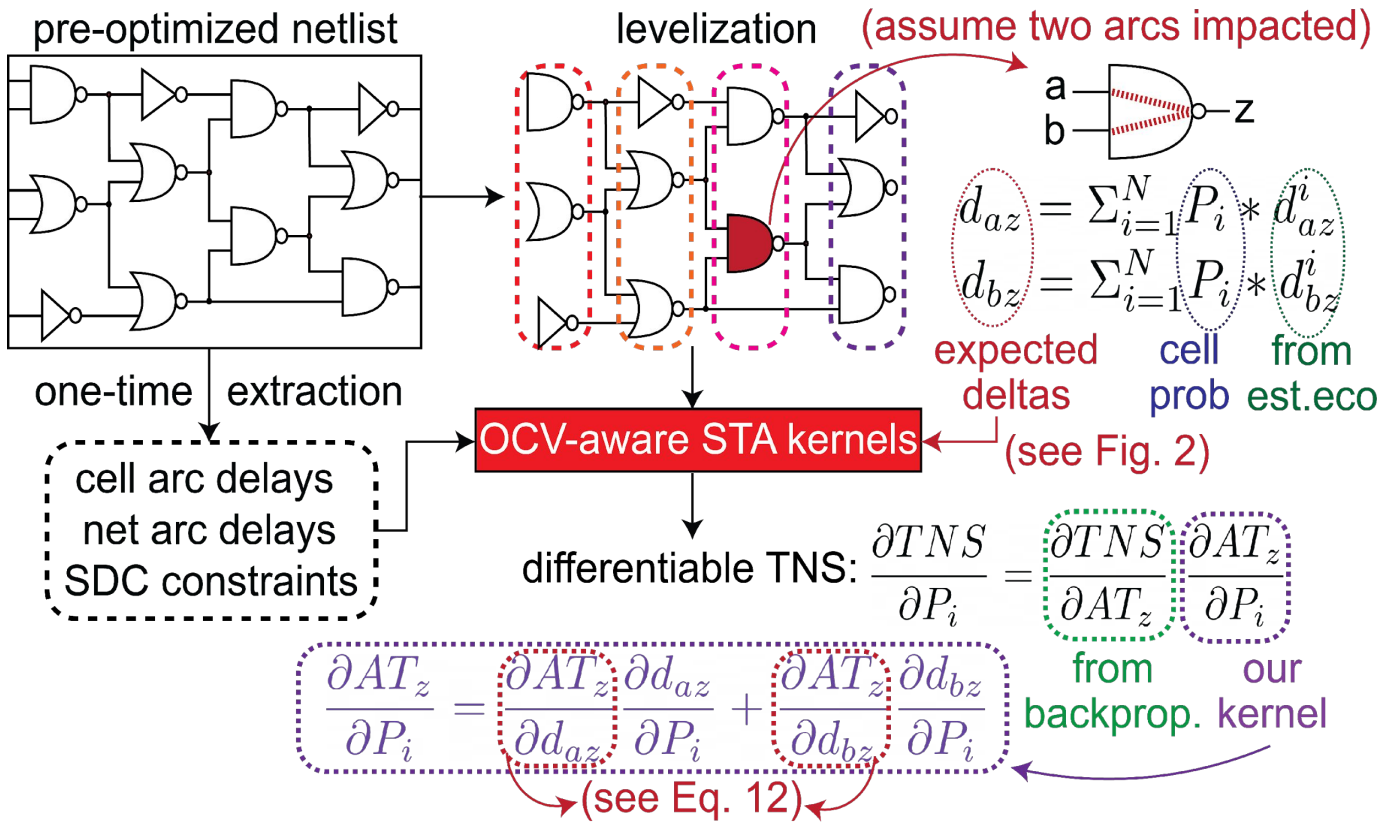


$$d_{az} = \Sigma_{i=1}^{N} P_i * d_{az}^i$$

$$\frac{\partial d_{az}}{\partial P_i} = d_{az}^i$$

$$\frac{\partial AT_z}{\partial P_i} = \frac{\partial AT_z}{\partial d_{az}} \frac{\partial d_{az}}{\partial P_i} = d_{az}^i * \frac{\exp\{\frac{AT_a+d_{az}}{\tau}\}}{\exp\{\frac{AT_a+d_{az}}{\tau}\} + \exp\{\frac{AT_b+d_{bz}}{\tau}\}}$$

Complete Differentiable Gate Sizing Process



- LibCell probabilities are from LLM predictions
- Delta-delays are estimated by a commercial tool
  - Each associated with a LLM-predicted prob.
- A differentiable STA engine is leveraged to perform differentiable timing propagation
  - Differentiable w.r.t. LLM-predicted probs.
- Remove path-based ambiguity!
  - By graph-based refinement

Experimental Setup and Optimization Results

- We collected 21 PD implementations with different recipes from 5 high-performance designs
- We generated around 28M timing paths with optimization labels from a commercial tool
- The pre-training of 28M paths spans across a week on a server with 8 A100 GPUs.
- For validation, we leave an implementation to be unseen and fine-tune on the remaining ones.

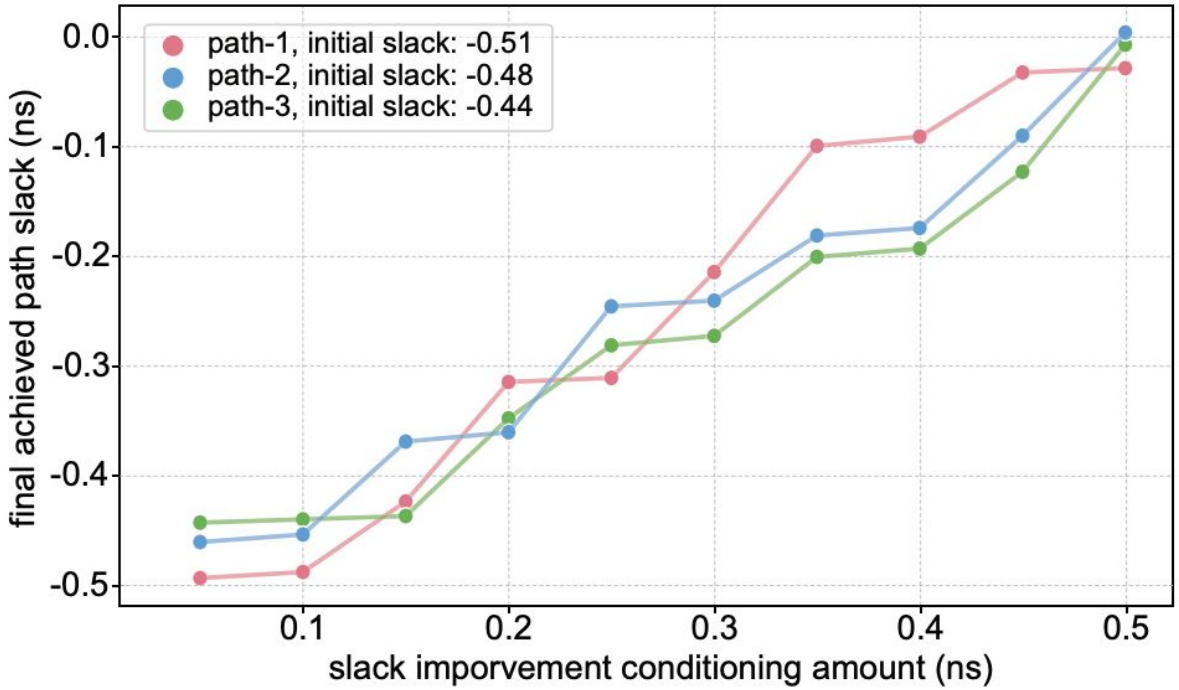
unseen design (# cells)	initial state (pre-optimized)				commercial signoff tool (32 threads)					LEGO-Size (GT + LLM + Differentiable STA)					
	WNS	TNS	#vio. EPs	total power	WNS	TNS (goal)	#vio. EPs	total power	sped-up	WNS	TNS (goal)	#vio. EPs	total power	sped-up	Δarea (um^2)
block1 (1.8M)	-0.066	-17.46	2410	238.6	-0.028	-1.953	271	238.8	1.00	-0.034	-1.237 (-36.7%)	184 (-47.3%)	238.8	125x	-1.42
block2 (1.3M)	-0.041	-30.38	5410	293.4	-0.038	-7.82	1201	293.7	1.00	-0.035	-5.47 (-30.0%)	943 (-27.4%)	293.9	111x	-0.36
block3 (1.2M)	-0.054	-10.03	1539	119.7	-0.032	-5.37	906	119.8	1.00	-0.033	-4.32 (-19.5%)	812 (-10.4%)	119.8	100x	+2.28
block4 (1.5M)	-0.102	-59.25	6955	248.5	-0.097	-24.77	1912	249.1	1.00	-0.083	-20.36 (-17.8%)	1658 (-15.3%)	249.3	119x	+3.52
block5 (1.4M)	-0.072	-11.33	1525	127.2	-0.038	-6.68	620	127.5	1.00	-0.036	-4.85 (-27.4%)	529 (-17.2%)	127.5	114x	-1.15

Ablation Studies

Table 4: Ablation study on optimization results of predictive models. Refer to Table 3 for commercial tool baseline.

Design	Method	WNS	TNS	NVE	F1	Speedup
block1	TransSizer [27]	-0.045	-7.43	1072	0.65	367x
	ECO-GNN [24]	-0.048	-11.06	1157	0.56	1575X
	GT-only (ours)	-0.041	-9.84	1068	0.61	1427X
	GT+LLM (ours)	-0.032	-2.32	483	0.92	870X
block4	TransSizer [27]	-0.114	-35.49	3419	0.70	328x
	ECO-GNN [24]	-0.104	-32.12	3270	0.67	1362X
	GT-only (ours)	-0.094	-29.83	2709	0.74	1185X
	GT+LLM (ours)	-0.086	-22.33	1852	0.87	627X

(optimization results by predictive models only)



Slack-conditioning for PPA-tradeoff

Conclusion and Future Work

- We show that language modeling techniques can be applied to PD for PPA optimization
  - Particularly, the effectiveness of introducing “PD tokens” for handling complex design space
- We show that GPU-powered differentiable techniques are effective for large-scale optimization
  - which further improves the limitations of predictive models
- In the future, we plan to improve LEGO by:
  - Simultaneously tackling more PPA objectives
    - Or having them as constraints
  - Multi-scenario handling
    - Currently focus on the dominant scenario

Thank you for listening!