NetTAG: A Multimodal RTL-and-Layout-Aligned Netlist Foundation Model via Text-Attributed Graph

Wenji Fang[1], Wenkai Li[1], Shang Liu[1], Yao Lu[1],
Hongce Zhang[2], Zhiyao Xie[1]

[1] *Hong Kong University of Science and Technology*
[2] *Hong Kong University of Science and Technology (Guangzhou)*

62 THE CHIPS TO SYSTEMS CONFERENCE

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

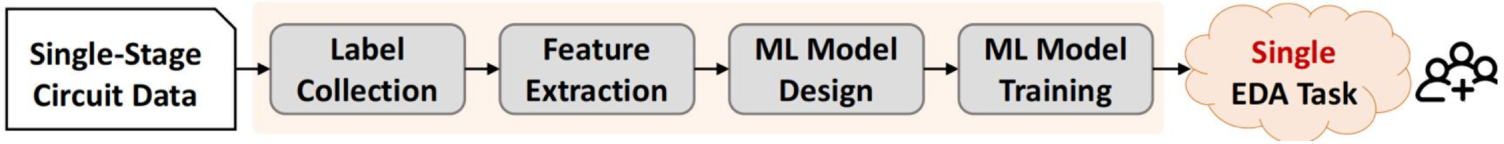
SPONSORED BY CEDA SIGDA

Circuit Foundation Model

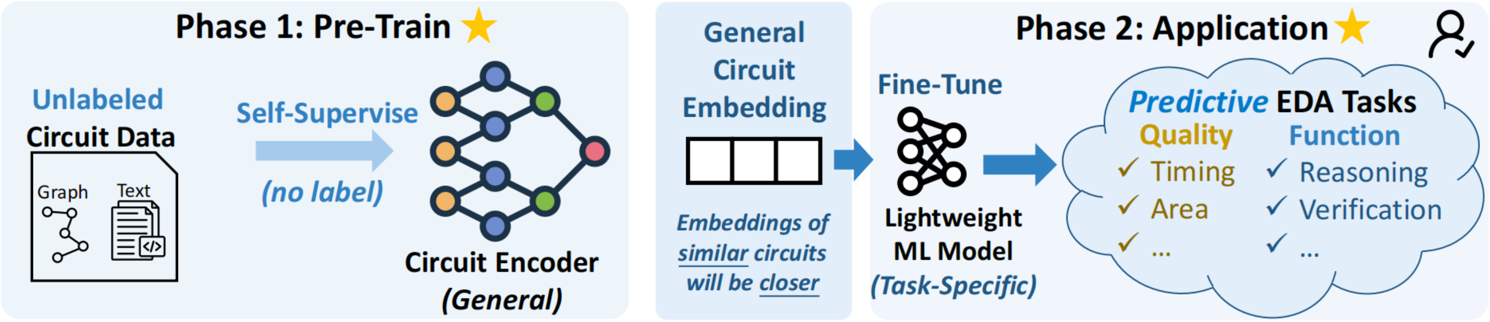
THE CHIPS TO SYSTEMS CONFERENCE

From Task-Specific to Circuit Foundation Model

- **Traditional: task-specific supervised predictive AI for EDA**
  - Tedious, time-consuming, not generalized



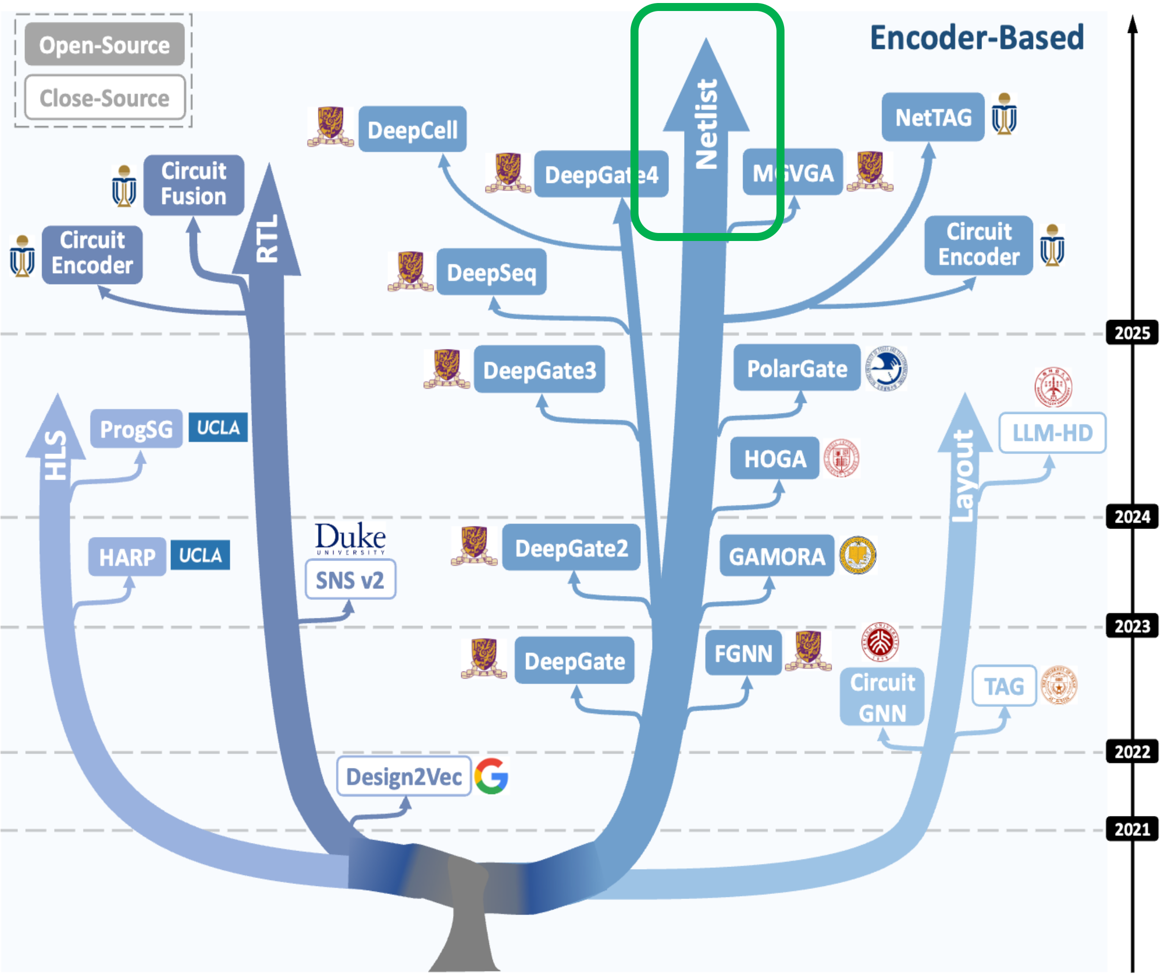
- **New trend[1]: general self-supervised Circuit Foundation Model**
  - Encode circuit into general embeddings for various EDA tasks (circuit encoder)
    - *Pre-training* to capture circuit intrinsics (self-supervised)
    - *Fine-tuning* for EDA tasks (supervised)



[1]Wenji Fang et al. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA

Circuit Representation Learning (Encoders)

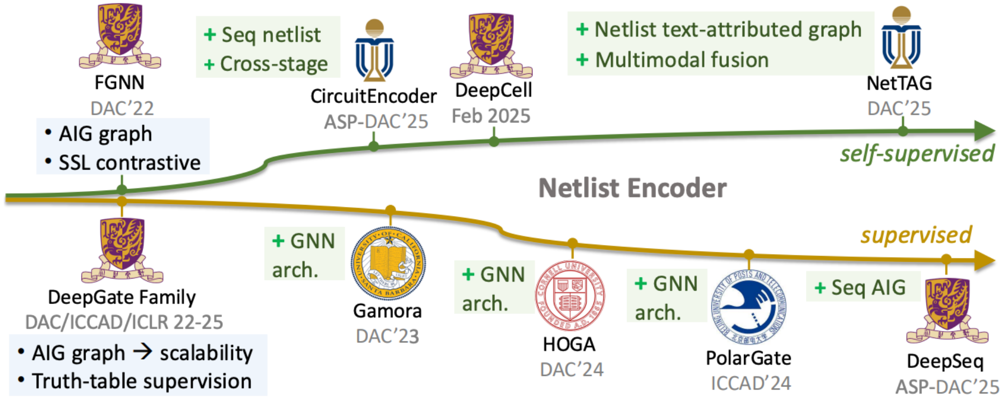
- **Netlist encoder:** **most actively explored**



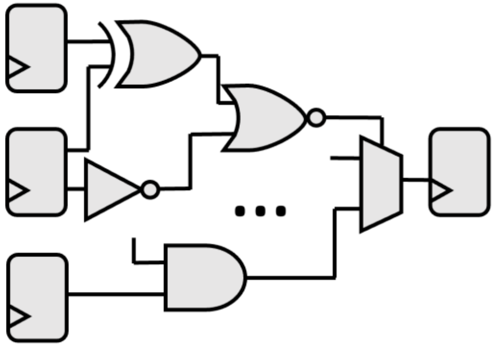
[1]Wenji Fang et al. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA

Existing Netlist (AIG) Encoders

- **Limitation: graph-only, only supports AIG**
  - Graph **structure** over **functionality**
    - Limited to AIG, post-syn netlists?
    - Rely on functional supervision (e.g., truth table)
    - Lack of physical encoding, PPA tasks?
- **How about LLM?**
  - Textual **semantics** rather than **structure**
    - Struggle with netlists (low-level, bit-blasted)
    - Lack of structural encoding



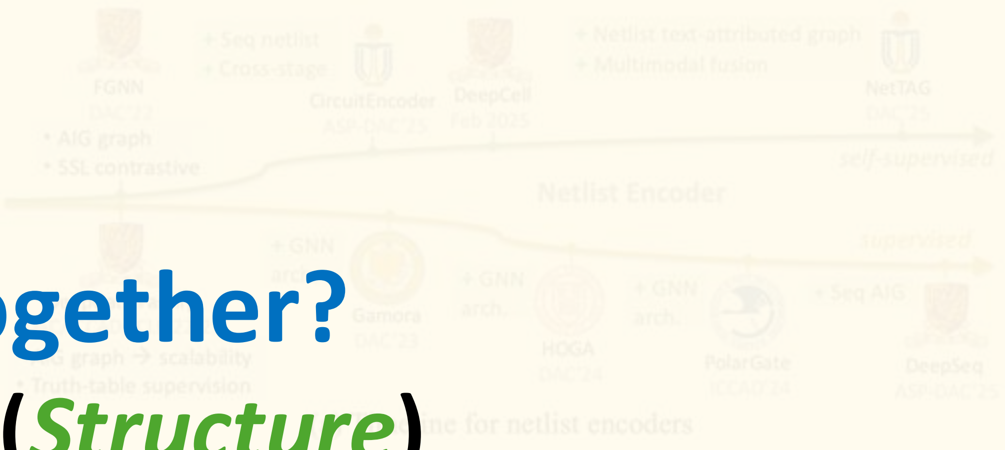
(a) Timeline for netlist encoders



[1]Wenji Fang et al. A Survey of Circuit Foundation Model: Foundation AI Models for VLSI Circuit Design and EDA

Motivation: LLM + Graph

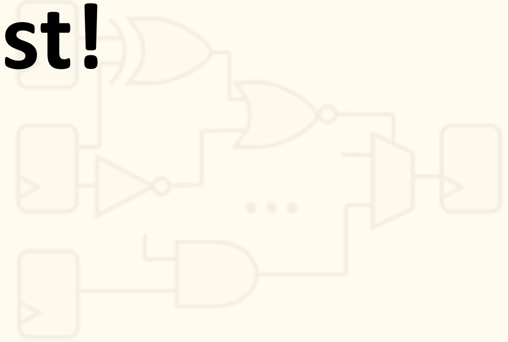
- Limitation: graph-only, only supports AIG
  - Graph structure over functionality
    - Limited to AIG, post-syn netlists?
    - Rely on fu[illegible]
    - Lack of physical encoding, PPA tasks?
- How about LLM?
  - Textual semantics rather than structure
    - Struggle with netlists (low-level, bit-blasted)
    - Lack of structural encoding



Can we fuse them together?

LLM (*Semantic*) + Graph (*Structure*)

Multimodal Fusion on Netlist!

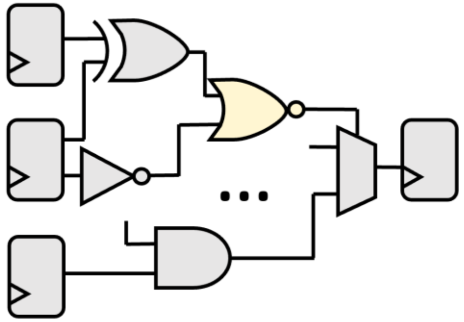


Multimodal Fusion on Circuit: RTL vs Netlist

- **Multimodal learning: fuse information from diverse modalities**
  - Vision-language
  - ……

- **Multimodal learning on RTL[2]**
  - ***Register-transfer*** *level* (RTL)
  - *Earlier* stage → more ***semantic***
  - Fuse 3 RTL modalities at ***register level***

- **Multimodal learning on netlist？**
  - ***Gate****-level* netlists
  - *Later* stage → more ***structure***
  - Should fuse at ***gate level***



[2]Wenji Fang et al. CircuitFusion: Multimodal Circuit Representation Learning for Agile Chip Design. [ICLR'25]

NetTAG: Multimodal Netlist Foundation Model

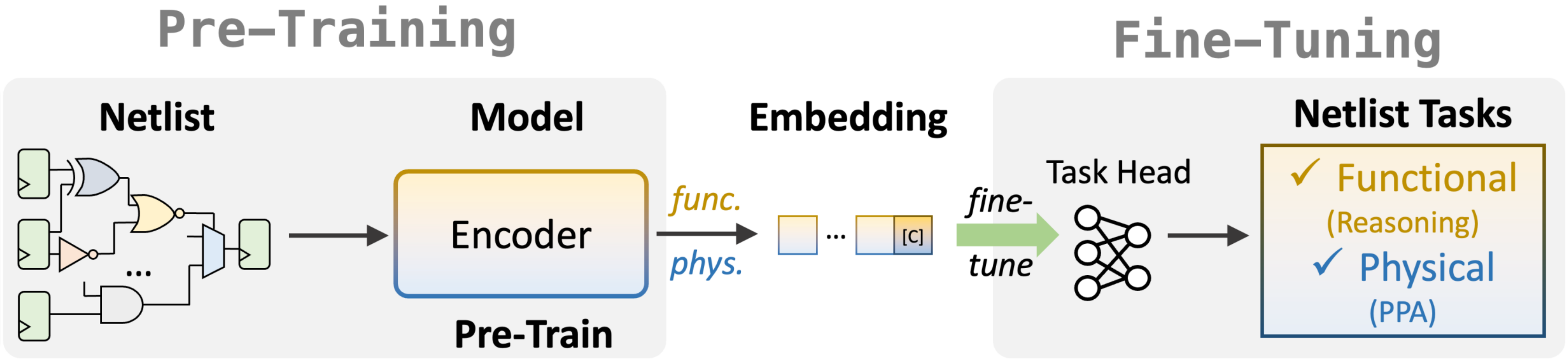
- Overview

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY

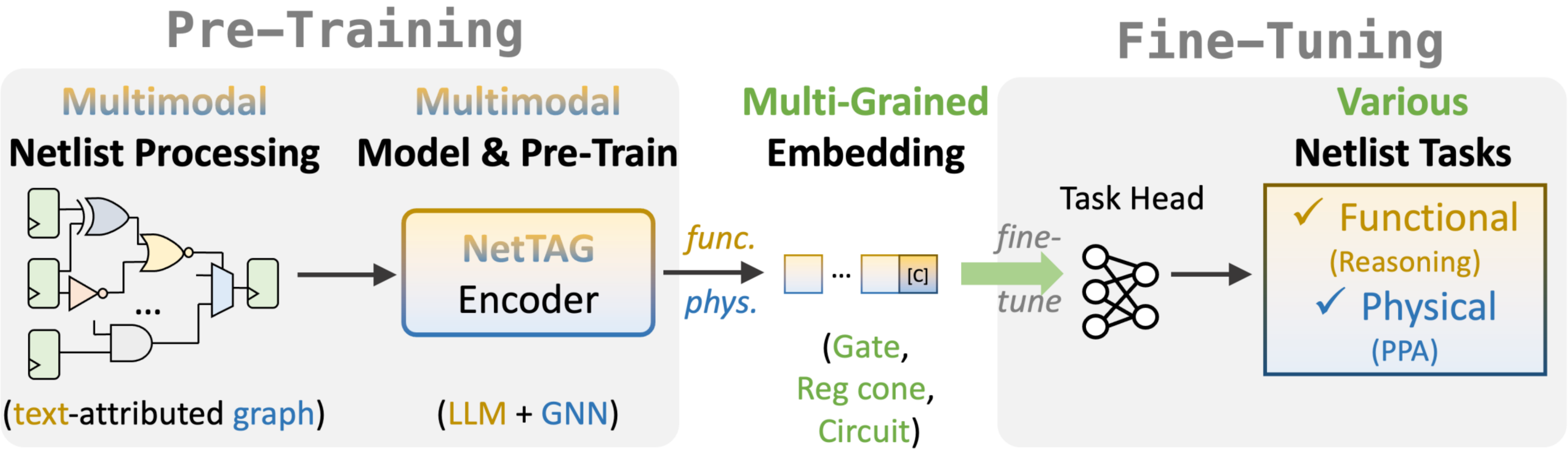
NetTAG: A General Netlist Foundation Model

- **Goal: develop a general netlist encoder**
  - **Pre-train:** netlist → embeddings w. functional and physical information
  - **Fine-tune:** support various EDA tasks w. the embedding



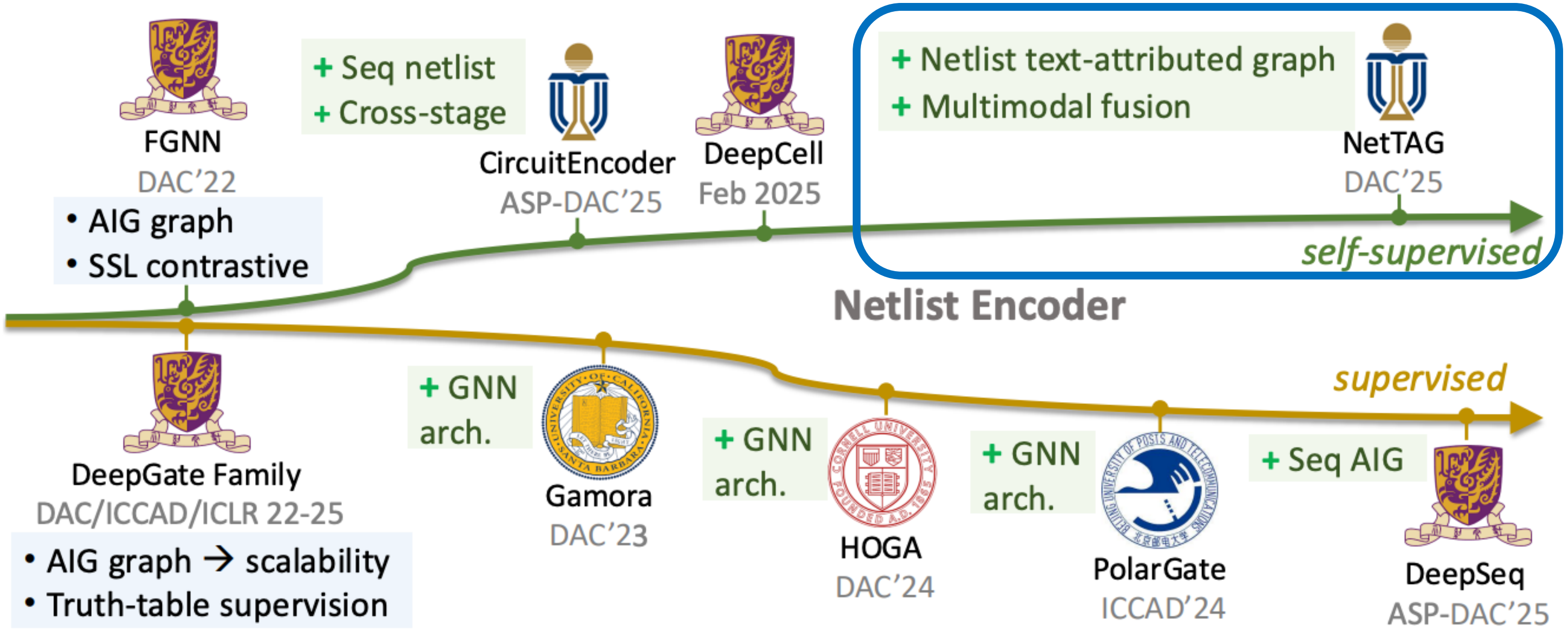
NetTAG: Multimodal Netlist Encoder via TAG

- **Key idea: fuse local gate text with global structure graph**
  - Multimodal preprocess: netlist → text-attributed graph
  - Multimodal 2-stage model: *gate text* (LLM) + *circuit graph* (GNN)
  - Multimodal pre-train: self-supervised & cross-stage-aware



NetTAG: Moving From AIG-Only to Any Netlist

- **Key advantages: first post-syn netlist encoder w. both func. & phys.**
  - Support diverse gate types *(any netlist gate)*
  - Support multiple circuit granularities *(gate & register & circuit)*
  - Support various netlist tasks *(func. & phys.)*



(a) Timeline for netlist encoders

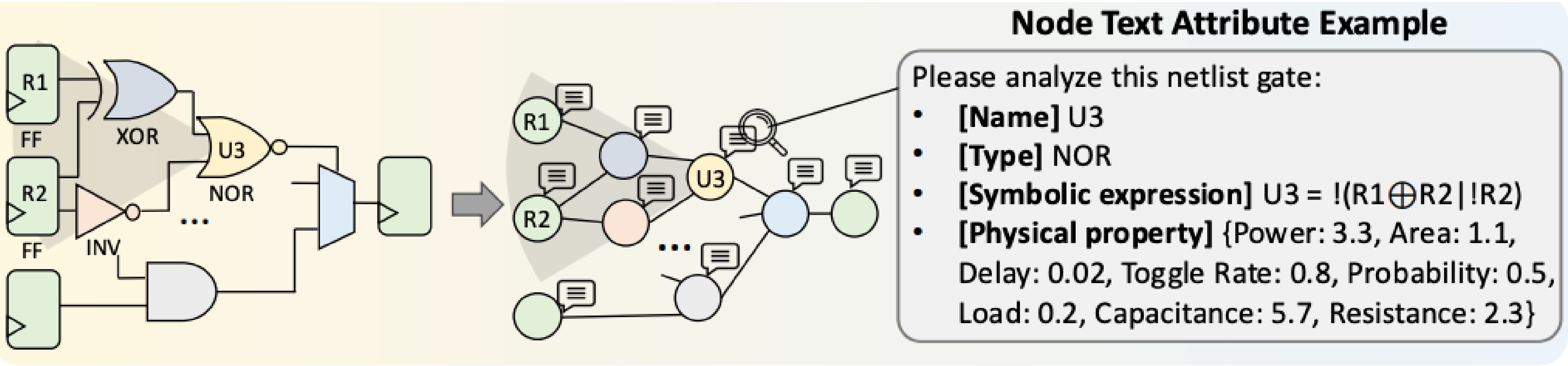
NetTAG: Multimodal Netlist Foundation Model

- Implementation Details

THE CHIPS TO SYSTEMS CONFERENCE

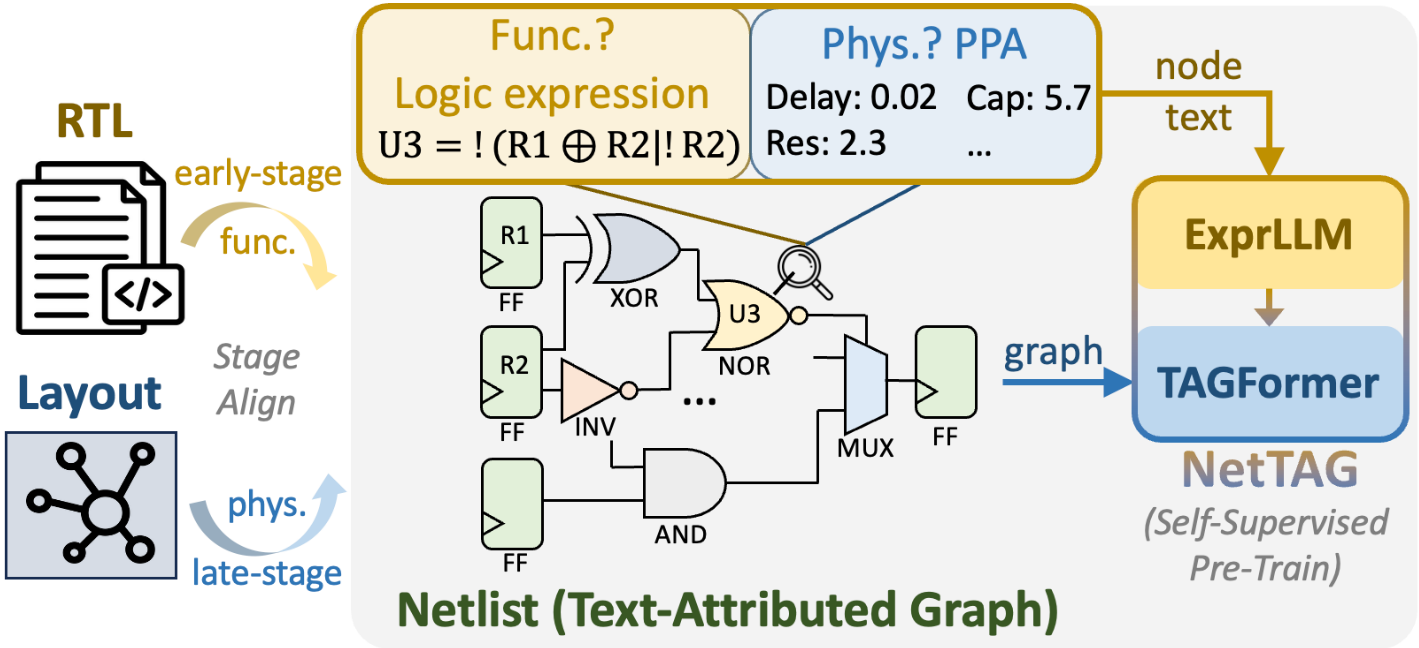
Preprocessing: Netlist as TAG

- **Text-attributed graph: 1. node (gate) text attribute + 2. graph connectivity**
  - **1. Gate → text:** local functional & physical info → ***semantics***
    - Functional: gate symbolic logic expression
      - *E.g., U3 = ! ((R1 ⊕ R2) | ! R2)*
    - Physical: gate physical characteristic vector
    - Advantages: various gate types & structure independence & LLM compatible
  - **2. Circuit → graph:** global connectivity → ***structure***



Model: NetTAG Multimodal Architecture

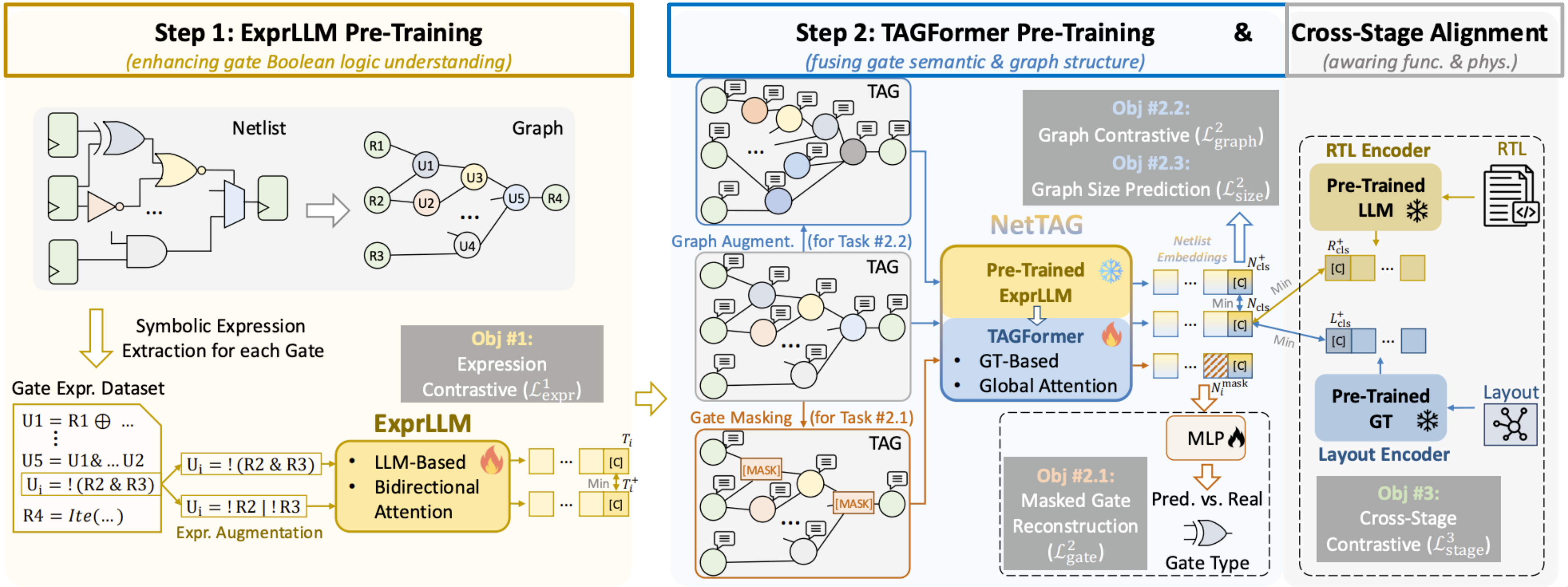
- **NetTAG two-stage encoding for netlist TAG**
  - **S1. ExprLLM** (LLM encoder): Gate-level text → node local initialization
  - **S2. TAGFormer** (GT): Circuit-level graph → graph global refinement
- **Auxiliary RTL and layout encoders → Cross-stage align**[3]



[3]Wenji Fang et al. A Self-Supervised, Pre-Trained, and Cross-Stage-Aligned Circuit Encoder Provides a Foundation for Various Design Tasks [ASP-DAC'25]

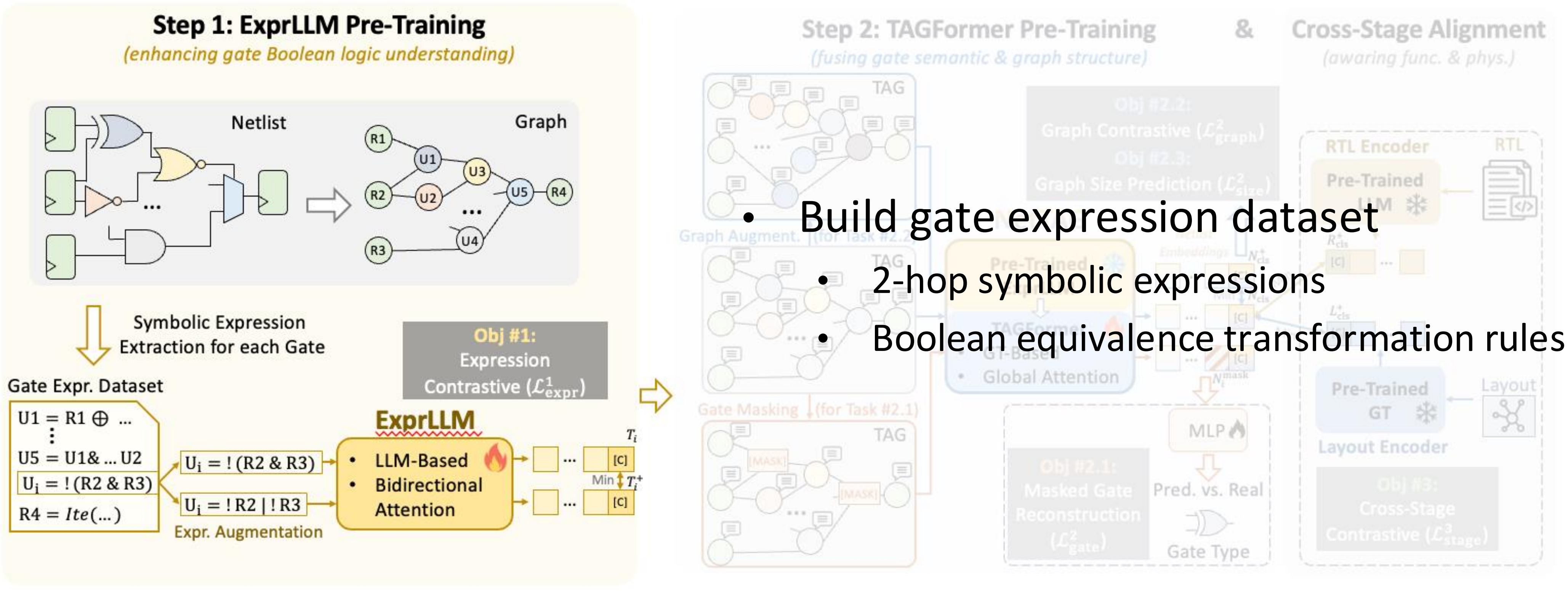
Pre-Train: Multimodal Self-Supervised (Overview)

- **Goal: enhance netlist *functional* and *physical* awareness**
  - Two-stage encoding → **Two-step pre-training + Cross-stage alignment**



Pre-Train: Multimodal Self-Supervised (Text)

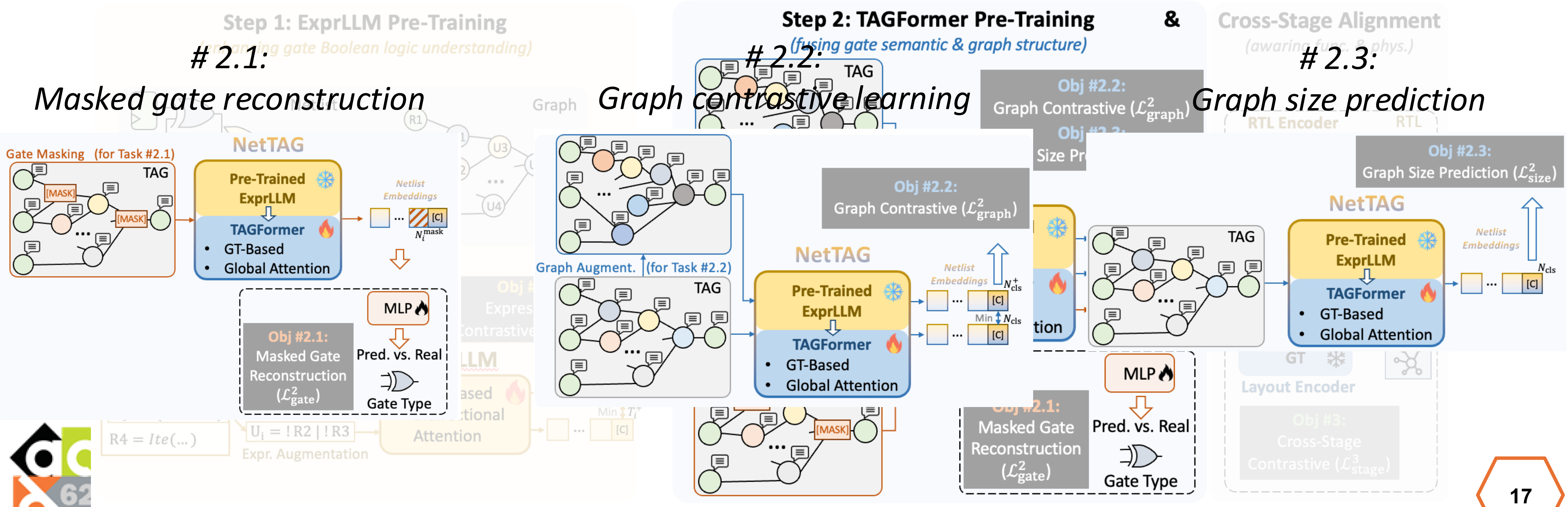
- **Step 1: Enhancing logic understanding in ExprLLM**
  - **Goal 1:** Differentiate gate expression text functionality
  - **Objective # 1:** Symbolic expression text contrastive learning



- Build gate expression dataset
  - 2-hop symbolic expressions
  - Boolean equivalence transformation rules

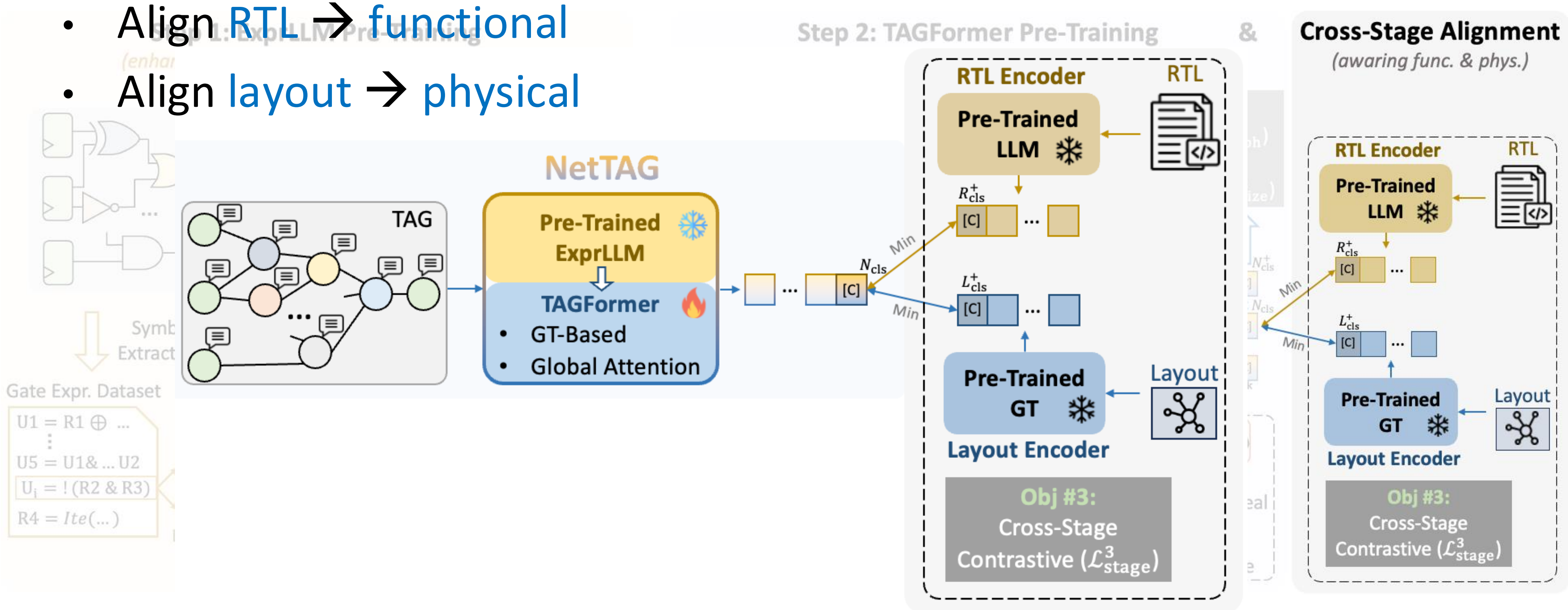
Pre-Train: Multimodal Self-Supervised (Graph)

- **Step 2: Fusion in TAGFormer**
  - **Goal 2:** Training within TAGFormer for semantic and structure fusion
  - **Objective # 2:** Node & graph-level self-supervised



Pre-Train: Multimodal Self-Supervised (Stage)

- **Step 3: Cross-stage alignment**
  - **Goal 3:** Training beyond NetTAG for cross-stage func. & phys. awareness
  - **Objective # 3:** Cross-stage contrastive alignment
    - Align RTL → functional
    - Align layout → physical



Experimental Results

THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Support Largely Different EDA Tasks – Functional

- **Functional tasks:** **reasoning** **earlier RTL function (classification)**
  - **Task 1:** Comb. gate function[4]
    - Node-level
  - **Task 2:** Seq. state/data register[5]
    - Subgraph-level

Design	GNN-RE [14] Acc. (%)	GNN-RE [14] Prec. (%)	GNN-RE [14] Recall (%)	GNN-RE [14] F1 (%)	NetTAG Acc. (%)	NetTAG Prec. (%)	NetTAG Recall (%)	NetTAG F1 (%)
1	79	82	79	74	97	97	97	97
2	96	96	96	96	100	100	100	100
3	94	94	94	94	100	100	100	100
4	78	83	78	78	100	100	100	100
5	91	92	91	90	99	99	99	99
6	74	78	74	68	94	94	94	93
7	80	80	80	80	84	87	84	81
8	89	90	89	87	95	96	95	96
9	65	77	65	67	100	100	100	100
Avg.	83	86	83	82	**97**	**97**	**97**	**96**

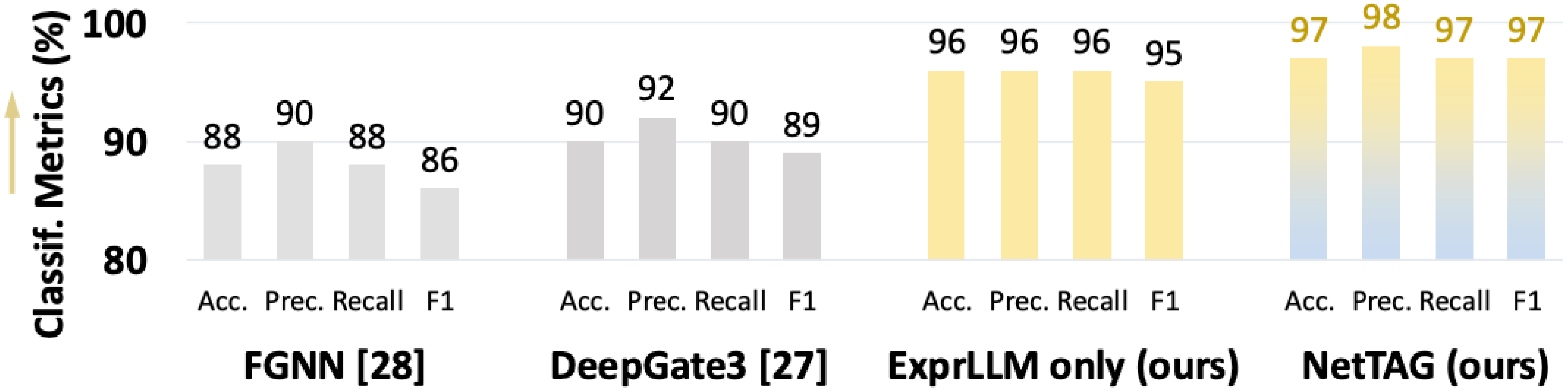
Design	Task 2 REIGNN [15] Sens. (%)	Task 2 REIGNN [15] Acc. (%)	Task 2 NetTAG Sens. (%)	Task 2 NetTAG Acc. (%)
itc1	50	72	100	98
itc2	100	92	100	100
chipyard1	30	65	80	79
chipyard2	30	65	90	86
vex1	50	74	82	74
vex2	32	60	86	82
opencores1	42	73	93	84
opencores2	37	80	92	82
Avg.	46	73	**90**	**86**

[4]Lilas Alrahis et al. GNN-RE Graph Neural Networks for Reverse Engineering of Gate-Level Netlists. [TCAD'22]

[5]Subhajit et al. ReIGNN: State Register Identification Using Graph Neural Networks for Circuit Reverse Engineering. [ICCAD'21]

Comparison with AIG Functional Encoders

- **Existing netlist encoders[6,7] only support AIG format**
  - AIG version of Task 1
  - ExprLLM-only can already achieve high accuracy



[6]Ziyi Wang et al. Functionality Matters in Netlist Representation Learning. [DAC'22]
[7]Zhengyuan Shi et al. DeepGate3: Towards Scalable Circuit Representation Learning. [ICCAD'22]

Support Largely Different EDA Tasks – Physical

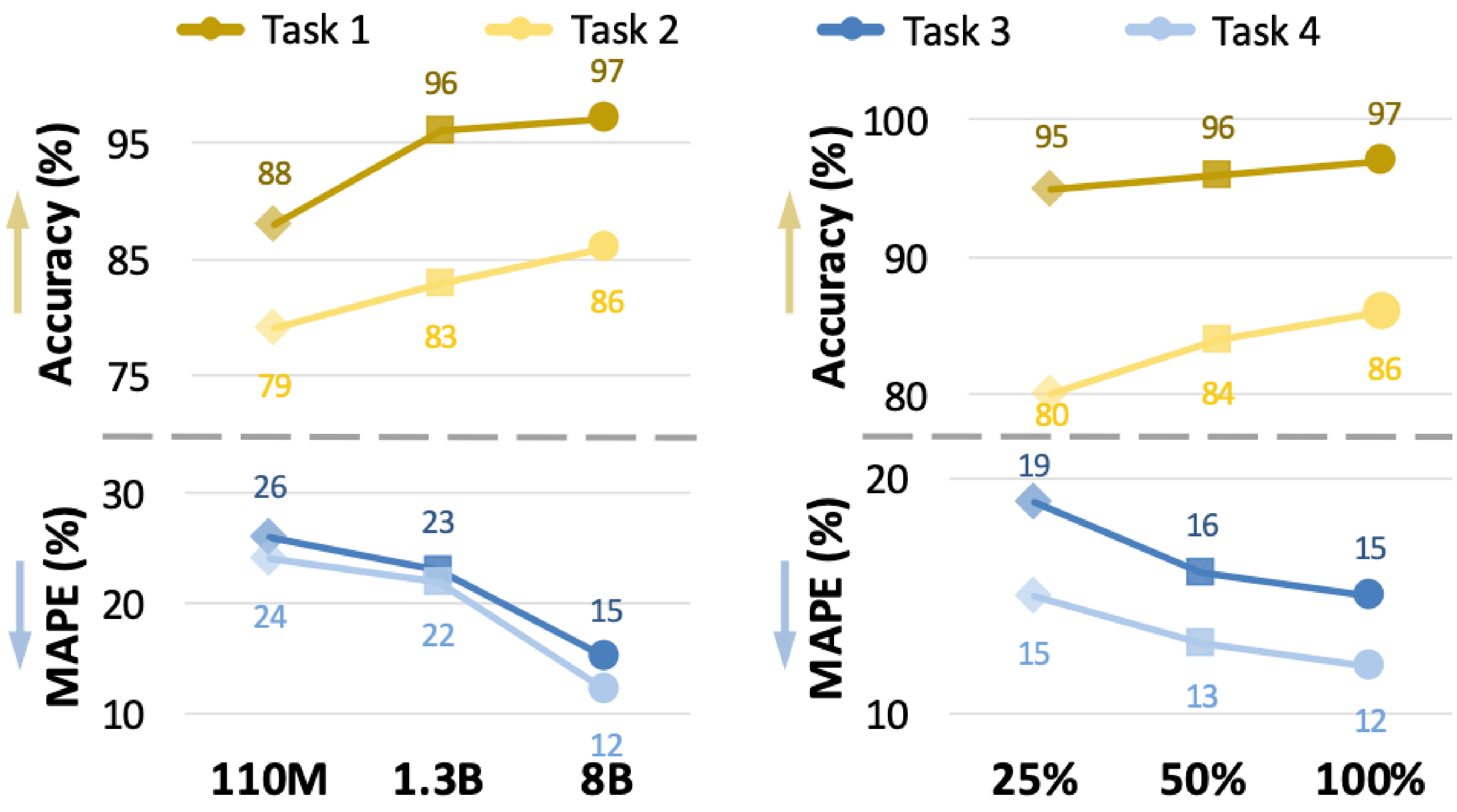
- **Physical tasks: predicting later layout PPA (regression)**
  - **Task 3:** Register slack
    - Subgraph-level
  - **Task 4:** Design power/area
    - Graph-level

Design	Task 3 GNN* R	GNN* MAPE (%)	NetTAG R	NetTAG MAPE (%)
itc1	0.89	13	0.94	9
itc2	0.91	10	0.93	9
chipyard1	0.72	17	0.7	20
chipyard2	0.86	12	0.95	9
vex1	0.94	11	0.93	12
vex2	0.97	18	0.97	9
opencores1	0.99	26	0.92	29
opencores2	0.93	30	0.98	26
Avg.	0.9	17	**0.92**	**15**

Target Metric†		EDA Tool R	EDA Tool MAPE (%)	GNN* R	GNN* MAPE (%)	NetTAG R	NetTAG MAPE (%)
Area	w/o opt	0.99	5	0.99	5	**0.99**	**4**
	w/ opt	0.95	34	0.95	18	**0.96**	**11**
Power	w/o opt	0.99	34	0.99	12	**0.99**	**8**
	w/ opt	0.73	38	0.76	19	**0.86**	**12**

Observation on Foundation Model Scaling Law

- **Scalability: performance scaling up w. model/data size**



(a) Increasing model size (b) Increasing data size

Conclusion & Future Work

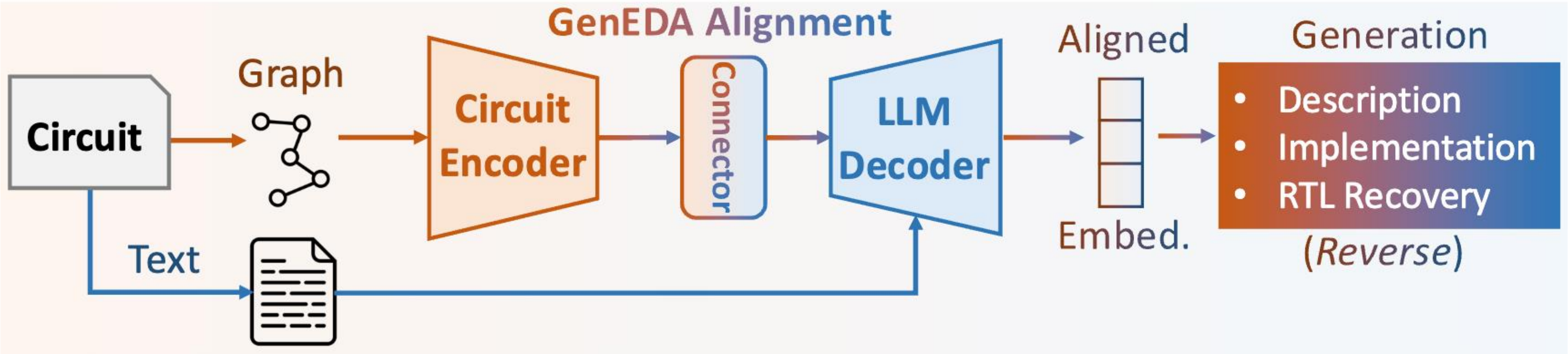
THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Conclusion and Future Work

- **NetTAG: first multimodal netlist encoder for any post-syn netlists**
- **Insight: graph-only → multimodal fusion paradigm via TAG**
- **Future work[8]**
  - Bridging the gap between Circuit Encoder and LLM Decoder
    - Integrate NetTAG into LLMs for generative netlist function reasoning

Paper *Code*



[8]Wenji Fang et al. GenEDA: Unleashing Generative Reasoning on Netlist via Multimodal Encoder-Decoder Aligned Foundation Model.

AI

Security

Systems

Thank You!
Questions?

EDA

Design

THE CHIPS TO SYSTEMS CONFERENCE

62

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm

Backup: Experimental Setup

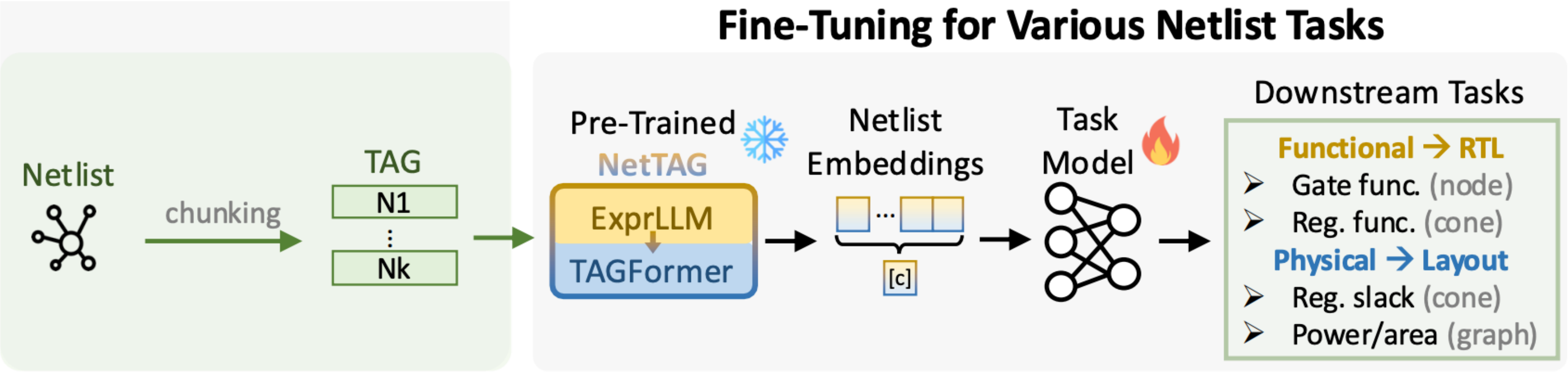
- **Dataset and preparation**
  - Dataset statistics: 626k expression & 200k netlist subcircuits & 10k aligned RTL/layout
  - Symbolic logic expression manipulation: PySMT
- **Model Implementation**
  - ExprLLM: LLM2Vec, an LLM encoder w. bidirectional attention (max 8k input)
  - TAGFormer: SGFormer, a graph transformer
  - RTL encoder: NV-Embed (max 32k input)
  - Layout encoder: another SGFormer

TABLE II: Statistics of circuit expression and netlist dataset.

Source	Gate Expression		Circuit Netlist	
	# Data	# Tokens (Avg.)	# Data	# Nodes (Avg.)
ITC99	47k	6,960	4k	1,025
OpenCores	76k	212	55k	173
Chipyard	109k	9,849	20k	2,813
VexRiscv	81k	5,289	21k	901
Total	**313k**	**5,810**	**100k**	**855**

Backup: Fine-Tuning for Various Netlist Tasks

- **Fine-tune embeddings with lightweight task models**
  - Reasoning earlier RTL function
  - Predicting later layout PPA



Backup: Ablation Study

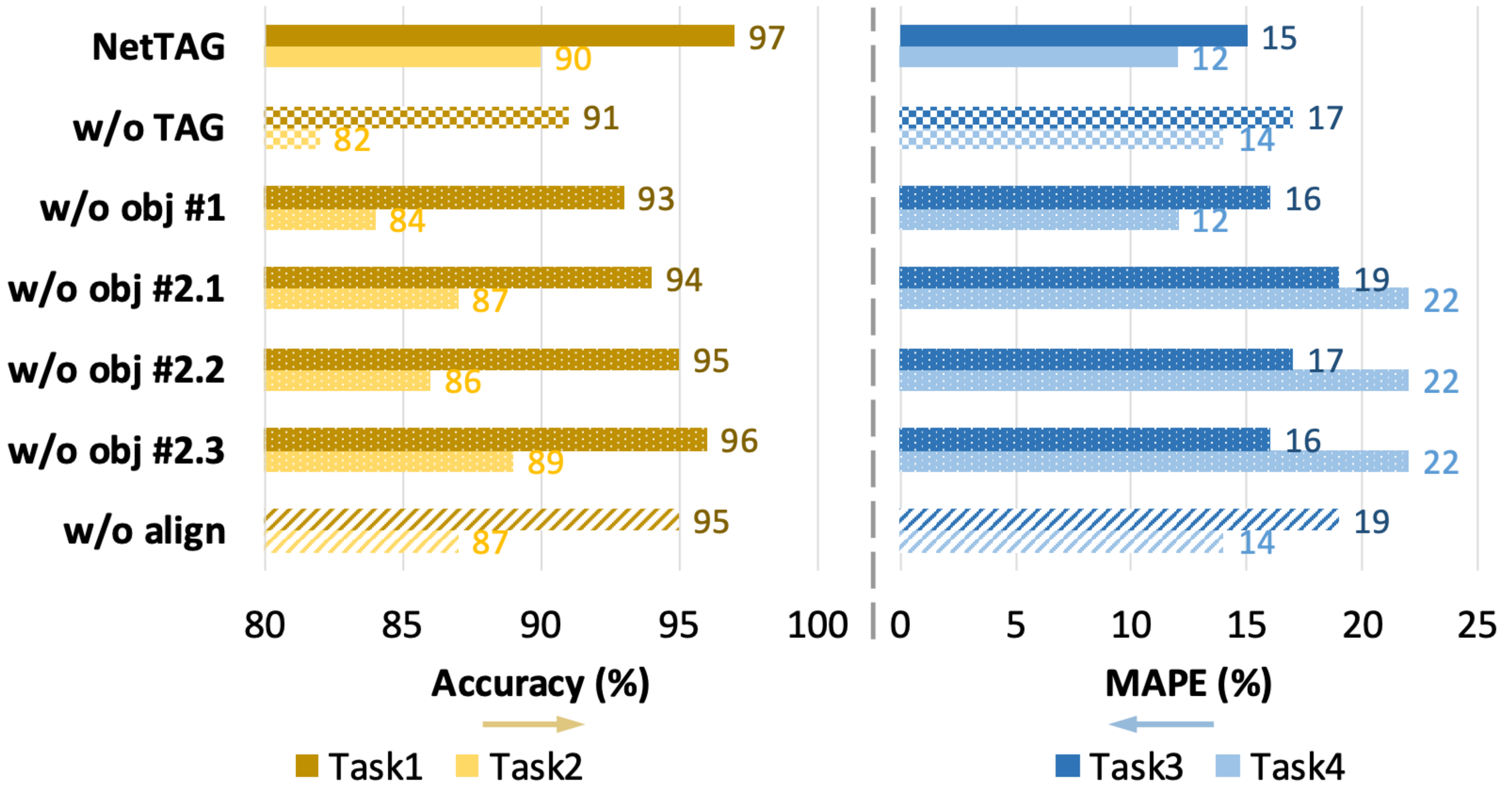


Fig. 6: Ablation study.

Backup: Runtime Analysis

- **10x speedup over physical design process**
- **Key runtime factor**
  - Symbolic expression extraction
    - Improve: gate-level parallelism
  - ExprLLM inference
    - Improve: more GPU resources

TABLE VI: Runtime (minutes) Comparision.

Source	EDA Tool	Ours (Avg.)			
	P&R (Avg.)	Pre*	Infer		Total
			ExprLLM	TAGFormer	
ITC99	164	2	5	0	7
OpenCores	288	18	12	1	31
Chipyard	251	15	10	1	26
VexRiscv	207	8	5	2	15
GNNRE	/	4	2	0	6

* Preprocessing (chunking into cones and converting netlist into TAG).