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

ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE

A Self-Supervised, Pre-Trained, and Cross-Stage-Aligned Circuit Encoder Provides a Foundation for Various Design Tasks

Wenji Fang[1], Shang Liu[1], Hongce Zhang[1,2], Zhiyao Xie[1]
wfang838@connect.ust.hk

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

Outline

- Background
- CircuitEncoder Framework
- Experimental Results
- Conclusion & Future Work

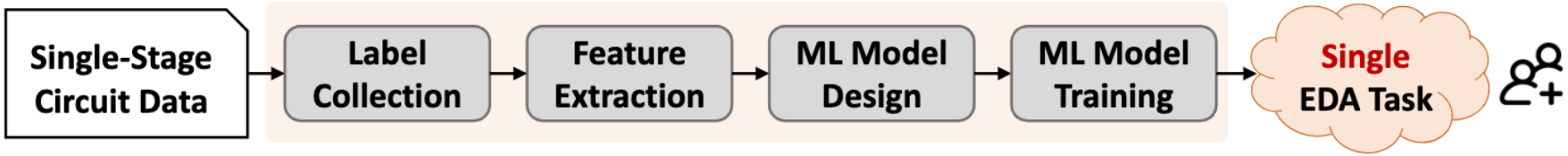
Background

Background: AI for EDA

- **Remarkable achievements**
  - **Design quality evaluation**
    - Power, timing, area, routability, etc.
  - **Functional reasoning**
    - Arithmetic word-level abstraction, SAT, etc.
  - **Optimization**
    - Design space exploration, etc.
  - **Generation**
    - RTL code, verification, etc.
  - **…**

Background: AI for EDA

- **Most existing predictive solutions are task-specific**
  - Supervised learning
  - **Tedious** and **time-consuming**
  - Hard to **generalize** to other tasks



Background: Foundation Models

- **AI foundation models**
  - **Pretrain-finetune paradigm**
    - Pre-training on large amounts of unlabeled data (**self-supervised**)
    - Fine-tuning based on task-specific labels (**supervised**)
  - **Applications**
    - **Natural language processing**: GPT, BERT, Llama, etc.
    - **Computer vision**: DALLE, stable-diffusion

CircuitEncoder Framework

Motivation: Towards Circuit Foundation Models

- **Large circuit model**

SCIENCE CHINA
Information Sciences

• POSITION PAPER •
Special Topic: AI Chips and Systems for Large Language Models

October 2024, Vol. 67, Iss. 10, 200402:1–200402:42
https://doi.org/10.1007/s11432-024-4155-7

Large circuit models: opportunities and challenges†

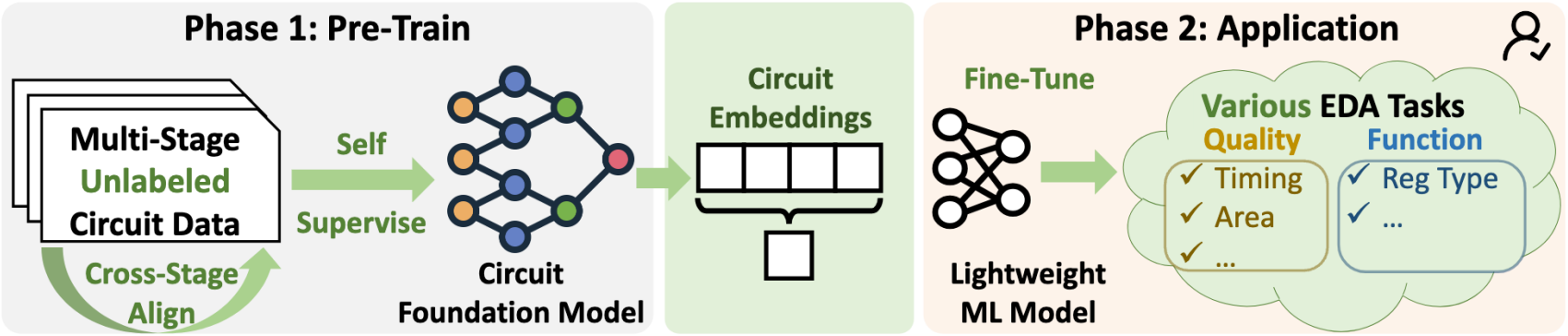
Lei CHEN[5], Yiqi CHEN[7], Zhufei CHU[6], Wenji FANG[3], Tsung-Yi HO[1], Ru HUANG[7,11], Yu HUANG[4], Sadaf KHAN[1], Min LI[5], Xingquan LI[9], Yu LI[1], Yun LIANG[7], Jinwei LIU[1], Yi LIU[1], Yibo LIN[7], Guojie LUO[8*], Hongyang PAN[2], Zhengyuan SHI[1], Guangyu SUN[7], Dimitrios TSARAS[5], Runsheng WANG[7], Ziyi WANG[1], Xinming WEI[8], Zhiyao XIE[3], Qiang XU[1*], Chenhao XUE[7], Junchi YAN[10], Jun YANG[11], Bei YU[1], Mingxuan YUAN[5*], Evangeline F.Y. YOUNG[1], Xuan ZENG[2], Haoyi ZHANG[7], Zuodong ZHANG[7], Yuxiang ZHAO[7], Hui-Ling ZHEN[5], Ziyang ZHENG[1], Binwu ZHU[1], Keren ZHU[1] & Sunan ZOU[8]

Motivation: Towards Circuit Foundation Model

- **Our targeted circuit foundation model**
  - **Capture unique circuit intrinsic property**
    - ***Cross-stage:*** RTL (functional) → netlist (Physical)
    - ***Equivalent transformation:*** semantic & structure
    - …
  - **Support various types of tasks**
    - ***Functionality:*** reasoning, verification, etc.
    - ***Design quality:*** performance, power, area, etc.
    - …

Key Idea: First RTL-Netlist Cross-Stage Alignment

- **General circuit foundation model solution**
  - **Two-phase paradigm**
    - Self-supervised pre-trainining
    - Supervised fine-funing



Comparison with Existing Solution

- **Circuit representation learning**
  - **Goal:** to learn a **general circuit embedding** for **various tasks**
  - **Explorations**
    - **Supervised**: HOGA, Gamora, etc.
    - **Pre-trained**: DeepGate Family, FGNN, SNS v2, etc.

Table 1: Existing two-phase circuit representation learning techniques for ASIC design.

Method	Downstream Tasks			Pre-Training		Design Stage		Support Seq. Circuit	Open-Source Model
	Multi-Type	Design Quality	Function	Self-Supervised	Train Task	Cross-Stage	Target Stage		
Design2Vec [25]			✓		Cover Point		RTL	✓	
SNS v2 [36]		✓		✓	Contrastive		RTL	✓	
FGNN [28]			✓	✓	Contrastive		Netlist		
DeepGate [17]			✓		Probability		Netlist		
DeepGate2 [23]			✓		Truth Table		Netlist		✓
DeepSeq [16]		✓⋆			Probability		Netlist	✓	
CircuitEncoder (Ours)	✓	✓	✓	✓	**Multi-Stage Contrastive**	✓	**RTL Netlist**	✓	✓

⋆ DeepSeq predicts netlist gate toggle rate at the node level to estimate power consumption, rather than directly modeling power.

Comparison with Existing Solution

- **Circuit representation learning**
  - **Limitations:** still do not provide perfectly general circuit representation
    - Mainly support **one type of task** (phys. PPA or func.)
    - Only target **single stage** (RTL or netlist)

Table 1: Existing two-phase circuit representation learning techniques for ASIC design.

Method	Downstream Tasks			Pre-Training		Design Stage		Support Seq. Circuit	Open-Source Model
	Multi-Type	Design Quality	Function	Self-Supervised	Train Task	Cross-Stage	Target Stage		
Design2Vec [25]			✓		Cover Point		RTL	✓	
SNS v2 [36]		✓		✓	Contrastive		RTL	✓	
FGNN [28]			✓	✓	Contrastive		Netlist		
DeepGate [17]			✓		Probability		Netlist		
DeepGate2 [23]			✓		Truth Table		Netlist		✓
DeepSeq [16]		✓*			Probability		Netlist	✓	
CircuitEncoder (Ours)	✓	✓	✓	✓	**Multi-Stage Contrastive**	✓	**RTL Netlist**	✓	✓

* DeepSeq predicts netlist gate toggle rate at the node level to estimate power consumption, rather than directly modeling power.

Comparison with Existing Solution

- **Our CircuitEncoder**
  - **Self-supervised pre-trained**: circuit graph function contrastive
  - **Cross-stage aligned**: RTL (*func.*)—netlist (*phys.*) alignment
  - **Support various design tasks**: PPA + functionality

Table 1: Existing two-phase circuit representation learning techniques for ASIC design.

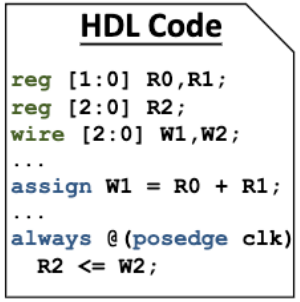
Method	Downstream Tasks			Pre-Training		Design Stage		Support Seq. Circuit	Open-Source Model
	Multi-Type	Design Quality	Function	Self-Supervised	Train Task	Cross-Stage	Target Stage		
Design2Vec [25]			✓		Cover Point		RTL	✓	
SNS v2 [36]		✓		✓	Contrastive		RTL	✓	
FGNN [28]			✓	✓	Contrastive		Netlist		
DeepGate [17]			✓		Probability		Netlist		
DeepGate2 [23]			✓		Truth Table		Netlist		✓
DeepSeq [16]		✓⋆			Probability		Netlist	✓	
CircuitEncoder (Ours)	✓	✓	✓	✓	**Multi-Stage Contrastive**	✓	**RTL Netlist**	✓	✓

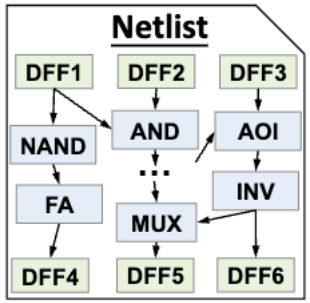
⋆ DeepSeq predicts netlist gate toggle rate at the node level to estimate power consumption, rather than directly modeling power.

Preprocessing: Circuit Design Stages

- **RTL**
  - **Earlier** design stage
  - Higher abstraction level
  - More **semantic** content
  - **Task**
    - Predicting later **netlist PPA**

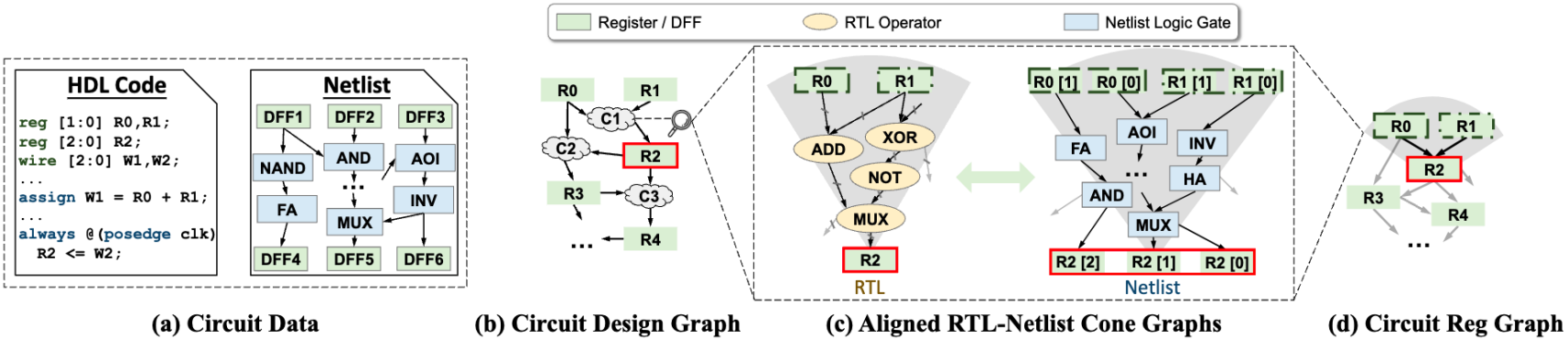
- **Netlist**
  - **Later** design stage
  - Lower abstraction level
  - More **implementation** details
  - **Task**
    - Reasoning earlier **RTL function**





Preprocessing: Circuit Data Alignment

- **Circuit-to-graph transformation**

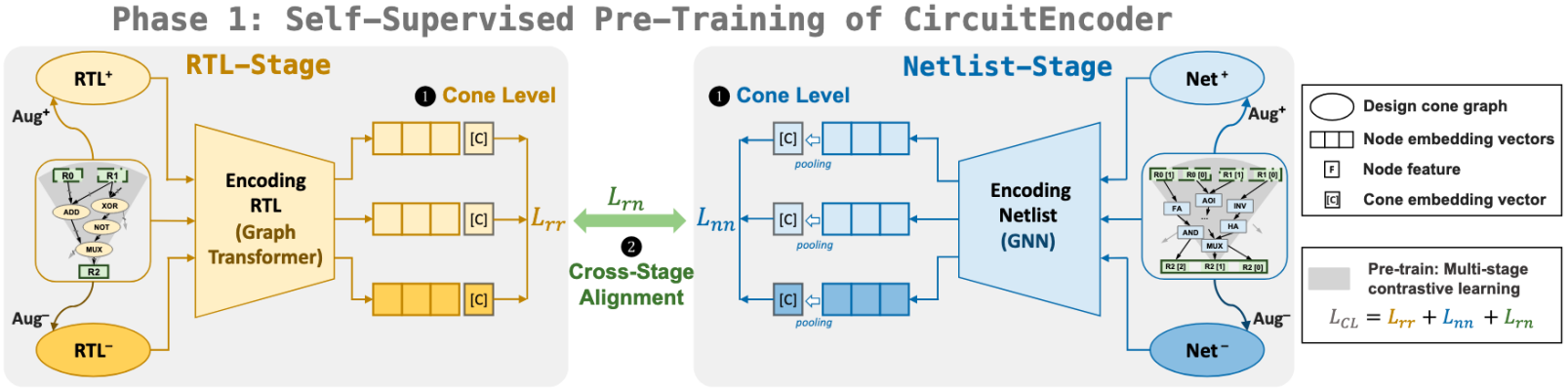


(a) Circuit Data (b) Circuit Design Graph (c) Aligned RTL-Netlist Cone Graphs (d) Circuit Reg Graph

- **RTL-Netlist *data* alignment via backtrace register cone**
  - **Advantages**
    - RTL-netlist cones are **strictly aligned** & **functionally equivalent**
    - Capture the entire **state transition** of each register
    - Intermediate granularity → **better scalability**

Encoding: Graph Learning Model for Circuits

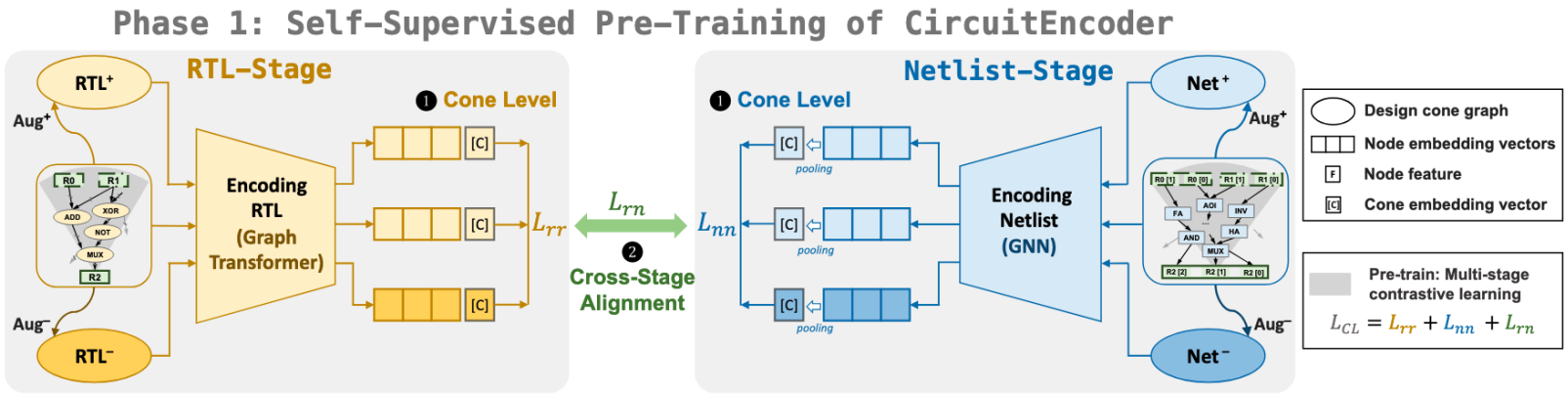
- **RTL graph**
  - **Graph transformer**
  - **Global positional encoding**
- **Netlist graph**
  - Graph neural network
  - Neighbor **aggregation**
- Node-level embeddings → Cone graph-level embeddings



CircuitEncoder Phase 1: Pre-Training

- **Self-supervised pre-training: intrinsic circuit property**
  - ❶ ***Intra-stage*** **contrastive learning within each stage**
    - **Minimizing** embed. distance between **positive pairs** (equiv. transform.)
    - **Maximizing** embed. distance among **negative pairs** (func. diff.)

$$L_{rr} = max(\left\|E_{\mathrm{R}} - E_{\mathrm{R}^+}\right\|_2 - \|E_{\mathrm{R}} - E_{\mathrm{R}^-}\|_2 + m_{rr}, 0),$$
$$L_{nn} = max(\left\|E_{\mathrm{N}} - E_{\mathrm{N}^+}\right\|_2 - \|E_{\mathrm{N}} - E_{\mathrm{N}^-}\|_2 + m_{nn}, 0),$$



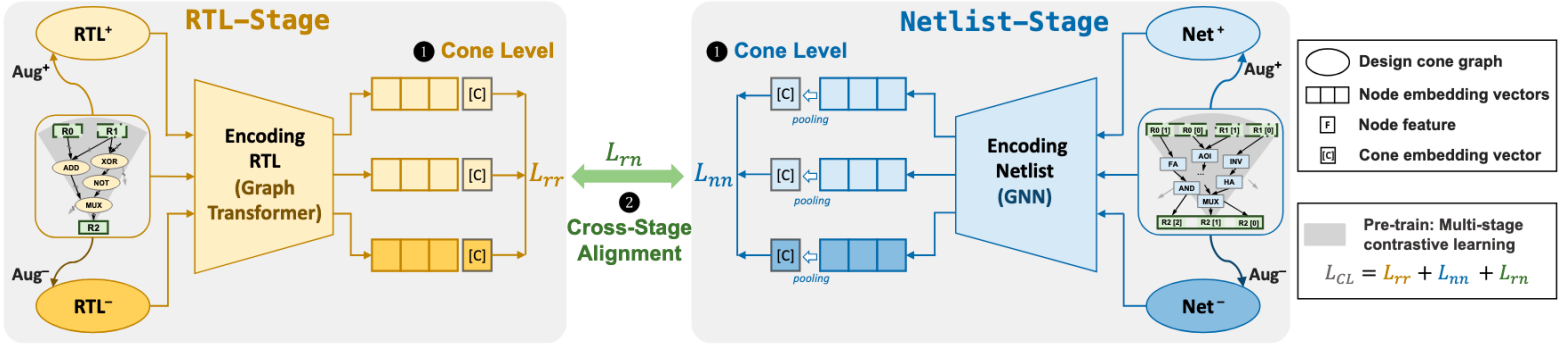
CircuitEncoder Phase 1: Pre-Training

- **Self-supervised pre-training: intrinsic circuit property**
  - ❷ ***Inter-stage*** **contrastive learning across stages**
    - **Cross-stage alignment** between RTL and netlist embed.

$$L_{rn} = max(\left\|E_{\mathrm{R}} - E_{\mathrm{N}^+}\right\|_2 - \|E_{\mathrm{R}} - E_{\mathrm{N}^-}\|_2 + m_{rn}, 0)$$
$$+ max(\left\|E_{\mathrm{N}} - E_{\mathrm{R}^+}\right\|_2 - \|E_{\mathrm{N}} - E_{\mathrm{R}^-}\|_2 + m_{rn}, 0).$$

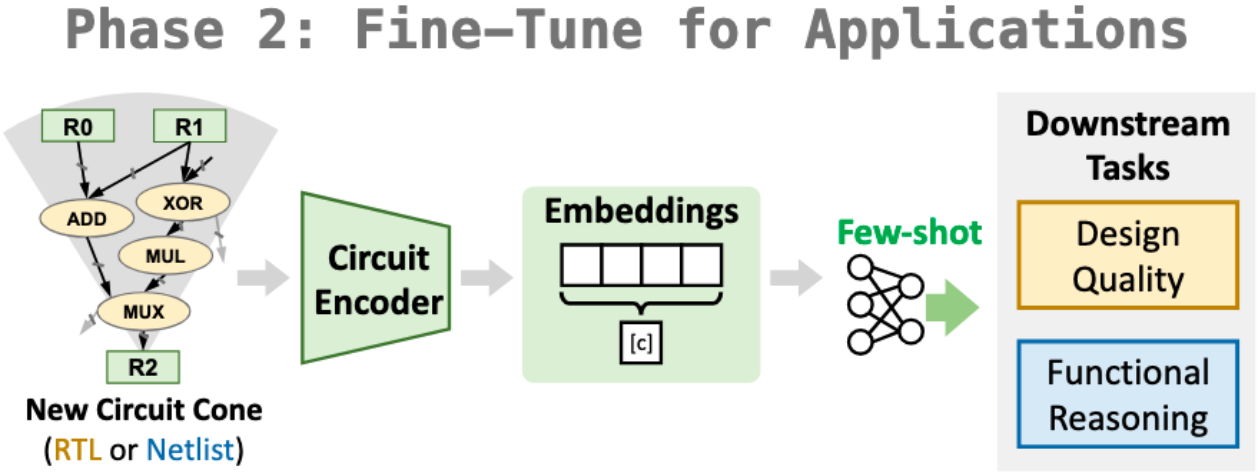
$$L_{CL} = \alpha_{rr} L_{rr} + \alpha_{nn} L_{nn} + \alpha_{rn} L_{rn},$$

Phase 1: Self-Supervised Pre-Training of CircuitEncoder



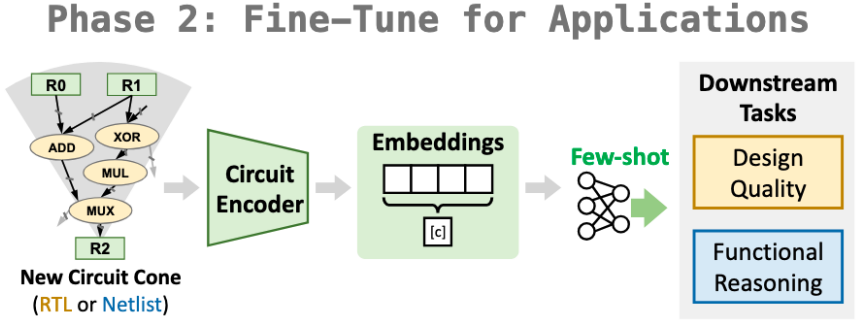
CircuitEncoder Phase 2: Fine-Tuning for Tasks

- **Supervised fine-tuning**
  - **Lightweight task models**: MLP, tree-based, etc.



CircuitEncoder Phase 2: Fine-Tuning for Tasks

- **Downstream tasks**
  - **Register cone-level:**
    - **[PPA] Timing slack** prediction – ***RTL***
    - **[Func] Register function** (control/data) identification – ***netlist*** *stage*
  - **Design-level:**
    - **[PPA] Overall PPA** prediction – ***RTL***
      - WNS
      - TNS
      - Area



Experimental Results

Circuit Design Statistics

- **41 open-source designs**
- **7,166 RTL and netlist cone pairs**
- **Data augmentation → 42,996 graphs in total**

Table 2: Benchmark design information.

Source Benchmarks	# Design	Design Size {Min, Avg, Max}		Original HDL Type
		#K Gates	# Cones	
ITC [6]	7	{7, 15, 22}	{12, 21, 31}	VHDL
OpenCores [1]	5	{2, 40, 59}	{12, 96, 173}	Verilog
Vex [26]	17	{8, 208, 591}	{39, 168, 694}	SpinalHDL
Chipyard [2]	12	{11, 49, 194}	{28, 461, 2730}	Chisel

Experimental Setup

- **Industrial-standard VLSI design flow**
  - RTL designs are synthesized using **DC / NanGate 45nm**
  - Design PPA metrics are obtained from **PT**
- **Circuit augmentation**
  - **Yosys / ABC** for functionally equivalent transformation
- **Graph model**
  - RTL: **Graphormer (graph transformer)**
  - Netlist: **GraphSage (GNN)**

Experimental Setup

- **Model training**

Table 4: ML model and training hyperparameters.

Training Phase	Pre-Training		Fine-Tuning		
ML Model	Graphormer (RTL)	GraphSage (Netlist)	MLP	GCN	XGBoost
# Layers	7	3	2	2	
Hidden Dim	256	256	128	128	
Activation	GELU	ReLU	ReLU	ReLU	100
Batch Size	128		32		estimator,
Optimizer	AdamW		Adam		20
LR	0.001		0.001		max depth
Dataset Size	33162		3278		
# Epochs	75		1000		
Training Time	20h		0.05h		

Task Evaluation and Supervised Baseline Methods

- **Design quality evaluation – regression metrics**
  - Register slack prediction at cone level
    - RTL-Timer [DAC'24]
  - RTL-stage overall quality evaluation at circuit level
    - MasterRTL [ICCAD'23]
    - SNS v2 [MICRO'23]
- **Functional reasoning – classification metrics**
  - Netlist-stage state register classification at cone level
    - ReIGNN [ICCAD'21]

Results: Comparison with SOTA Solutions

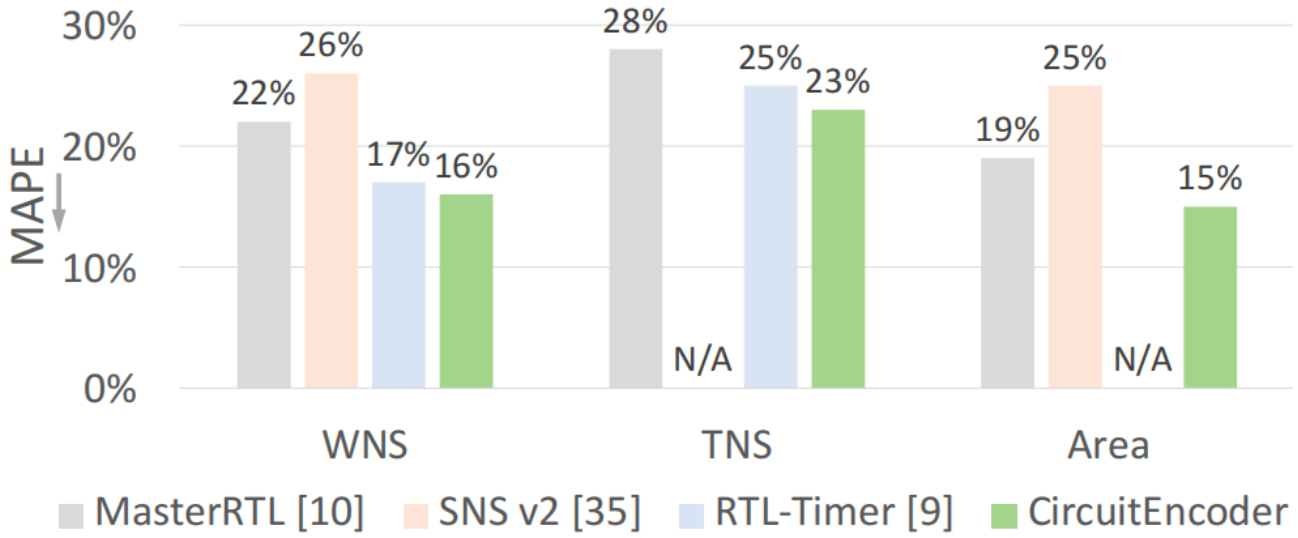
- **Outperforming each task-specific SOTA solution**
  - Cone-level tasks
  - **Few-shot** learning during fine-tuning
    - **50%** data for CircuitEncoder > **100%** data for supervised baselines

Table 3: Accuracy comparison for the cone-level tasks for RTL and netlist designs.

Method	RTL-Stage (Register Slack Prediction)										Netlist-Stage (State Register Identification)									
	RTL-Timer (supervised learning)						CircuitEncoder (pre-train + few-shot)				ReIGNN* (supervised learning)						CircuitEncoder (pre-train + few-shot)			
% of train	13%		50%		100%		13%		50%		13%		50%		100%		13%		50%	
Test Designs	R	MAPE	R	MAPE	R	MAPE	R	MAPE	R	MAPE	Sens.	Acc.	Sens.	Acc.	Sens.	Acc.	Sens.	Acc.	Sens.	Acc.
ITC1	0.48	22%	0.77	20%	0.82	18%	0.91	21%	0.96	9%	0%	72%	50%	72%	50%	72%	100%	98%	100%	98%
ITC2	0.43	26%	0.83	12%	0.88	10%	0.92	19%	0.96	9%	0%	92%	100%	92%	100%	92%	100%	100%	100%	100%
Chipyard1	0.57	30%	0.89	12%	0.92	18%	0.81	15%	0.83	18%	0%	50%	0%	50%	30%	65%	78%	77%	79%	79%
Chipyard2	0.56	31%	0.85	19%	0.88	12%	0.84	12%	0.85	13%	0%	50%	0%	50%	30%	65%	84%	78%	89%	85%
Vex1	0.28	27%	0.65	15%	0.87	24%	0.69	25%	0.88	26%	0%	50%	0%	50%	50%	74%	76%	79%	82%	72%
Vex2	0.73	29%	0.93	17%	0.86	16%	0.85	13%	0.87	13%	15%	57%	21%	57%	32%	60%	73%	76%	79%	78%
Vex3	0.27	36%	0.56	40%	0.84	16%	0.81	14%	0.89	12%	16%	48%	0%	48%	50%	72%	81%	82%	85%	84%
Vex4	0.12	40%	0.76	18%	0.87	12%	0.83	16%	0.86	14%	30%	63%	33%	63%	33%	63%	88%	79%	90%	81%
Avg.	**0.43**	**30%**	**0.78**	**19%**	**0.87**	**16%**	**0.83**	**17%**	**0.89**	**14%**	**8%**	**60%**	**26%**	**60%**	**47%**	**70%**	**85%**	**84%**	**88%**	**85%**

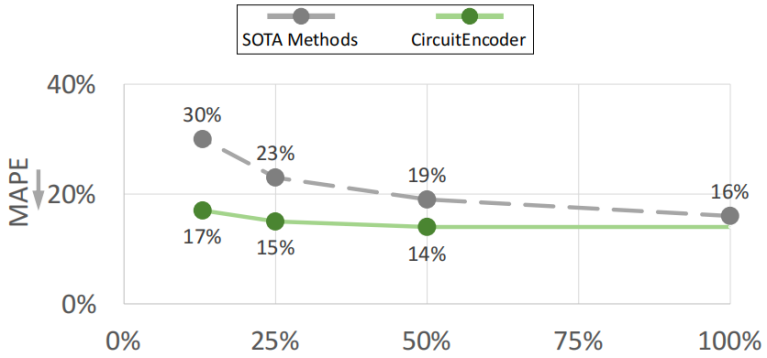
Results: Comparison with SOTA Solutions

- **Outperforming each task-specific SOTA solution**
  - Circuit-level tasks

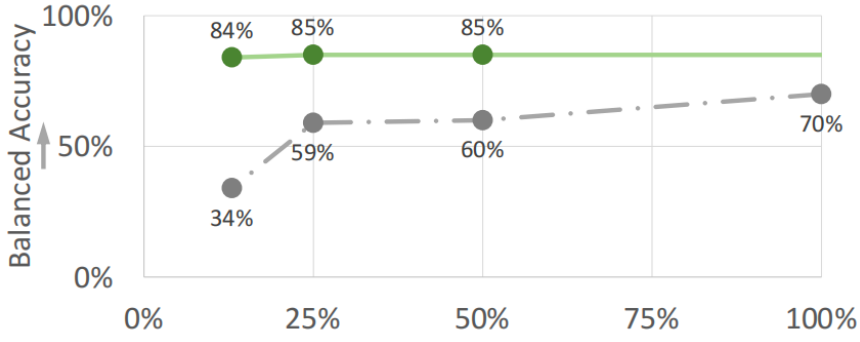


Results: Comparison with SOTA Solutions

- **Fine-tuning data size scaling**
  - 100% → 50% → 25% → 12%
  - **Pre-trained** CircuitEncoder remains **stable**



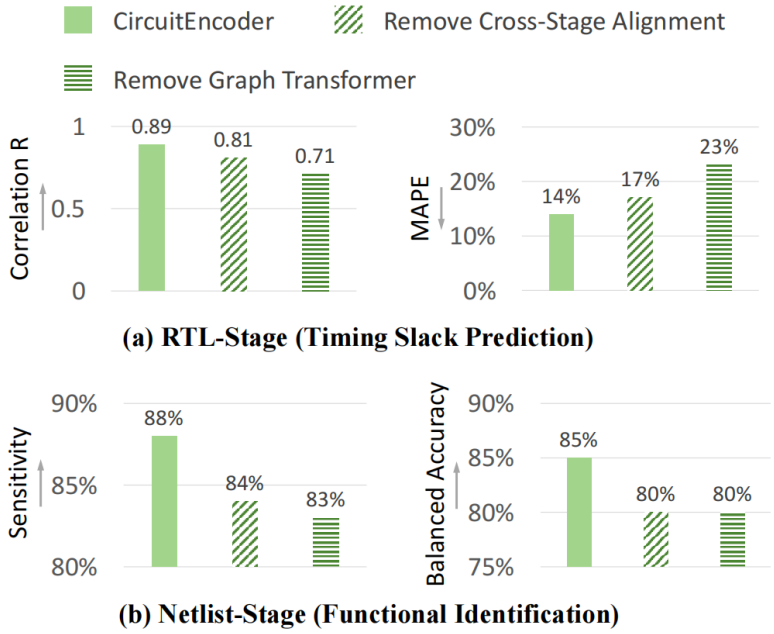
(a) RTL-Stage Timing Slack Prediction



(b) Netlist-Stage Functional Identification

Ablation Study

- Impact of cross-stage alignment
- Impact of graph transformer



Conclusion & Future Work

Conclusion

- **CircuitEncoder**
  - **Self-supervised & pre-trained**
    - Graph function **contrastive** learning
  - **Cross-stage alignment**
    - RTL ***function*** – netlist ***physics***
  - **Support various tasks**
    - **Design quality:** slack, WNS, TNS, area prediction
    - **Functional reasoning:** state register identification

Future Work

- **Advancing circuit foundation model**
  - **Contrastive learning → Circuit-specific self-supervised learning**
  - **Graph modality → Multimodality for each design stage**
    - **RTL:** AST/control-data flow graph, Verilog code, specification text
    - **Netlist:** connectivity graph, annotated node text
    - **Layout:** image, netlist graph
  - **Existing encoders and decoders work separately**
    - Encoder (graph) – predictive task
    - Decoder (text, LLM) – generative task
    - **Unified encoder-decoder?**

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

Thank You!
Questions?