61 THE CHIPS TO SYSTEMS CONFERENCE

SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

61 THE CHIPS TO SYSTEMS CONFERENCE
SHAPING THE NEXT GENERATION OF ELECTRONICS
JUNE 23-27, 2024
MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Annotating Slack Directly on Your Verilog: Fine-Grained RTL Timing Evaluation for Early Optimization

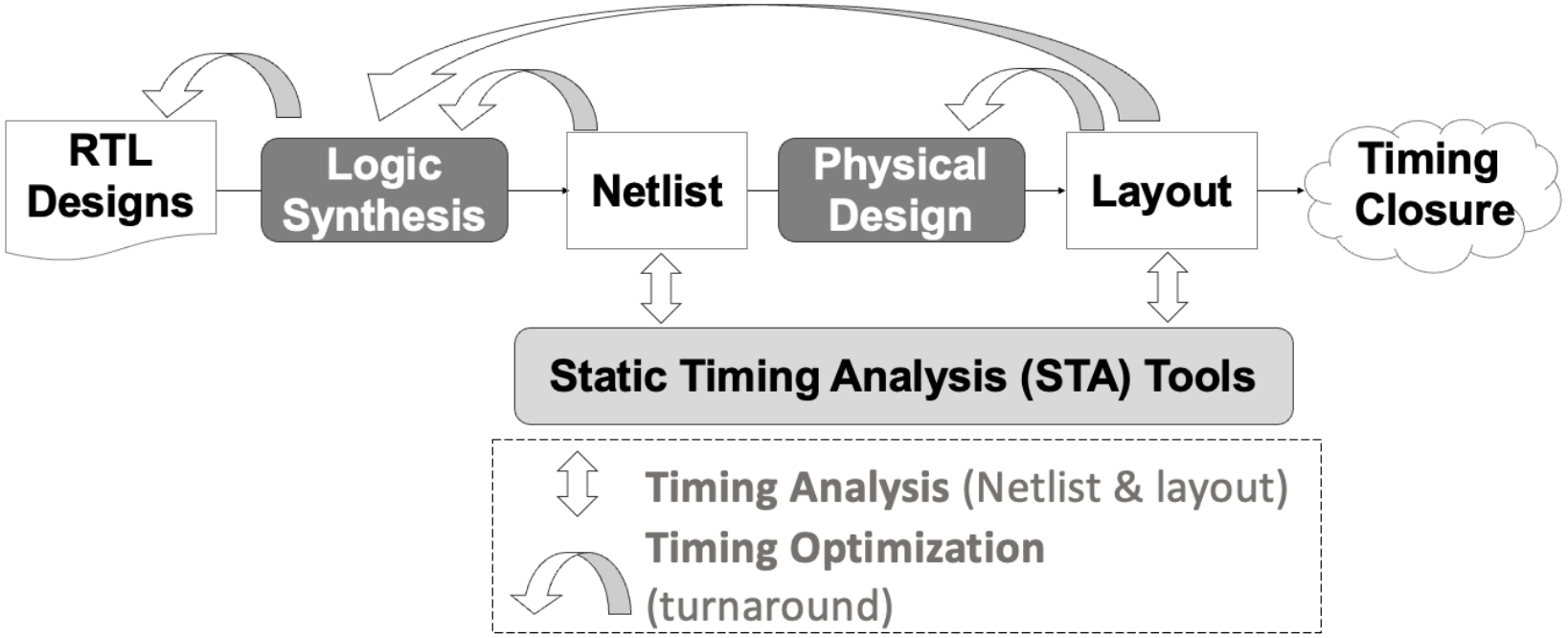
Wenji Fang[1,2], Shang Liu[1], Hongce Zhang[1,2], Zhiyao Xie[1]

wenjifang1@ust.hk

[1] Hong Kong University of Science and Technology

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

Background: Timing Evaluation in Design Flow



- **Timing in the design flow**
  - Performance – sign-off criterion
  - Turnaround optimization for timing closure
- **Existing timing evaluation methods**
  - **1. Traditional:** STA
  - **2. ML-based:** early-stage prediction

Prior Works: ML for Early-Stage Timing Evaluation

- **Netlist & Layout Stage**
  - Existing explorations most target these stages ([DAC'22[1], DAC'23[2], etc.])
  - Predict later-stage **register slack** with timing-related **physical features**
  - Customize ML models based on **STA mechanism**
  - Only **available after logic synthesis**
- **RTL Stage**
  - More challenging (vs. netlist & layout): **no physical information in RTL code!**

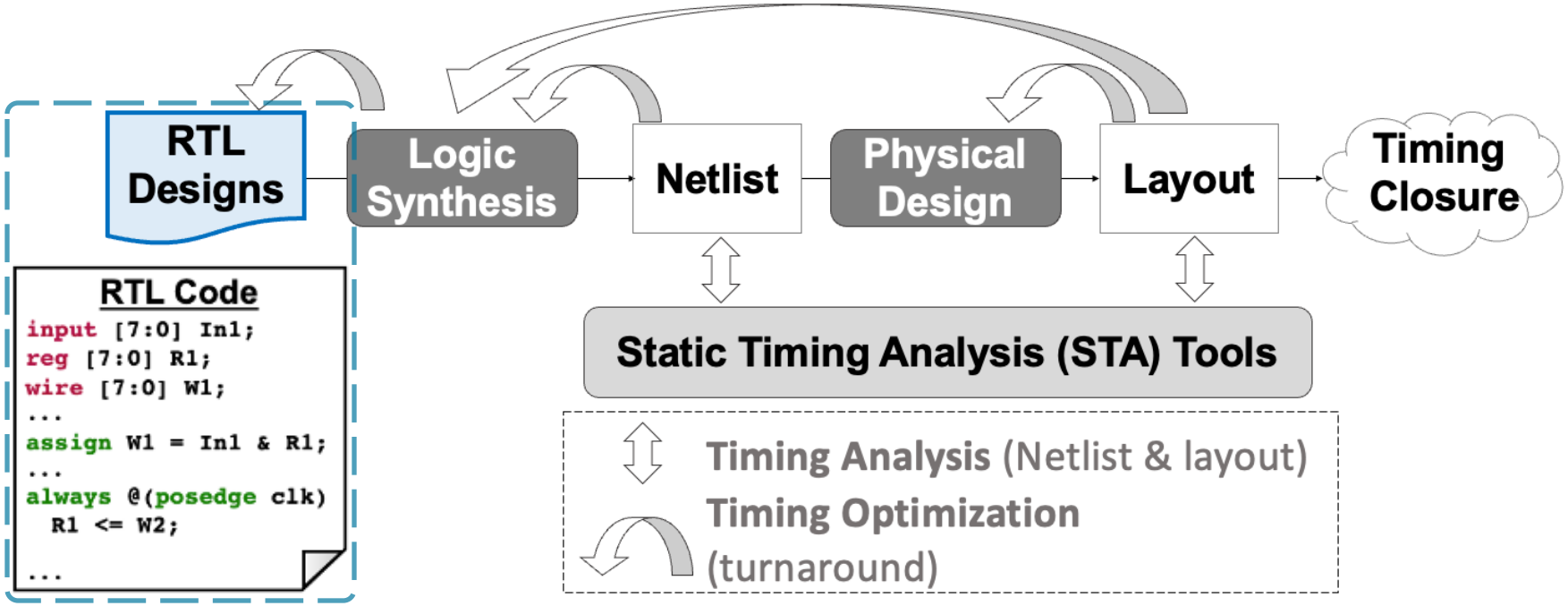
[1]Guo et al. A timing engine inspired graph neural network model for pre-routing slack prediction. In DAC'22.
[2]Wang et al. Restructure-Tolerant Timing Prediction via Multimodal Fusion. In DAC'23.

Prior Works: ML for Early-Stage Timing Evaluation

- **RTL Stage**
  - Only coarse-grained overall WNS/TNS or combinational delay
  - No prior work evaluates fine-grained timing slack at the RTL stage
  - No prior work applies early timing optimization at the RTL stage

Methods	Fine-Grained	General Solution		Applied in Optimization
		Sequential	Cross-Design	
Lopera (ICCAD'22)			✓	
Xu (ISCA'22)		✓	✓	
Sengupta (ICCAD'22)		✓	✓	
Fang (ICCAD'23)		✓	✓	
Ouyang (MLCAD'23)			✓	
Lopera (MLCAD'23)	✓		✓	
Sengupta (MLCAD'23)	✓	✓		✓*
RTL-Timer	✓	✓	✓	✓

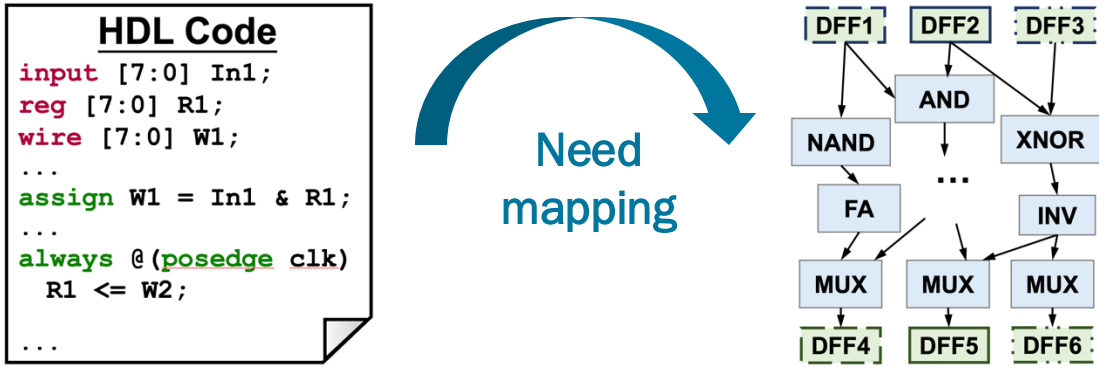
Motivation: RTL-Stage Timing Slack Evaluation



- Can we evaluate fine-grained timing slack earlier at the RTL stage?
  - RTL describes functional behaviors with HDL code
  - Annotate slack directly on HDL
  - Earlier stage with higher optimization flexibility for designers/EDA tools vs. post-syn

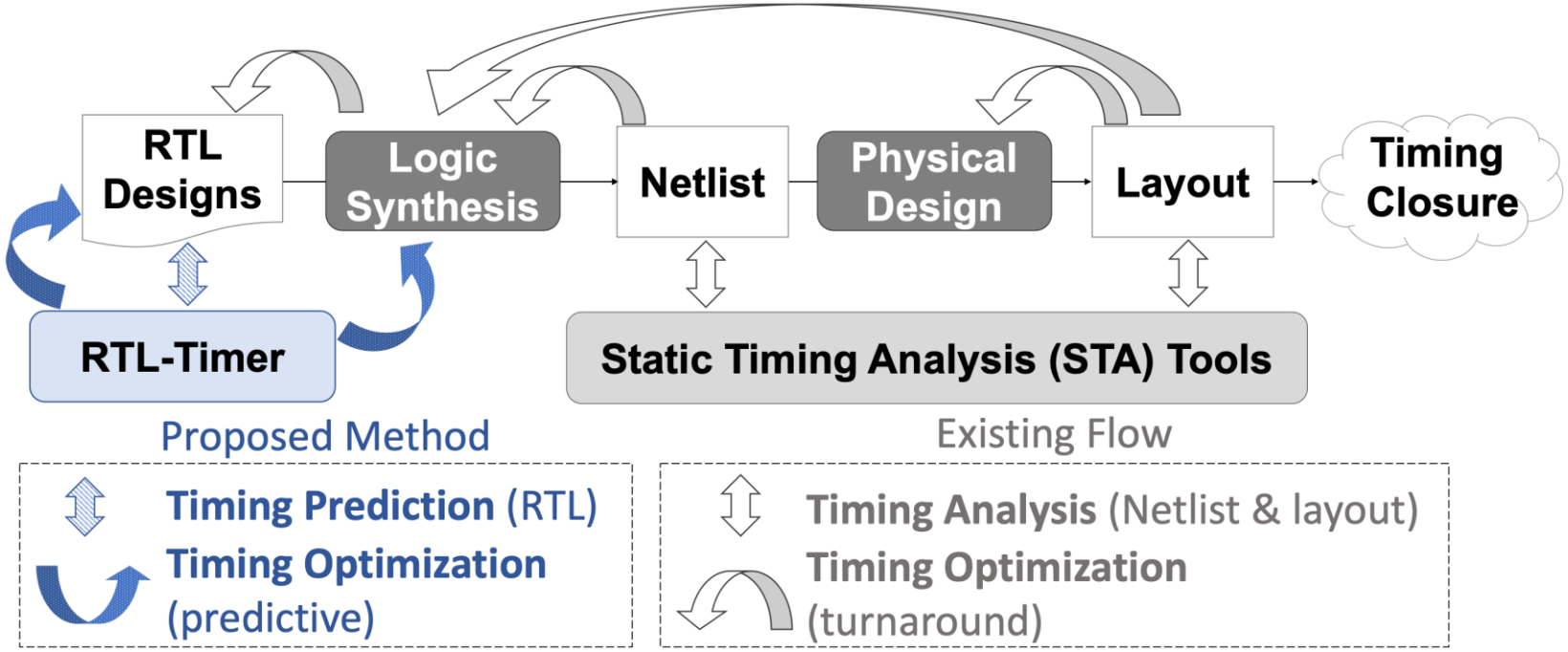
Challenges in RTL-Stage Prediction

- Design RTL is originally in HDL code format
  - Cannot be directly processed by either ML or traditional STA tools
- No direct mapping between most RTL signals and post-syn cells/nets
  - Cannot annotate delay labels



(a) RTL (b) Netlist

Our Solution: RTL-Timer in Design Flow

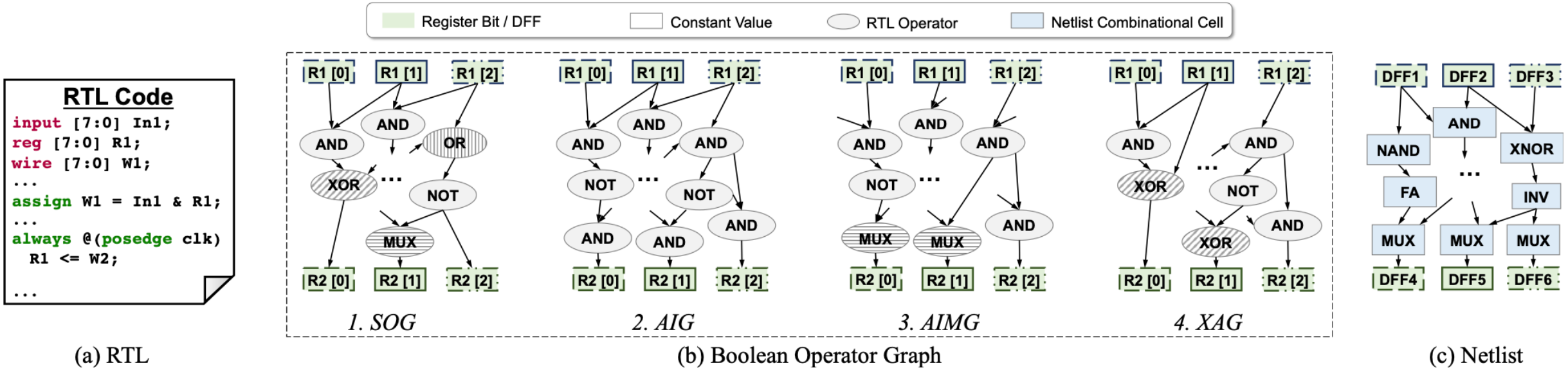


- RTL-stage fine-grained timing slack evaluation
  - Arrival time value prediction
  - Arrival time critical ranking
- Enable predictive optimizations
  - Annotate slack on HDL → RTL designer
  - Opt in commercial logic synthesis tool

Existing RTL Representations

- **Logic transformation**
  - Logic synthesis & verification
    - Binary Decision Diagrams (BDD)
    - Conjunctive Normal Form (CNF)
    - And-Inverter Graphs (AIGER)
    - Btor2
  - Not optimized for ML-based solutions
  - Without the correlation between RTL and netlist
- **ML-based modeling**
  - Design quality prediction
    - Abstract syntax tree (AST) ([SNS])
    - Simple operator graph (SOG) ([MasterRTL])
    - And-Inverter Graphs (AIGER)
  - Ad-hoc solutions
  - Without systematically exploring better candidates

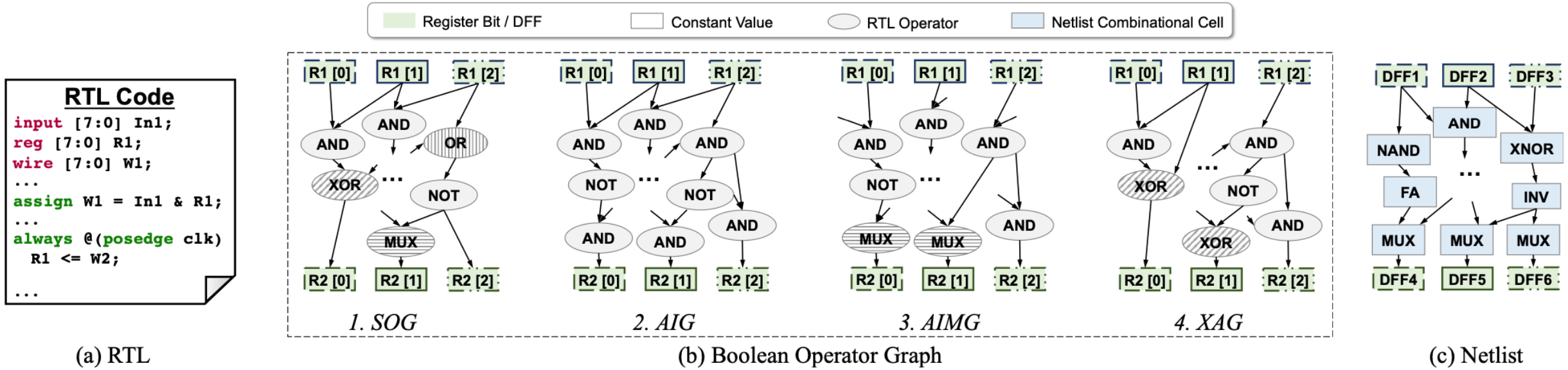
Universal ML-friendly RTL Representation



(a) RTL (b) Boolean Operator Graph (c) Netlist

- **Proposed representation: Boolean Operator Graph (BOG)**
  - Universal bit-level RTL representation
  - Specialized into different variants (SOG, AIG, XAG, etc.) → multi-view for each design

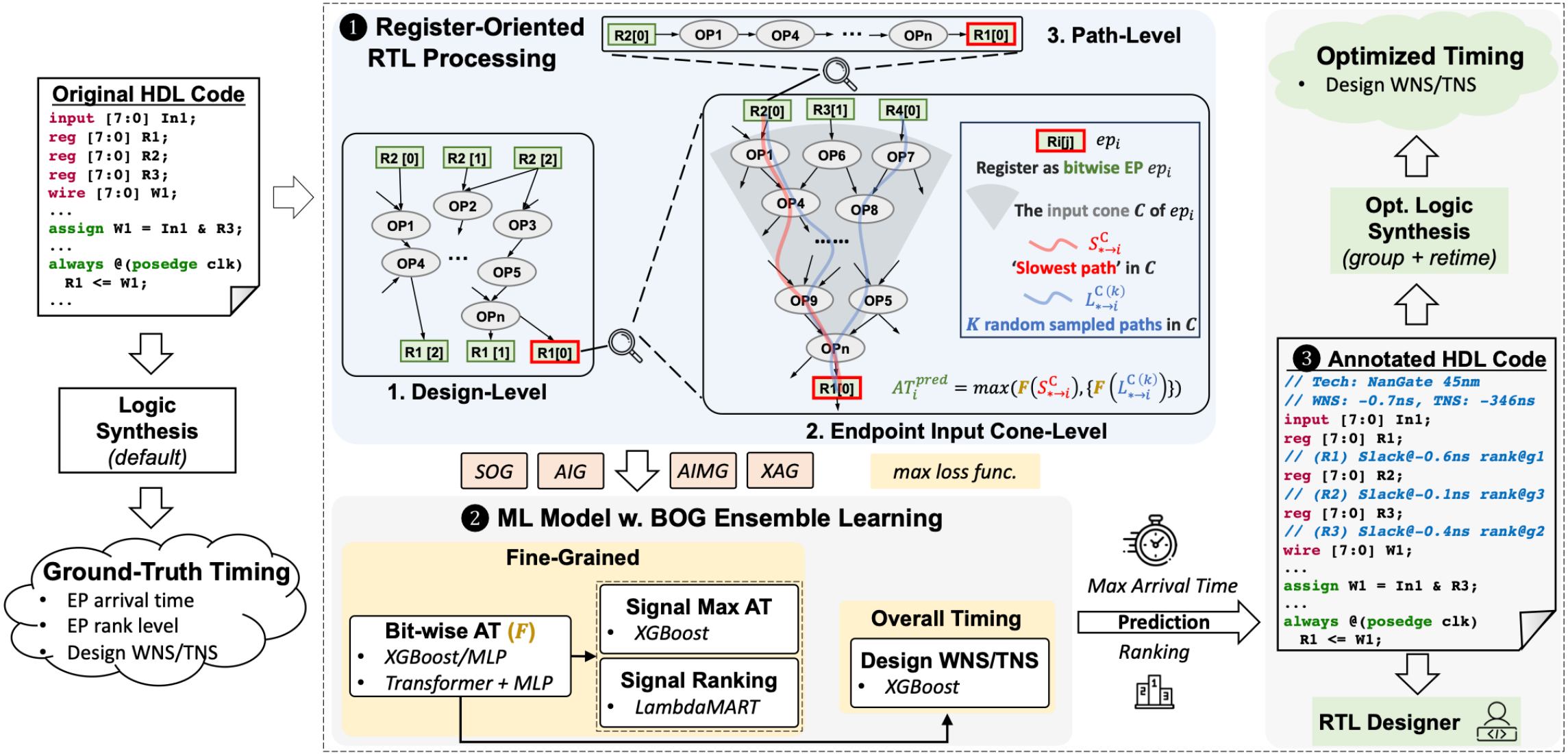
Boolean Operator Graph



(a) RTL (b) Boolean Operator Graph (c) Netlist

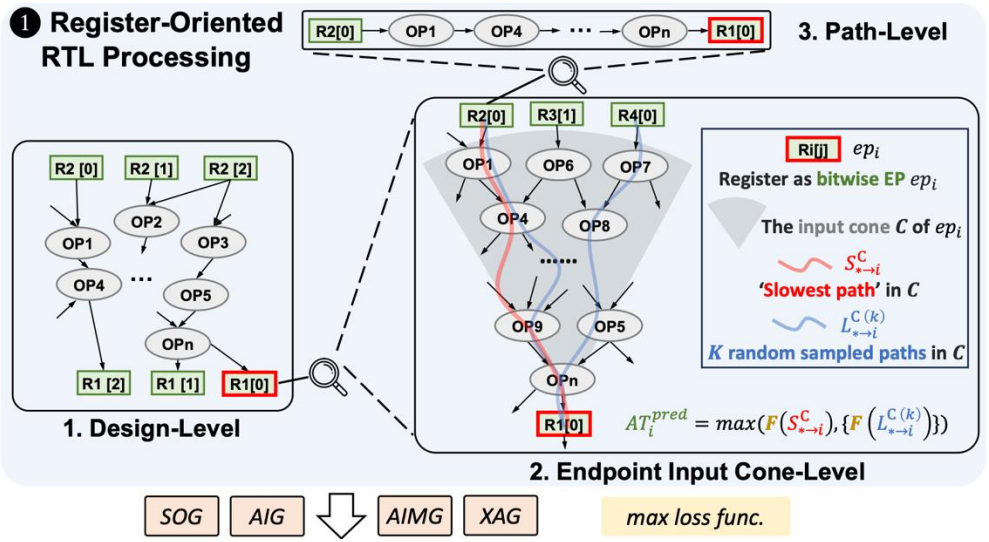
- One-to-one mapping of registers
  - Sequential RTL signal bits ⇔ bit-wise netlist registers
  - Annotate slack label on each bit-wise RTL register for fine-grained ML training
- Treat BOG as a pseudo netlist – We can directly perform STA on it!
  - Registers and operators: standard cells from the liberty file

Workflow of RTL-Timer



1. Register-Oriented RTL Processing Workflow

- **Inspired by STA propagation**
  - Endpoint accumulates arrival time AT from all its driving registers (cone)
- **Capture timing-related information from register cone**
  - STA on BOG → slowest path $S^{C}_{*\to i}$ (not real critical path)
  - *K* random sampled paths $L^{C(k)}_{*\to i}$
  - Customized max AT loss
    - $AT_i^{pred} = max(F_{AT}(S^{C}_{*\to i}), \{F_{AT}(L^{C(k)}_{*\to i})\})$



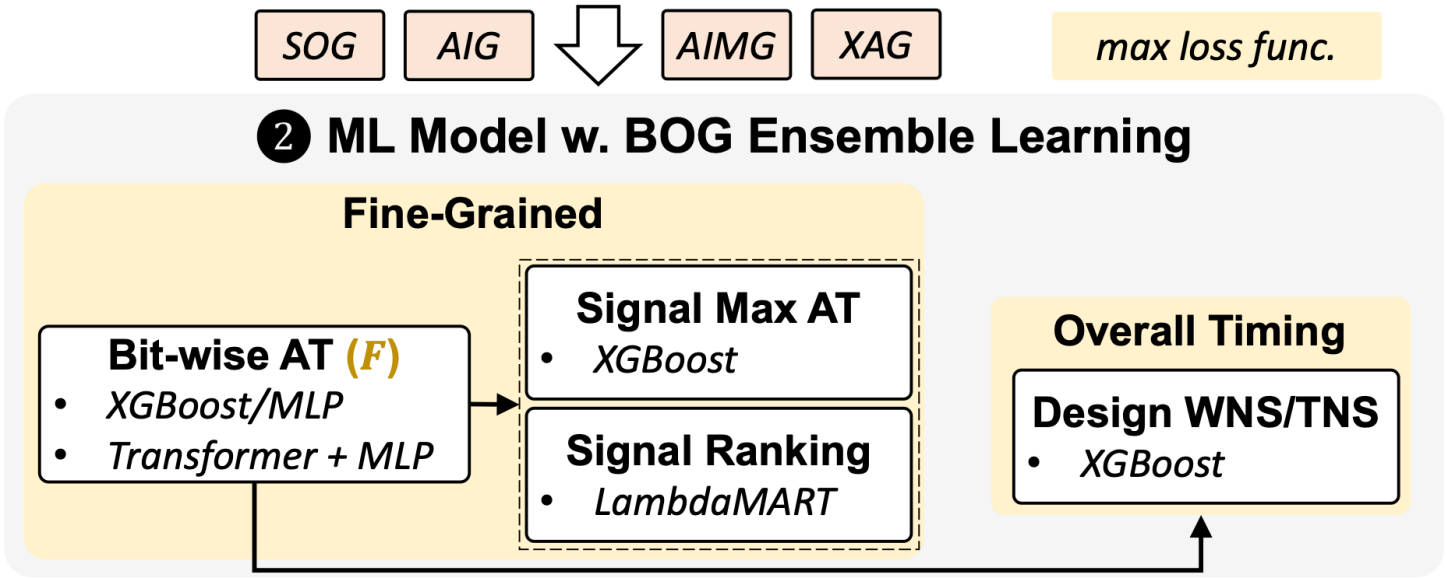
2. ML Modeling in RTL-Timer

- Feature exploration
  - Design-level
    - Global features
    - Compare endpoints across designs
  - Cone-level
    - Size of the cone
  - Path-level
    - Physical-related features on timing paths

Type	Feature	Avg. R
Design	Rank level	/
	% of the endpoint rank	
	# of sequential cells	
	# of combinational cells	
	# of total cells	
Cone	# driving reg of input cone	0.45
Path	Arrival time by STA on $\mathcal{R}$	0.43
	# of level of the timing path	0.51
	# of operators	0.56
	Fanout	0.40
	Load capacitance	0.38
	Slew	0.38

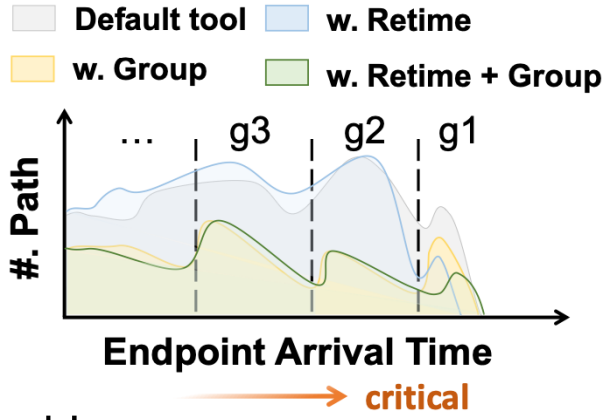
2. ML Modeling in RTL-Timer

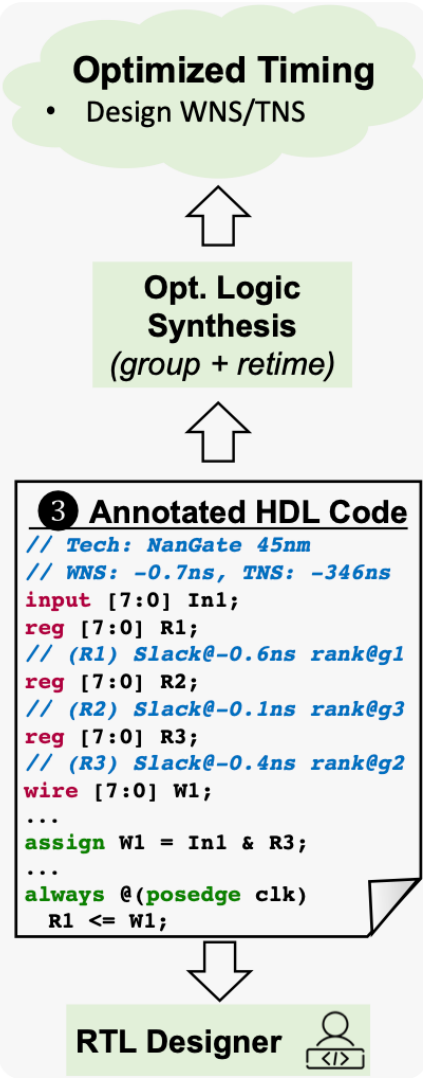
- **Capture max slack on each register endpoint**
  - Customized loss function
- **Ensemble learning with four BOG variants**
  - Multi-view for each design
- **Timing evaluation**
  - Bit-wise endpoint slack
  - Signal-wise endpoint
    - Signal max slack
    - Signal critical ranking
  - Design overall WNS/TNS



3. Optimization Enabled by RTL-Timer

- **Enhancing logic synthesis process**
  - Constraints for commercial tools
    - **Path grouping:** group all endpoints based on ranking
    - **Register retiming:** only top-10% critical endpoints
- **Automatic slack annotation on HDL**
  - Benefit RTL designers
  - Find and optimize timing-critical components





Experiment Setup

- **Dataset**
  - 21 open-source RTL designs

Benchmarks★	#Designs	Design Size Range		HDL Type
		#K Gates	#K Endpoints	
ITC'99	6	9 - 45	0.4 - 1.3	VHDL
OpenCores	4	6 - 56	0.2 - 3.8	Verilog
Chipyard	3	20 - 32	2.5 - 4.1	Chisel
VexRiscv	8	7 - 510	1.2 - 146	SpinalHDL

★ Small designs (<5K Gates) and those dominated by huge memory modules are excluded from the original benchmarks.

- **Label collection**
  - Synopsys Design Compiler / Prime Time + NanGate 45nm PDK
  - Slack on each endpoint register
- **Evaluation metrics**
  - **Regression**
    - Correlation/Determination coefficient (R/R^2)
    - Mean absolute percentage error (MAPE)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y_i}|}{y_i} \times 100\%$$

  - **Ranking**
    - Critical level ranking coverage (COVR)

$$\text{COVR} = \frac{1}{m}\sum_{g=1}^{m}\frac{\#(S_g \cap \hat{S_g})}{\#S_g} \times 100\%$$

Modeling Performance

- **Fine-grained modeling (slack)**
  - Accurate prediction on each reg bit/signal bus
  - **Regression**: XGBoost w. cus. loss performs best (R=0.89)
  - **Ranking**: L2R outperforms all regression models (COVR=80%)
- **Design overall timing modeling (WNS/TNS)**
  - Calibration based on fine-grained modeling
  - RTL-Timer outperforms all baselines

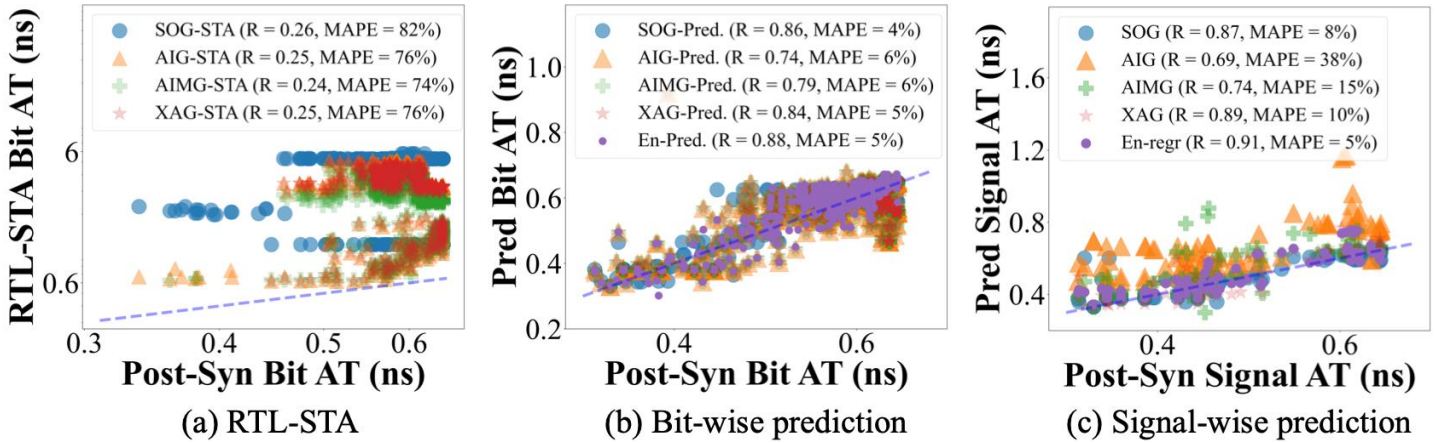
Fine-Grained	Method	R	MAPE (%)	COVR (%)
Bit-wise	Tree-based w/o sample	0.80	26	59
	MLP	0.71	35	56
	MLP w/o sample	0.65	38	54
	Transformer	0.73	35	57
	Customized GNN	0.25	53	46
	RTL-Timer	0.88	12	66
Signal-wise	Regression w/o bit-wise	0.56	28	56
	Ranking w/o bit-wise	/	/	39
	RTL-Timer (regression)	0.89	15	71
	RTL-Timer (ranking)	/	/	80

Overall	Method	R	R^2	MAPE (%)
WNS	SNS [17]	0.73	0.58	33
	MasterRTL [4]	0.89	0.74	15
	RTL-Timer	0.91	0.86	12
TNS	ICCAD'22 [13]	0.65	0.32	42
	MasterRTL [4]	0.96	0.94	34
	RTL-Timer	0.98	0.97	18

Modeling Performance

- BOG variants ensemble learning
  - Each variant contributes to the prediction
  - Robustness across benchmarks and tasks
- Visualization of prediction
  - RTL-STA is not accurate enough
  - Bit-wise calibrates RTL-STA
  - Signal-wise calibrates bit-wise

	Metrics	SOG	AIG	AIMG	XAG	Ensemble
Bit-wise	Avg. R	0.85	0.75	0.76	0.77	**0.88**
	Std. R	0.18	0.25	0.26	0.21	**0.08**
Signal-wise	Avg. R	0.82	0.81	0.84	0.8	**0.89**
	Std. R	0.15	0.22	0.1	0.1	**0.06**
	Avg. COVR	65	71	72	71	**80**
	Std. COVR	18	19	21	21	**8**



(a) RTL-STA (b) Bit-wise prediction (c) Signal-wise prediction

Optimization Performance

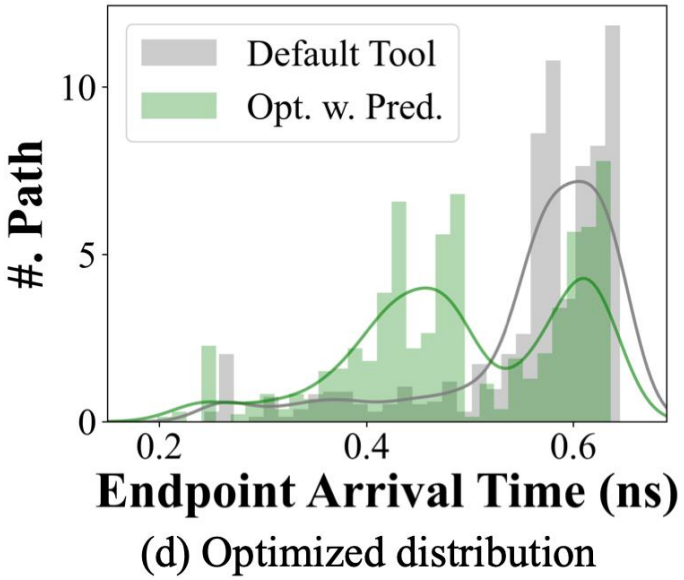
- Guiding commercial synthesis tool
  - Improve WNS(3.1%)/ TNS(9%) for most designs
  - Maintain or even decrease area/power
  - Reduce time-consuming iterations:

    Concurrently run default and opt flows
- Impact after placement
  - Still remains significant after place (w. place opt)
  - WNS(3.1%)/ TNS(6.8%)

Design	Singal-wise Pred.			Opt. w. Pred. (%)				Opt. w. Real (%)			
	R	MAPE	COVR	WNS	TNS	Pwr	Area	WNS	TNS	Pwr	Area
syscdes	0.94	26%	94%	-1.3	-17	-0.8	1.8	-1.8	-17.3	0	2.6
syscaes	0.86	23%	77%	-1.3	-13.7	2.1	3.2	-0.1	-14.4	2.1	3.5
Vex_1	0.87	24%	86%	6.9	7	-3.1	-2.8	-0.3	-1.1	0.5	-0.8
b20*	0.91	7%	86%	5.6	-4.3	26.2	25	0.2	-6.6	26.2	23.4
Vex_2	0.86	16%	83%	-0.2	-1.6	-0.9	0	-0.7	-1.8	-0.6	0.3
Vex_3	0.93	30%	86%	-2.8	-4.8	3.9	1.2	-0.1	-2.2	1.9	1
b22*	0.74	18%	83%	0.7	-4.8	23.4	23.3	2.9	0.4	22.7	20.2
b17	0.93	8%	75%	1.9	-5.2	2.2	0	-0.9	-5.6	0.2	1
b17_1	0.94	5%	79%	5.8	-3.2	1.7	2.1	0.9	-5.8	-0.6	0.4
Rocket1	0.89	11%	63%	-7.1	-21.4	2.8	1.7	-3.7	-25.4	-66	2.9
Rocket2	0.92	18%	64%	-7	-23.1	-69.4	-0.6	-4.1	-23.1	1.6	-0.8
Rocket3	0.88	12%	69%	-7.2	-17.4	-69.4	-0.6	-6.2	-18.3	-69.7	0.8
conmax	0.91	12%	83%	-1.9	-3	3.4	2.7	-0.9	-0.6	3	2.1
b18	0.82	13%	84%	-16.4	-33.5	3.8	3.9	-17.9	-35.5	3.6	3.4
b18_1	0.88	10%	86%	-3.9	-26	0.5	-0.4	-9.8	-27	1.8	0.2
FPU	0.89	31%	85%	6.2	0.2	0.5	1	4.3	-1.7	-86.6	0.6
Marax	0.88	16%	78%	-1.7	-3	-0.1	-0.1	2.4	-2.2	0	-0.5
Vex_4	0.79	16%	67%	-4.8	-18.6	0	-0.7	-7.2	-21.9	0.4	-1
Vex5	0.92	4%	81%	-1.8	-4.6	0.2	-1.3	-3.6	-10.7	0.3	-0.6
Vex6	0.94	6%	82%	-1.5	-12.8	0.4	-0.5	-3	-5.3	0.3	-1.3
Vex7	0.87	6%	81%	-5.7	-7.2	0.5	-0.8	-3	-12.3	0.5	-0.2
Avg1	**0.89**	**15**	**80**	**-1.9**	**-10.4**	**-3.4**	**2.8**	**-2.5**	**-11.2**	**-7.5**	**2.7**
Avg2				**-3.1**	**-9.9**	**-5.9**	**0.5**	**-3**	**-10.6**	**-5.7**	**0.6**

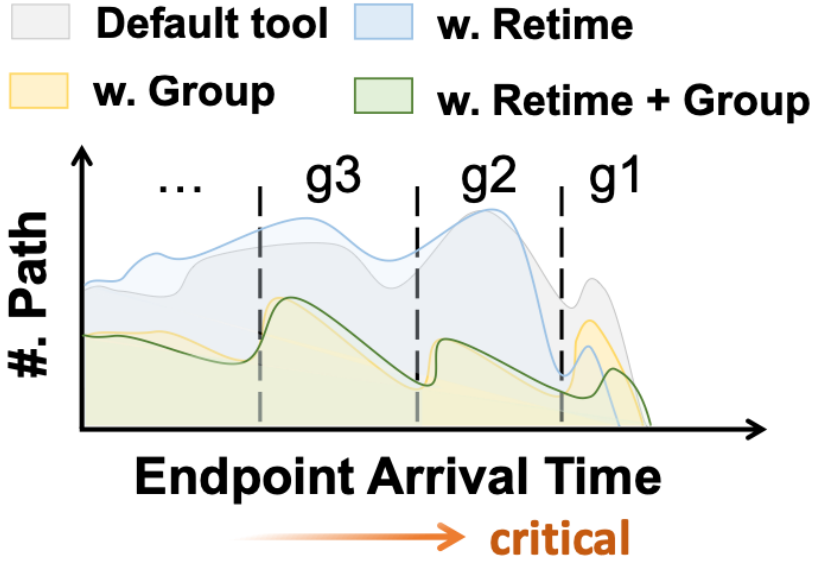
Optimization Performance

- Optimization impact on slack distribution
  - **Path grouping**: single high peak → two lower peaks (better TNS)
  - **Register retiming**: improved WNS



(d) Optimized distribution

Experimental



Theoretical

Runtime Analysis

- **Prediction**
  - 4% of default logic synthesis runtime
  - Two key parts:
    - RTL processing
      - HDL to BOG (parallel): **3.2%**
      - Register-oriented processing: **0.8%**
    - Model inference: **<0.1s**
- **Optimization**
  - Logic synthesis w/ optimization
  - Runtime extends by an average of **45%** vs. default synthesis flow

Conclusion

- **RTL-Timer: estimate slack on each register at the RTL stage**
  - Ensemble four ML-friendly RTL representations
  - Capture max slack with register-oriented RTL processing and customized ML model
- **Enable early timing optimization**
  - Annotate slack on HDL code for RTL designers
  - Predictive timing optimization for logic synthesis process

Available at: https://github.com/hkust-zhiyao/RTL-Timer