



復旦大學
FUDAN UNIVERSITY

PASTE: A Physical-Aware Surrogate for FPGA Pre-Routing Timing Estimation

Yunfei Dai, Kaixiang Zhu, Mingyang Chen, Lingli Wang

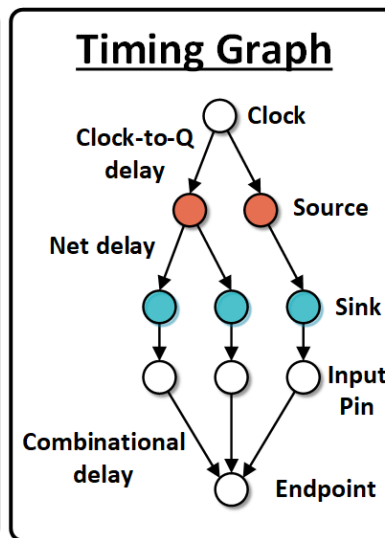
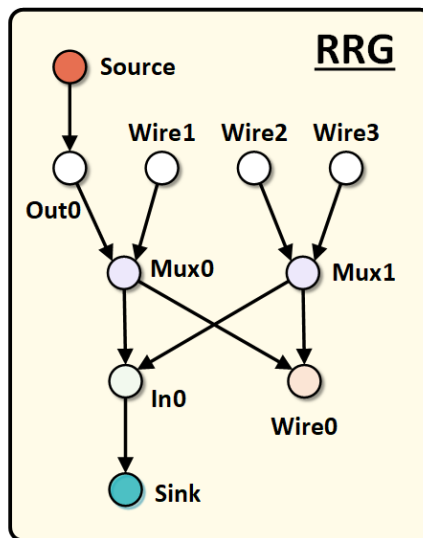
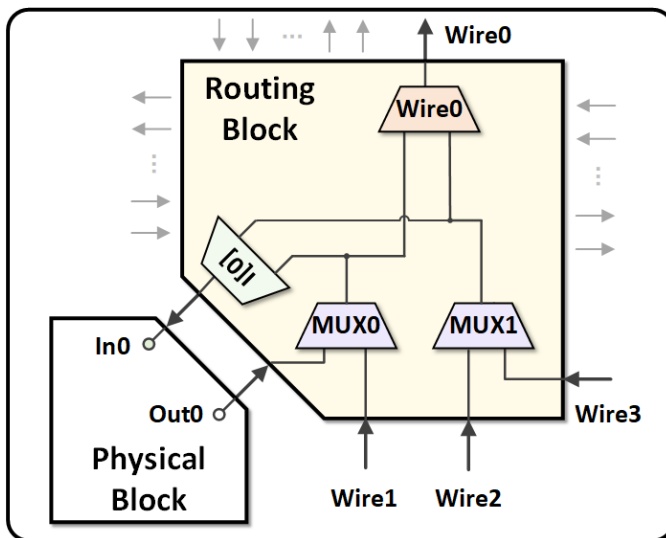
State Key Lab of Integrated Chips and Systems, Fudan University

- 1 Intro & Prior Works
- 2 Method — PASTE
- 3 Experiments
- 4 Application: Predictor-Guided Retiming
- 5 Conclusion

- **Difficulty:**
 - **Routing Delay:**
 - timing-graph topology
 - placement
 - architecture-constrained routing fabric
- **Beneficial to Timing-driven optimizations**
 - logic restructuring
 - placement refinement
 - sequential optimization
- **Pre-routing timing prediction provides early timing guidance without incurring the prohibitive cost of iterative routing.**

- **Indirect proxies (not direct timing)**
 - Wirelength prediction — Pandit & Akoglu, FPL 2007
 - Congestion estimation — Maarouf et al., FPL 2018
 - **Restricted delay subproblems**
 - HLS operation-level delay — Ustun et al., ICCAD 2020
 - **Routing-cost heuristics inside CAD tools**
 - AIR lookahead router — Murray et al., ASP-DAC 2020
 - **Closest prior work**
 - RRG-GAE (node-level arrival time) — Dai et al., FPT 2022
 - but: per-net routing modeling (scales w/ #nets), not end-to-end
- Gap: a scalable, end-to-end, node-level pre-routing timing surrogate

- **Physically informative features on the post-placement timing graph**
 - normalized timing prior + routing-density signals
 - from a design-aware pruned Routing Resource Graph
- **Timing Propagation Network (TPN) — two-stage GNN**
 - heterogeneous message passing + edge-gated max propagation
 - explicitly mirrors STA max-plus semantics
 - sub-second inference on 400k+ node designs
- **Strong results, validated downstream**
 - node-level MAPE 8.48% (48 benchmarks)
 - PASTE-guided retiming: Fmax +30.83% (beats ABC retiming)



Timing graph

- Path delay = clock-to-Q + net delay + combinational delay
- Arrival time propagates along directed timing arcs and depends on both local delays (net/combinational) and topology.

FPGA ≠ ASIC for interconnect

- In ASIC flows, a net is often approximated as a geometric interconnect whose delay mainly depends on placement.
- An FPGA net is routed through a fixed routing fabric by traversing programmable switches
- delay set by switch-box topology & local congestion, not distance

Routing Resource Graph (RRG)

- nodes = routing resources / pins; edges = programmable connections

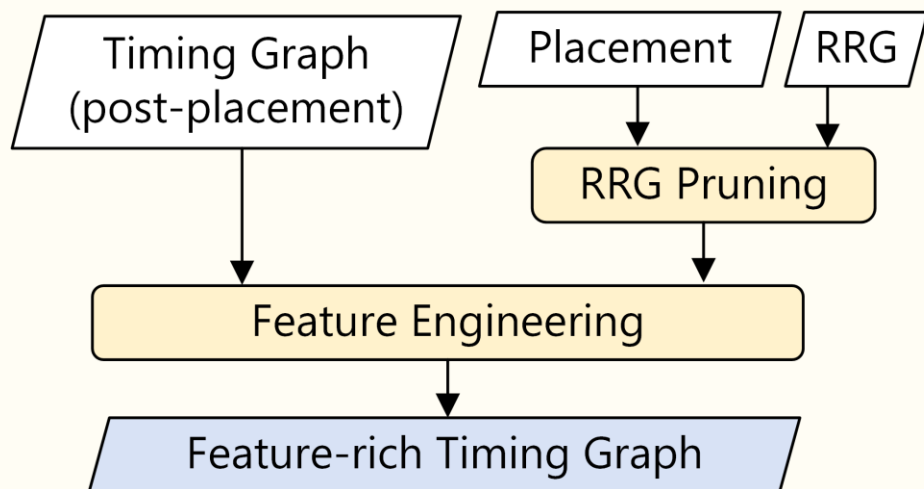
GNN Layer

$$\mathbf{h}_v^{(l+1)} = \phi\left(\mathbf{h}_v^{(l)}, \text{AGG}_{u \in \mathcal{N}(v)} \psi\left(\mathbf{h}_v^{(l)}, \mathbf{h}_u^{(l)}, \mathbf{e}_{uv}\right)\right), \quad (1)$$

where $\mathbf{h}_v^{(l)}$ is the representation of node v at layer l , $\mathcal{N}(v)$ denotes its neighbors, and $\mathbf{e}_{uv}$ denotes edge features.

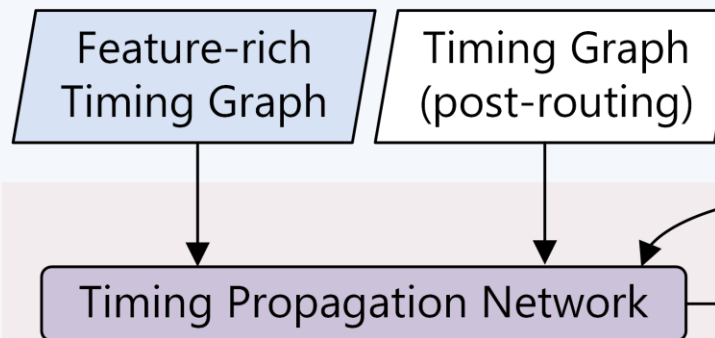
Overview

Data Processing:

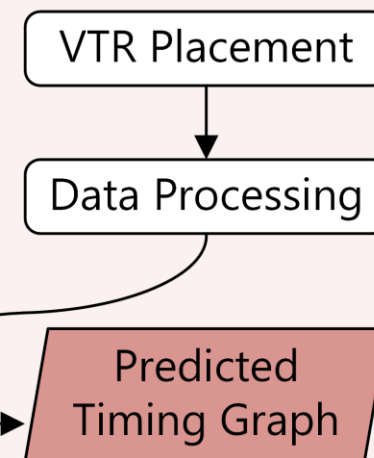


Training Flow:

4 archs × 5 channel widths
48 circuits × 64 placement configs



Inference Flow:



1 Feature engineering

Design-aware RRG pruning → placement geometry, routing-density & timing-prior features on the timing graph.

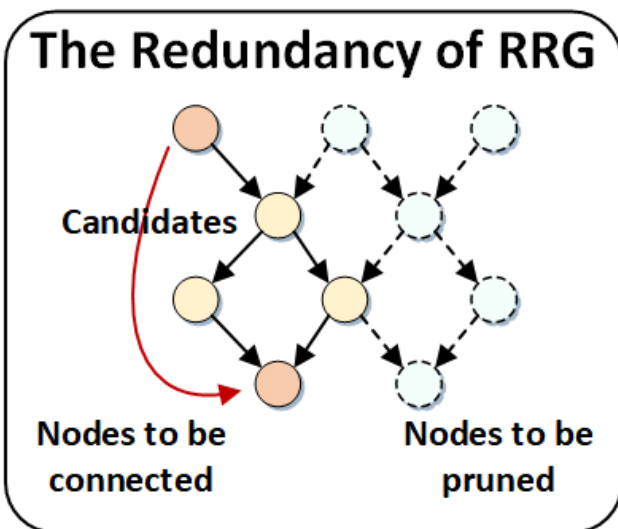
2 TPN (2-stage GNN)

Stage I heterogeneous message passing (local) + Stage II edge-gated max propagation (long-range).

3 Dual prediction

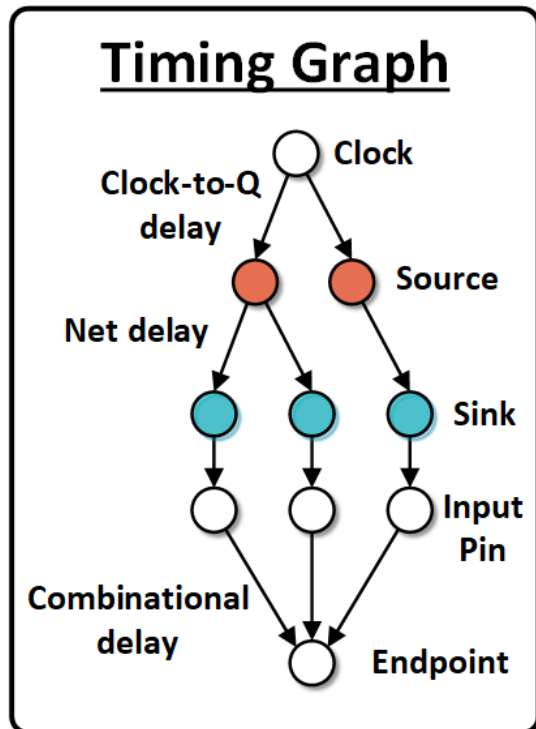
Node-level arrival time + graph-level CPD.
No per-net routing model → sub-second on 400k+ nodes.

- **Problem: full RRG counts resources the design never uses**
 - e.g. a CLB-heavy design still counts idle memory-block resources
 - → inflates the resource space, dilutes congestion statistics
- **Solution: BFS reachability pruning**
 - leave only the routing resources actually relevant to the placed design



Keep only resources on a source→sink path

- ① MAP each timing node → its RRG source / sink
- ② forward-BFS from valid sources
- ③ backward-BFS from valid sinks
- ④ keep $V = S_valid \cap T_valid$ (edges induced)



Node types:

- SOURCE/SINK
- IPIN/OPIN
- CPIN

Edge types:

- combinational
- clock-to-Q
- Setup
- Interconnect

NODE features (Table III)

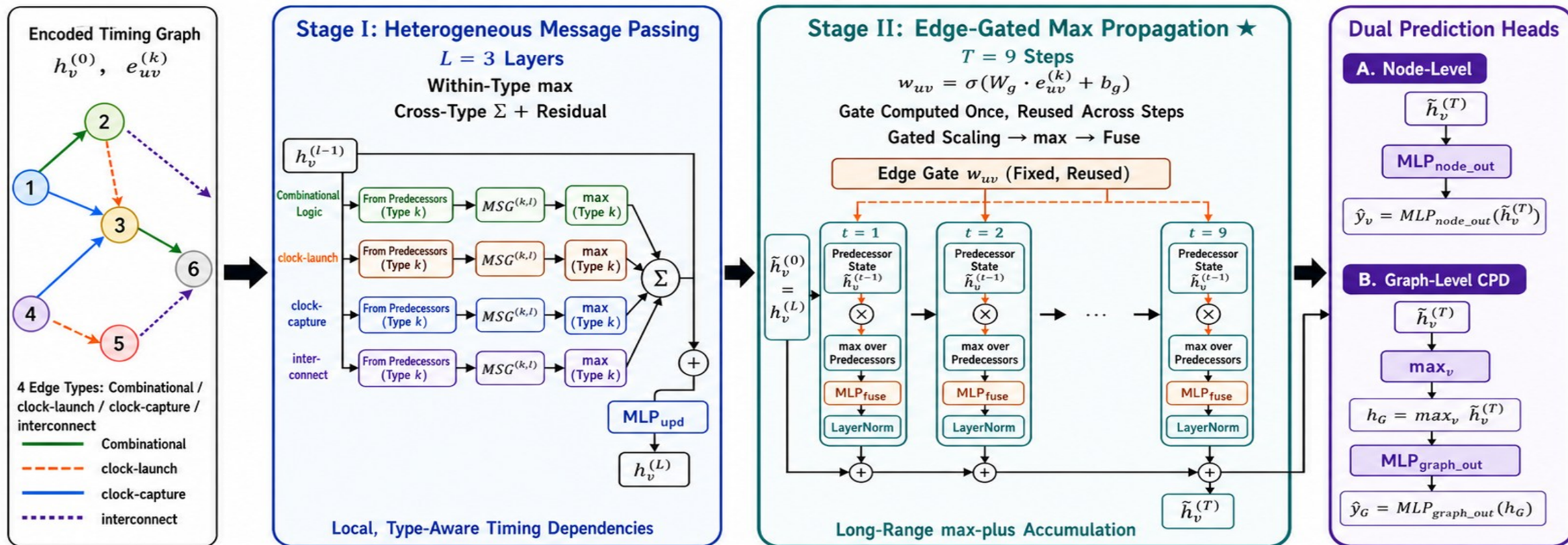
Placement	x, y coordinates
Topology	logic level · global-max level · type (1-hot) · fanin/fanout
Timing prior	post-placement arrival time, $\tilde{t}_v = t_v / t_{\max}$

EDGE features (Table IV) · on pruned RRG

Geometry	HPWL · Manhattan distance
BBox density	node-density (max/avg) · edge-usage-density (max/avg)

Post-placement graph already carries a timing prior: edge delays include logic delay and nodes carry placement-stage timing annotations.

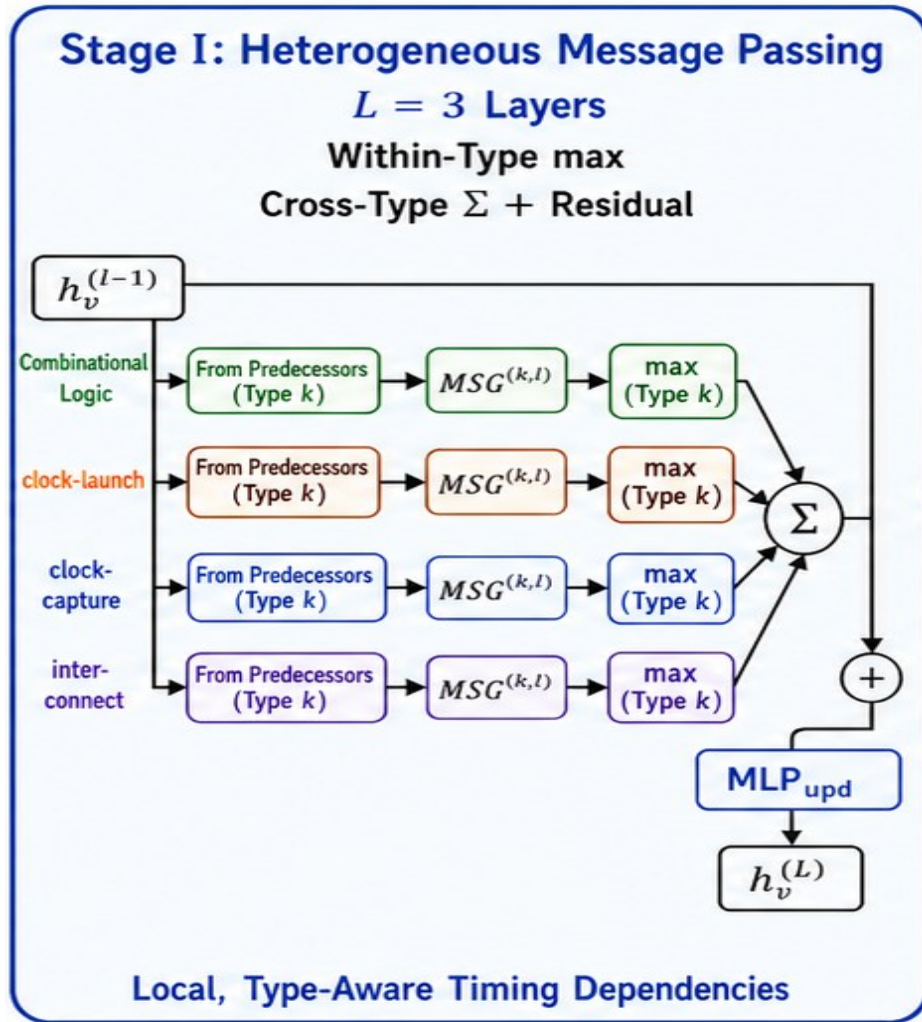
Timing Propagation Network



Key Idea:

- The expensive full STA runs only once, at convergence, for final evaluation. Runtime is dominated by placement.
- We keep the efficiency of heuristic retiming but replace the crude unit-delay assumption with learned physical timing.

Timing Propagation Network — Stage I

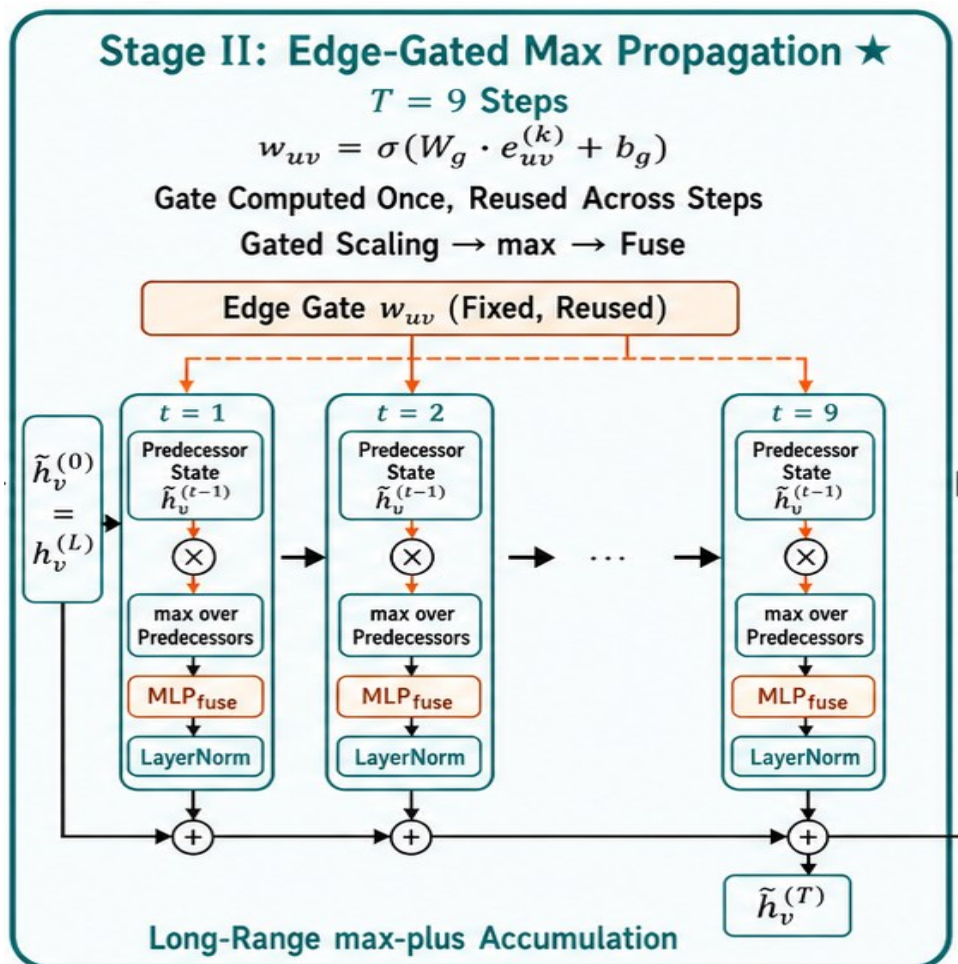


$$\mathbf{m}_v^{(k,l)} = \max_{u \in \mathcal{N}_k(v)} \text{MSG}^{(k,l)} \left(\mathbf{h}_u^{(l-1)}, \mathbf{e}_{uv}^{(k)} \right)$$

- The messages are aggregated using a maximum operation within each edge type.
- It reflects STA physics, because only the latest-arriving predecessor determines the arrival time.

$$\mathbf{h}_v^{(l)} = \text{MLP}_{\text{upd}}^{(l)} \left(\mathbf{h}_v^{(l-1)} + \sum_k \mathbf{m}_v^{(k,l)} \right)$$

Timing Propagation Network — Stage II



$$w_{uv} = g(\mathbf{e}_{uv}^{(k)}) = \sigma(\mathbf{W}_g \mathbf{e}_{uv}^{(k)} + b_g)$$

- It acts as a fixed, learnable prior that controls how strongly timing information propagates through each edge.

$$\mathbf{m}_v^{(t)} = \max_{u \in \text{Pred}(v)} (w_{uv} \tilde{\mathbf{h}}_u^{(t-1)}),$$

$$\tilde{\mathbf{h}}_v^{(t)} = \text{Fuse}^{(t)}(\tilde{\mathbf{h}}_v^{(t-1)}, \mathbf{m}_v^{(t)}),$$

It captures long-range max-plus accumulation while keeping nine-step propagation stable

$$\text{Fuse}^{(t)}(\mathbf{h}, \mathbf{m}) = \text{LN}(\mathbf{h} + \text{MLP}^{(t)}([\mathbf{h} \parallel \mathbf{m}]))$$

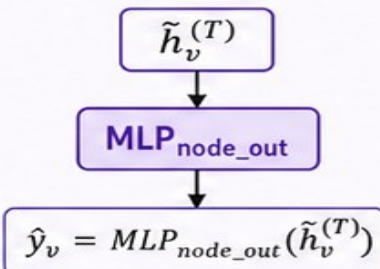
Timing Propagation Network — Prediction Heads



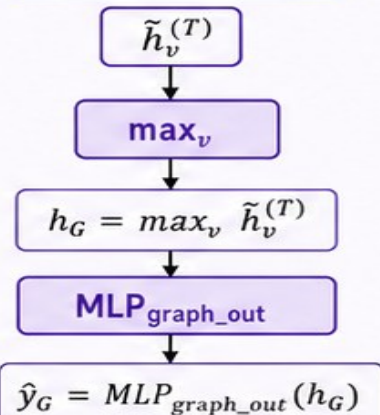
復旦大學

Dual Prediction Heads

A. Node-Level

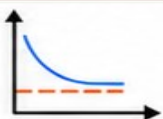


B. Graph-Level CPD



Joint Training: $L = L_{\text{node}} + \alpha \cdot L_{\text{graph}}$, $\alpha = 1.0$

$$\mathcal{L} = \frac{1}{|\mathcal{V}_{\text{tg}}|} \sum_{v \in \mathcal{V}_{\text{tg}}} \left(\hat{y}_v - \frac{t_v}{t_{\text{max}}^{\text{pl}}} \right)^2 + \alpha \left(\hat{y}_G - \frac{\text{CPD}}{t_{\text{max}}^{\text{pl}}} \right)^2$$



Normalize by $t_{\text{max_pl}}$; De-normalize at Inference

$$t_{\text{max}}^{\text{pl}} = \max_{u \in \mathcal{V}_{\text{tg}}} t_u^{\text{pl}}, \quad \tilde{t}_v^{\text{pl}} = \frac{t_v^{\text{pl}}}{t_{\text{max}}^{\text{pl}}}$$

The maximum post-placement arrival time of the design.

Experimental Setup

- **Dataset: 48 circuits (VTR8 + MCNC), 3 variation axes**

- Resource: 5 logic blocks $\times$ 5 channel widths
- Placement: seeds $\times$ inner_num $\times$ fix_pins $\times$ algorithm
- $\approx 70,000$ valid design instances

- **Supervision: post-routing STA (node arrival + CPD)**

- **3 generalization protocols**

- cross-circuit (24/7/17) · cross-arch (3/1/1) · cross-channel (3/1/1)

- **Baselines: GCN · GIN · GraphSAGE · GAT**

- **4 Metrics**

MAPE ↓

relative node arrival-time error

R² ↑

overall regression quality

Spearman ρ ↑

ranking consistency of critical nodes

CPD MAPE ↓

graph-level critical-path-delay error

TABLE VI: Comparison of different methods under circuit-level, cross-architecture, and cross-channel-width evaluation settings. Best results in each row are highlighted in bold.

Circuit	GCN			GIN			GraphSAGE			GAT			Ours		
	R^2	MAPE (%)	ρ	R^2	MAPE (%)	ρ	R^2	MAPE (%)	ρ	R^2	MAPE (%)	ρ	R^2	MAPE (%)	ρ
Circuit-level test results															
bgm	0.94	46.37	0.94	0.86	55.95	0.94	0.93	50.42	0.95	0.94	49.43	0.91	0.93	16.47	0.93
blob_merge	0.86	9.61	0.86	0.87	9.35	0.86	0.85	10.21	0.84	0.84	10.42	0.84	0.91	5.39	0.91
boundtop	0.86	10.15	0.88	0.87	9.35	0.88	0.85	10.01	0.87	0.87	9.76	0.88	0.85	6.71	0.87
ch_intrinsics	0.83	10.23	0.91	0.86	9.06	0.91	0.84	9.92	0.90	0.85	9.25	0.92	0.87	5.99	0.92
diffeq1	0.96	12.71	0.88	0.94	12.39	0.87	0.92	16.00	0.85	0.92	14.83	0.85	0.99	9.93	0.92
diffeq2	0.99	14.92	0.92	0.99	14.19	0.92	0.99	15.22	0.92	0.99	14.36	0.92	0.99	11.59	0.92
mcml	0.99	10.31	0.91	0.99	13.88	0.90	0.99	11.23	0.90	0.99	11.28	0.90	0.99	3.61	0.92
mkDelayWorker32B	0.72	13.99	0.89	0.71	13.92	0.88	0.73	13.60	0.88	0.74	13.57	0.89	0.72	9.33	0.88
mkPktMerge	0.90	10.14	0.85	0.90	9.96	0.85	0.89	11.16	0.84	0.89	10.35	0.85	0.88	7.56	0.84
mkSMAadapter4B	0.94	11.65	0.97	0.95	10.77	0.97	0.94	12.05	0.96	0.94	11.91	0.96	0.95	7.57	0.97
or1200	0.96	6.24	0.97	0.96	6.22	0.97	0.95	6.82	0.97	0.96	6.40	0.97	0.97	3.67	0.98
raygentop	0.80	11.21	0.90	0.80	10.55	0.89	0.81	12.10	0.89	0.81	12.87	0.90	0.84	8.12	0.91
sha	0.84	10.68	0.87	0.85	9.64	0.88	0.83	11.07	0.87	0.84	11.11	0.87	0.80	6.97	0.92
s38417	0.67	20.64	0.81	0.68	16.64	0.81	0.75	25.33	0.87	0.75	21.77	0.85	0.86	9.11	0.90
frisc	0.89	16.78	0.95	0.81	22.00	0.91	0.83	21.33	0.93	0.81	19.44	0.92	0.92	8.12	0.97
dsip	0.89	13.42	0.85	0.90	13.11	0.83	0.74	19.56	0.89	0.94	9.06	0.89	0.86	11.55	0.85
tseng	0.55	37.36	0.80	0.66	29.11	0.83	0.74	28.63	0.85	0.68	33.37	0.88	0.85	12.52	0.91
Average	0.85	15.67	0.89	0.86	15.65	0.89	0.86	16.74	0.89	0.87	15.83	0.89	0.89	8.48	0.91
Cross-architecture test results															
Test set average	0.87	12.74	0.90	0.88	12.28	0.90	0.87	13.55	0.89	0.87	12.81	0.90	0.88	8.51	0.91
Cross-channel-width test results															
Test set average	0.85	13.52	0.90	0.86	12.96	0.89	0.86	13.74	0.89	0.86	13.08	0.89	0.89	10.01	0.91

Results — Circuit-Level Comparison & Ablation Study



復旦大學

TABLE VII: Comparison of Top-10 overlap and CPD MAPE under different evaluation settings, including generic GNN baselines, RRG-GAE [2], and ablated variants of the proposed model.

Method	Top-10 overlap			CPD MAPE		
	Circuit	Channel	Arch	Circuit	Channel	Arch
GCN	0.47	0.49	0.48	12.04	15.78	16.98
GIN	0.45	0.52	0.55	42.85	38.78	45.32
GraphSAGE	0.52	0.59	0.54	20.27	19.44	18.35
GAT	0.49	0.52	0.54	12.47	13.47	15.56
RRG-GAE [2]	0.34	0.51	0.50	31.77	24.77	21.83
w/o Stage II	0.53	0.64	0.54	17.88	16.15	17.14
w/o Stage I	0.57	0.68	0.60	40.16	39.58	40.62
Ours	0.58	0.70	0.60	8.28	7.55	8.39

$$\text{Top-10 Overlap} = \frac{|\mathcal{T}_{\text{pred}}^{10} \cap \mathcal{T}_{\text{true}}^{10}|}{10} \times 100\%,$$

Circuit-level Comparison:

- Three settings: cross-circuit, cross-channel and cross-architecture.
- PASTE achieves the highest overlap on the predicted and real longest 10 paths.
- Low CPD MAPE shows the capability in absolute timing value prediction.

Ablation Study:

- Stage I gives local accuracy, Stage II gives long-range propagation.

GCN: T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in ICLR, 2017.

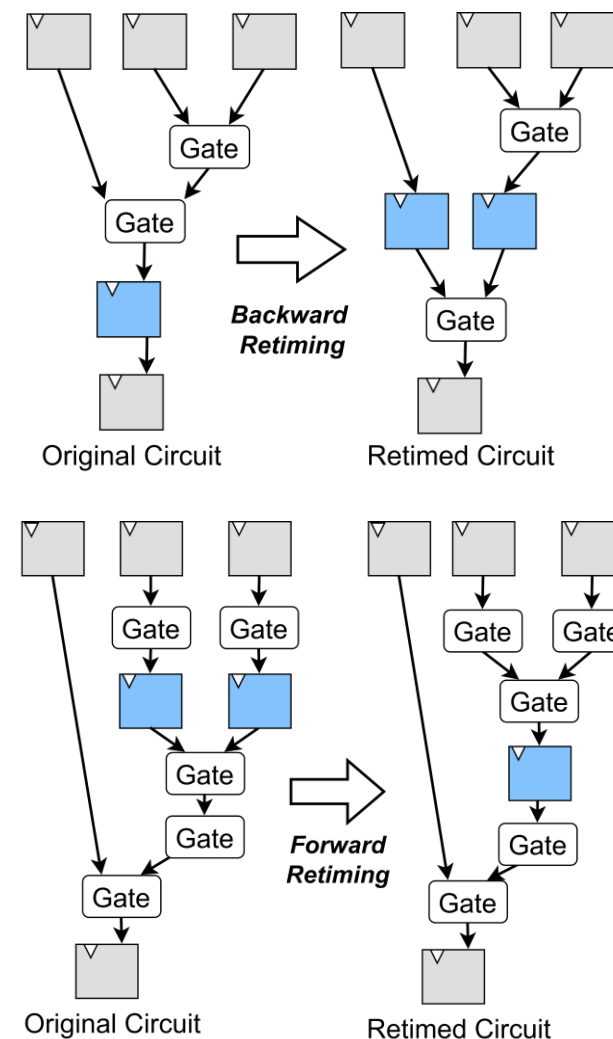
GIN: K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in ICLR, 2019.

GraphSAGE: W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in NeurIPS, 2017.

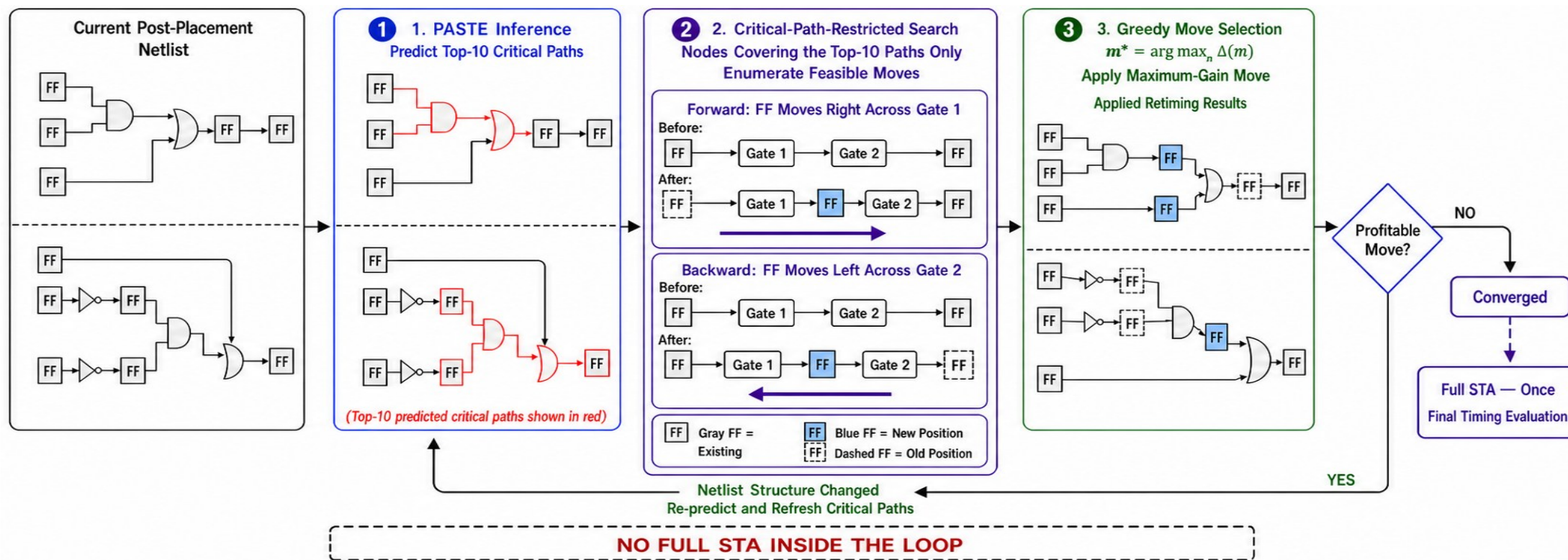
GAT: P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in ICLR, 2018.

Application — Predictor-guided Retiming

- **Traditional ABC retiming: unit-delay, decisions by logic depth**
 - efficient heuristic search
 - FPGA critical paths depend on placement, interconnect, routing
 - logic-critical $\neq$ physical-critical $\rightarrow$ suboptimal moves
- **Prior physical-aware retiming [Zhu et al., 2023]**
 - iterative, verifiable retiming pulls critical-path info from real STA reports
 - but depends on downstream STA each step (costly)
- **PASTE: embed a learned, lightweight timing oracle into the loop**
 - physical-aware critical paths at the post-placement stage
 - no real STA per step



Application — Retiming Algorithm



Key Design Point:

- The expensive full STA runs only once, at convergence, for final evaluation. Runtime is dominated by placement.
- We keep the efficiency of heuristic retiming but replace the crude unit-delay assumption with learned physical timing.

Application — Result



復旦大學

TABLE VIII: Performance Comparison of Our Method against Baseline and ABC Retiming on the Test Dataset.

Benchmark	Baseline			ABC Retiming				Our method				
	#Node	#Level	$F_{\max}$ (MHz)	#Node	#Level	$F_{\max}$ (MHz)	Impr.	#Node	#Level	$F_{\max}$ (MHz)	Impr.	
bgm	76213	95	58.41	-	-	-	-	76673	87	60.07	2.85%	
blob_merge	40081	102	93.29	40081	102	96.46	3.41%	40089	95	104.34	11.84%	
boundtop	376	9	456.73	376	9	446.63	-2.21%	378	9	455.15	-0.34%	
ch_intrinsics	346	15	412.43	346	15	452.85	9.80%	346	12	437.29	6.03%	
diffeq1	2749	77	40.20	2749	75	124.19	208.96%	2749	75	124.19	208.96%	
diffeq2	2196	75	54.73	2203	65	135.45	147.47%	2198	69	116.69	113.19%	
mcml	407351	397	14.46	-	-	-	-	407397	397	20.09	38.88%	
mkDelayWorker32B	1362	21	125.32	1362	21	76.41	-39.03%	1362	21	76.41	-39.03%	
mkPktMerge	550	5	226.65	550	5	313.09	38.14%	550	5	285.05	25.76%	
mkSMAdapter4B	4890	35	166.94	4885	35	211.73	26.84%	4898	35	223.12	33.65%	
or1200	12833	148	68.46	12878	143	76.98	12.44%	12845	148	73.92	7.97%	
raygentop	4321	41	165.49	-	-	-	-	4325	30	254.79	53.96%	
sha	11793	205	70.21	11835	98	75.77	7.90%	11815	94	76.73	9.28%	
s38417	13870	28	121.34	13885	37	107.42	-11.47%	13885	29	130.64	7.67%	
frisc	8011	66	75.08	8011	66	72.63	-3.26%	8015	61	85.92	14.44%	
dsip	3176	10	345.68	3176	10	362.09	4.75%	3176	8	424.67	22.85%	
tseng	2501	45	147.46	2501	45	139.82	-5.18%	2506	39	156.76	6.30%	
Average	-			-				28.47%	-			30.83%

- **PASTE: A physical-aware surrogate for FPGA pre-routing timing**
 - **Physical-aware features:**
 - design-aware RRG pruning
 - routing-density
 - timing-prior features
 - **Two-stage GNN aligned with STA's max-plus semantics**
 - heterogeneous message passing for local dependencies
 - edge-gated max propagation for long-range accumulation
- **High accuracy**
 - Node-level MAPE: **8.48%**; graph-level CPD MAPE: **8.28%**
 - sub-second inference on designs over 400k nodes
- **Downstream Validation**
 - PASTE-guided retiming improves Fmax by **30.83%**



復旦大學
FUDAN UNIVERSITY

Q & A