IEEE/ACM
2025 INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN
ICCAD
44th Edition

NUA-Timer: Pre-Synthesis Timing Prediction Under Non-Uniform Input Arrival Times

Ziyi Wang[1], Fangzhou Liu[1], Tsung-Yi Ho[1], David Pan[2], Bei Yu[1]

[1] The Chinese University of Hong Kong
[2] University of Texas at Austin

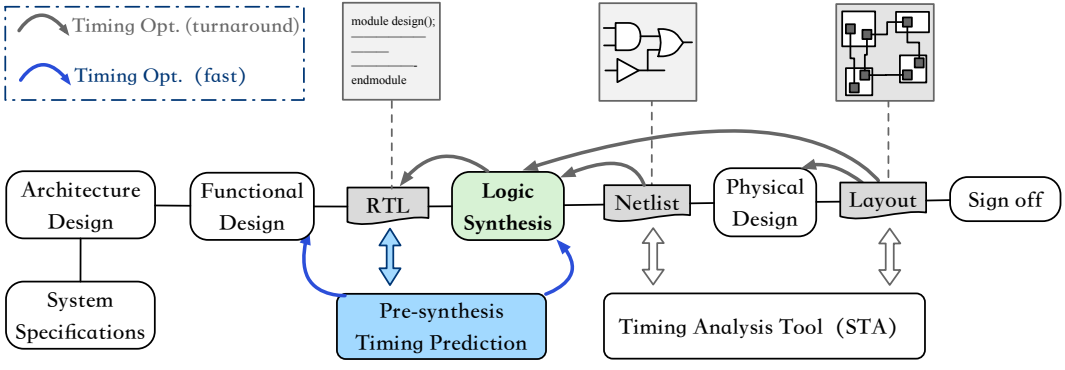
Oct. 28, 2025

Outline

Introduction

Background I: Pre-synthesis Timing Prediction

- Logic Synthesis: transforms RTL codes into gate-level netlists
- Pre-synthesis Timing Prediction: **enable early timing optimization** and reduce TAT



Challenge I: Gap Between Pre/Post-synthesis Structure

- The key to optimizing timing is **fine-grained estimation**
  - ✔ Answer 'where is it slow' besides 'how slow is it'
  - ✔ Identify critical paths besides predict endpoint arrival time
- The structural gap prevents the annotation of critical paths.

.

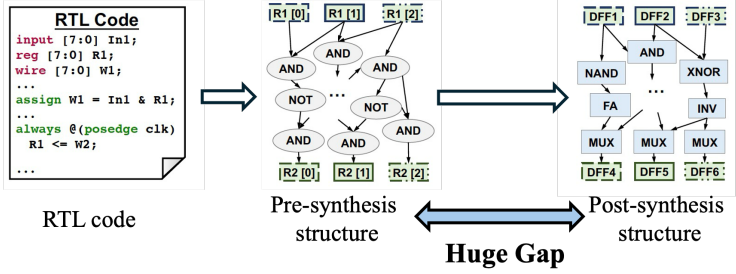


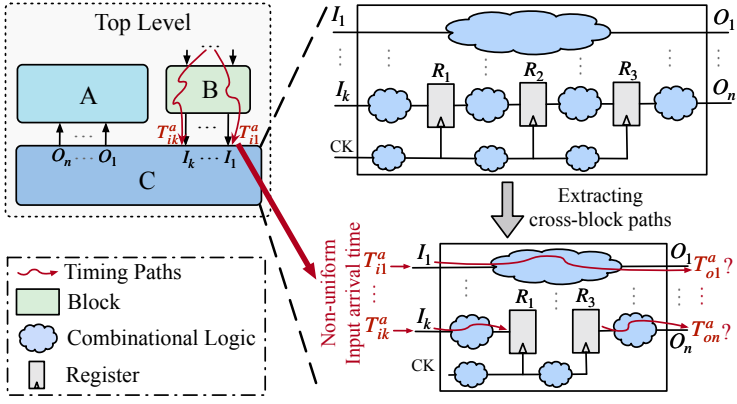
Figure source:[1]

[1] **Wenji Fang, Shang Liu, et al. (2024). "Annotating slack directly on your verilog: Fine-grained rtl timing evaluation for early optimization".** In: *Proc. DAC*, pp. 1–6.

Background II: Hierarchical Timing Analysis

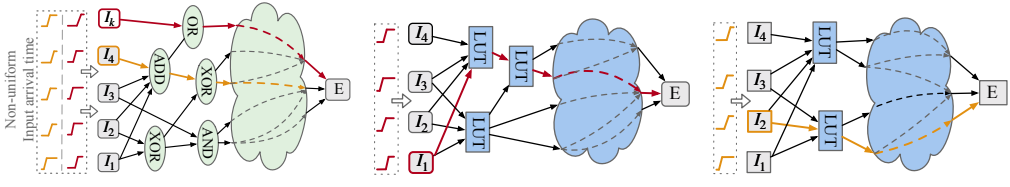
A circuit is segmented into multiple blocks at various levels

- Cross-block timing paths
- Non-uniform input arrival times



Challenge II: Non-uniform Input Arrival Times (NUIAT)

- **Extra Uncertainty**: Post-synthesis netlists/critical paths vary with different NUIATs.
- **Challenge**: Capturing long-range timing dependencies under different NUIATs
  - ✖ Existing methods lack this ability

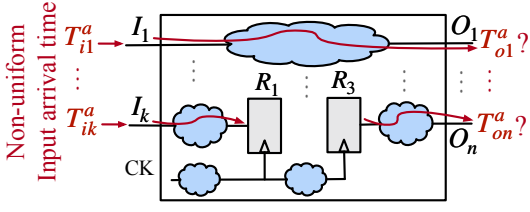


Here, the input I_k is a don't care term of endpoint E. (a) highlights pre-synthesis critical paths, each color per a specific NUIAT scenario. (b) and (c) display the resulting post-synthesis netlists and critical paths under two distinct NUIAT conditions.

Problem Definition

Pre-synthesis Timing Prediction under NUIAT

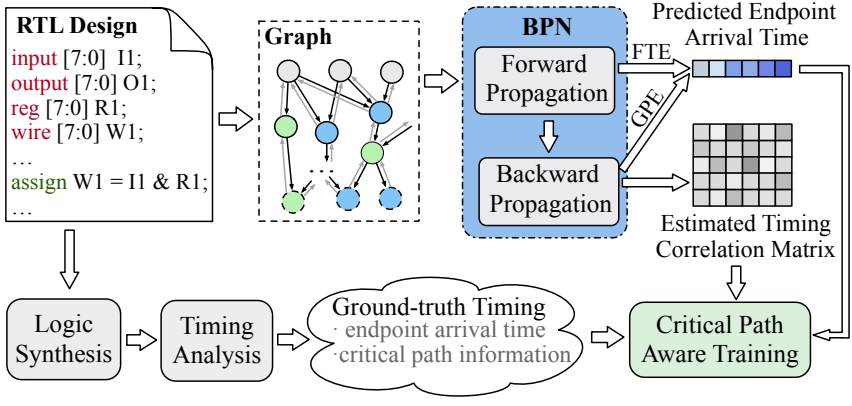
- **Setting**: Given a RTL block with NUIAT, we target the **cross-block** timing paths.
- **Task**: 1) Predict post-synthesis endpoint arrival time
  2) Identify critical paths.



Proposed Algorithm

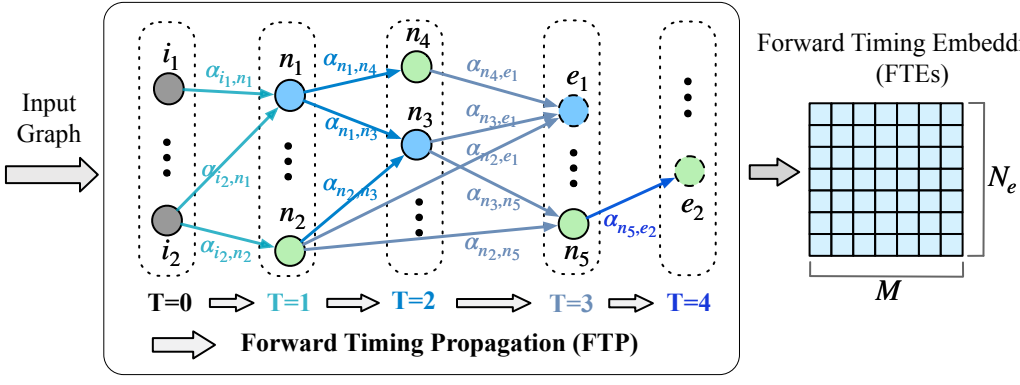
Overview of NUA-Timer

- Bidirectional Propagation Neural Network (BPN): modeling
- Timing Correlation Matrix: quantification
- Critical Path Aware Training: alignment



Forward Timing Propagation: Short-range

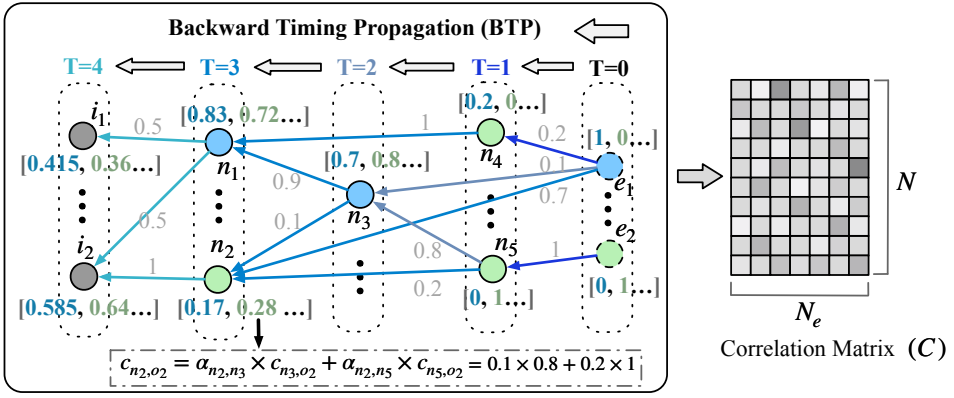
- Propagate delay (arrival time) from PIs to endpoints in the topological order.
- Use attention to capture **local timing dependencies** with immediate predecessors.
  - adaptive to different NUIATs.



Backward Timing Propagation: Long-range

- Operate a batch of endpoints each time
- Use **correlation matrix** to capture **long-range dependencies**
  - Initialization: one-hot vector for endpoints, all zero for others
  - Update at each topological level:

$$\boldsymbol{C}_i = \sum_{v_j \in \mathcal{N}^r(v_i)} \alpha_{v_i,v_j} \times \boldsymbol{C}_j. \tag{1}$$



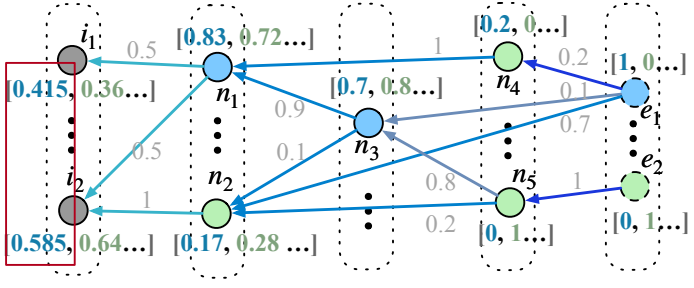
Property: Normalized Input Correlation Scores

Normalized Input Correlation Scores

For any endpoint, the sum of the correlation scores for all associated inputs equals one.

Consider the backward propagation from an endpoint as a random walk.

- Attention scores: transition probabilities.



Global Path Embedding

Correlation matrix shares similarities with **self-attention** in Graph Transformers

- Each element $\boldsymbol{C}_{i,j}$, ranging from 0 to 1, quantifies the probability of the i-th node being part of the critical path(s) for the j-th endpoint.

Global path embedding:

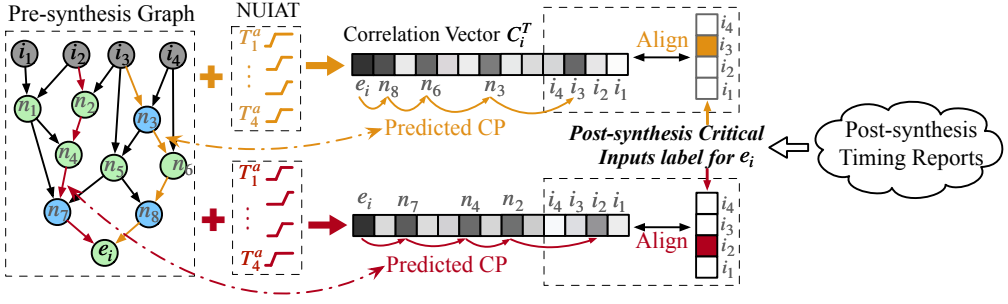
$$h^g_{e_j} = \sigma\Big(\sum_{v_k \in \mathcal{V}} \frac{\boldsymbol{C}_{k,j}}{\sum_{v_k \in \mathcal{V}} \boldsymbol{C}_{k,j}} \times h^f_{v_k}\Big), \tag{2}$$

where $\mathcal{V}$ represents the set of all nodes, and h_{v_k} denotes the embedding of v_k

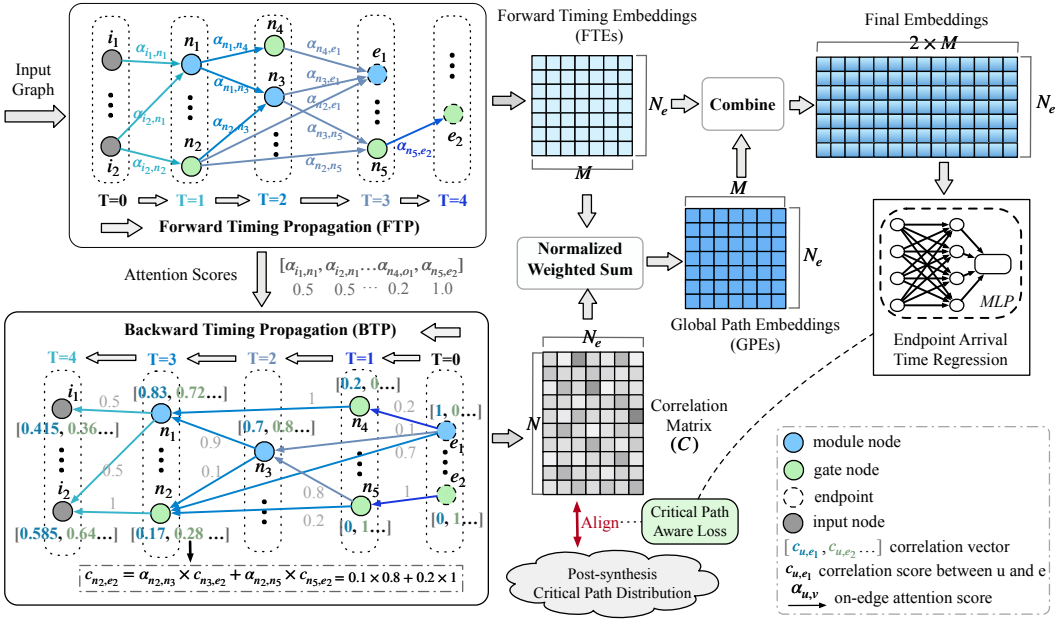
Critical Path Aware Loss (CPL)

Align correlation matrix with post-synthesis timing dependencies

- Pairwise (input-endpoint) post-synthesis critical paths labeling
- Align predicted critical input distribution with post-synthesis critical input labels



Overview


Input Graph
T=0
T=1
T=2
T=3
T=4
Forward Timing Propagation (FTP)
Attention Scores
$[\alpha_{i_1,n_1}, \alpha_{i_2,n_1} \ldots \alpha_{n_4,o_1}, \alpha_{n_5,e_2}]$
0.5 0.5 ⋯ 0.2 1.0
Backward Timing Propagation (BTP)
[0.83, 0.72…]
[0.2, 0…]
[0.415, 0.36…]
[0.7, 0.8…]
[1, 0…]
[0.585, 0.64…]
[0.17, 0.28 …]
[0, 1…]
$c_{n_2,e_2} = \alpha_{n_2,n_3} \times c_{n_3,e_2} + \alpha_{n_2,n_5} \times c_{n_5,e_2} = 0.1 \times 0.8 + 0.2 \times 1$
Forward Timing Embeddings (FTEs)
Combine
Final Embeddings
$2 \times M$
Normalized Weighted Sum
Global Path Embeddings (GPEs)
MLP
Endpoint Arrival Time Regression
Correlation Matrix (C)
Align
Critical Path Aware Loss
Post-synthesis Critical Path Distribution
module node
gate node
endpoint
input node
$[c_{u,e_1}, c_{u,e_2} \ldots]$ correlation vector
c_{u,e_1} correlation score between u and e
$\alpha_{u,v}$ on-edge attention score

Experimental Results

Experimental Setting

Table: The statistics of the synthetic dataset

Group	#Cases	#K Nodes		# Endpoints	
		range	average	range	average
SYN1	300×100	1∼20	8	10∼150	35
SYN2	100×100	20-80	40	30∼350	150

- Labels from Anlogic's commercial flow; Each sampled **100** sets of input arrival times
- Train on synthetic designs; Test on unseen open-source designs from OpenCores
- Baselines include 1) **graph-based** methods: ACCNN[2], GraphGPS[3]; and 2) **path-based** methods: MasterRTL[4], RTLtimer[5]

[2] **Haisheng Zheng et al. (2024). "LSTP: A Logic Synthesis Timing Predictor".** In: *Proc. ASPDAC*, pp. 728–733.

[3] **Ladislav Rampášek et al. (2022). "Recipe for a general, powerful, scalable graph transformer".** In: *Proc. NeurIPS* 35, pp. 14501–14515.

[4] **Wenji Fang, Yao Lu, et al. (2023). "MasterRTL: A pre-synthesis PPA estimation framework for any RTL design".** In: *Proc. ICCAD*, pp. 1–9.

[5] **Wenji Fang, Shang Liu, et al. (2024). "Annotating slack directly on your verilog: Fine-grained rtl timing evaluation for early optimization".** In: *Proc. DAC*, pp. 1–6.

Results

Our method can identify 80% critical start-endpoint pairs.

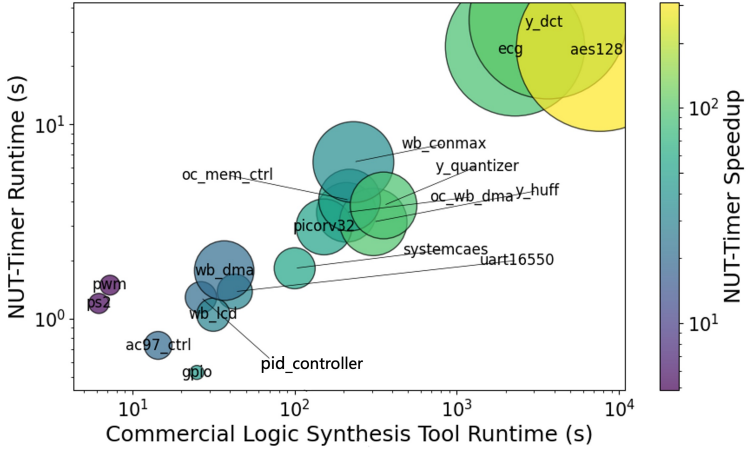
Table: Results on endpoint arrival time prediction.

Benchmark			ACCNN		GraphGPS		MasterRTL		RTLtimer		NUA-Timer	
Designs	#K nodes	#K ep	R^2	MAPE	R^2	MAPE	R^2	MAPE	R^2	MAPE	R^2	MAPE
gpio	1.8	0.4	0.267	14.2	-0.108	19.8	0.130	16.7	0.577	11	**0.901**	**5.6**
pwm	3.6	0.2	0.680	10.5	-0.414	24.9	0.765	10.5	0.764	9.6	**0.844**	**7.7**
ps2	3.7	0.2	0.553	17	-0.061	30.2	0.293	21.1	0.719	13.3	**0.856**	**9.8**
ac97_ctrl	7.1	0.4	0.640	14.7	0.000	29	0.020	24.5	0.503	13.7	**0.839**	**8.4**
pid_controller	9	0.4	0.471	11.2	-0.700	23.1	0.505	12.5	0.784	7.9	**0.827**	**6.7**
wb_lcd	10.1	0.6	0.653	9.4	-0.262	21.4	-0.010	19.4	0.680	9.7	**0.914**	**5**
uart16550	11.3	0.6	0.390	9	-0.737	19	0.032	14.4	0.296	10.9	**0.859**	**4.9**
systemcaes	15.4	0.7	0.234	12.6	-1.181	28.6	0.620	11.5	**0.748**	**8.7**	0.682	9.9
picorv32	28.6	1.6	0.736	6.3	-2.147	22.5	0.367	10.1	0.485	9.2	**0.746**	**6**
wb_dma	32.5	0.7	0.115	21.8	-0.405	31.7	0.717	12.5	0.670	11.4	**0.916**	**5.9**
oc_wb_dma	32.5	2.0	0.046	11.6	-1.467	28.9	0.649	11.1	0.712	8.6	**0.911**	**4.9**
oc_mem_ctrl	34.9	1.8	-0.718	24.1	-1.117	34.4	0.513	18.2	0.413	23.2	**0.718**	**12**
y_quantizer	40.7	2.8	0.585	18	-0.036	32.8	0.227	19.3	-0.435	27.4	**0.656**	**14.8**
y_huff	41.6	2.4	0.578	13.4	-0.009	22.1	-0.206	21.8	-0.307	20.2	**0.785**	**9.2**
wb_conmax	59.8	2.0	0.239	11.1	-1.489	21.3	0.557	9.8	0.838	5.4	**0.847**	**4.8**
ecg	177.7	7.3	-0.001	13.8	-0.057	21.4	0.182	16.5	0.081	16.9	**0.831**	**6.4**
y_dct	227.6	5.3	-0.256	15.3	-0.406	23	0.694	12	0.497	17	**0.732**	**9.3**
aes128	256.5	10.7	0.518	22.1	-0.137	43.9	0.528	16.8	0.114	25.1	**0.921**	**6.8**
AVG.	55.2	2.2	0.318	14.2	-0.596	26.6	0.366	15.5	0.452	13.8	**0.821**	**7.67**

Runtime Analysis

Evaluated on the same CPU machine.

- 60x on average, up to 300x (aes128)
- larger speedup on larger designs



Q&A