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

Disentangle, Align and Generalize: Learning A Timing Predictor from Different Technology Nodes

Xinyun Zhang*, Binwu Zhu*, Fangzhou Liu, Ziyi Wang, Peng Xu, Hong Xu, Bei Yu

Department of Computer Science & Engineering
Chinese University of Hong Kong

Outline

Introduction

Pre-routing Timing Prediction

- Pre-routing timing prediction method directly estimates the timing information without the need for time-consuming routing.
- This ”look-ahead” mechanism provides preliminary feedback for timing optimization, potentially expediting the chip design process.

Typical Pre-routing Timing Prediction Methods

- **Before the machine learning era**, we can use traditional timing prediction model, e.g., Elmore's model[1], with only placement results and its accuracy is unsatisfactory due to the absence of routing information.
- **With the development of machine learning**, many methods[2][3] rely on deep neural networks to extract timing path features and make predictions.

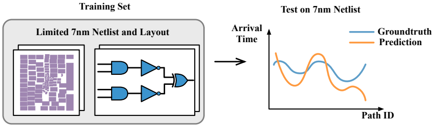
[1] Jorge Rubinstein, Paul Penfield, and Mark A Horowitz (1983). "Signal delay in RC tree networks". In: *TCAD*.

[2] Zizheng Guo et al. (2022). "A timing engine inspired graph neural network model for pre-routing slack prediction". In: *DAC*.

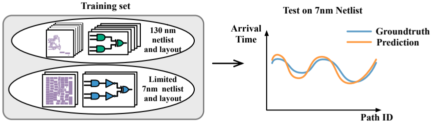
[3] Ziyi Wang et al. (2023). "Restructure-Tolerant Timing Prediction via Multimodal Fusion". In: *DAC*.

Switching to New Technology Node

However, collecting many data from the advanced node will be time-consuming. Training with limited data results in poor performance. To solve this issue, we propose a transfer learning framework that leverages data from **previous node** to enhance the performance for the target node.



(a)



(b)

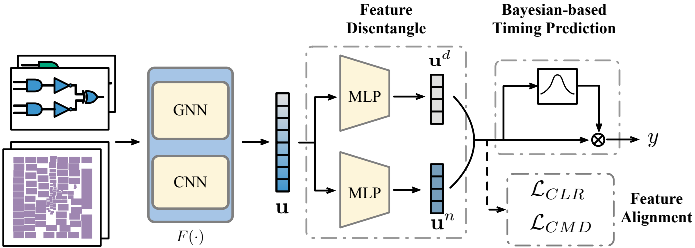
(a) Trained on limited 7nm netlist data; (b) Trained on both limited 7nm netlist data and 130nm netlist data.

Potential Challenges

- Netlist data consists of two kinds of knowledge: **node-dependent** and **design-dependent**. These two kinds of knowledge are highly intertwined in the netlist graph, making it difficult to leverage the common and transferable parts across different nodes.
- The arrival time values of different timing paths can vary dramatically, even by one or two orders of magnitude, which poses significant challenges for the ML-based regression model.
- The limited target node data makes the timing predictor susceptible to overfitting the training designs, which hinders the broad application of the learned model.

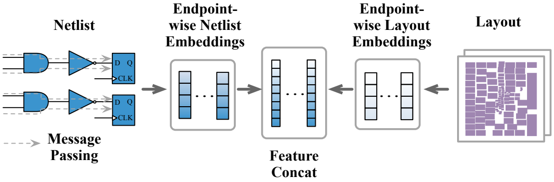
Method

Overall Architecture



- Timing Path Feature Extractor
- Timing Feature Disentanglement and Alignment
- Bayesian-based Timing Prediction

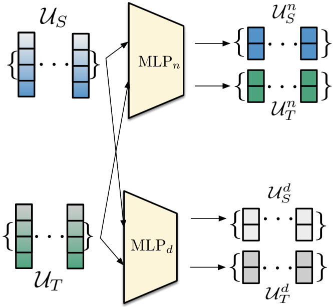
Multimodal Timing Path Feature Extractor



- Inspired by Wang *et. al.*[4], we first collect two types of input: the netlist graph $\mathcal{H}$ and the layout image set $\mathcal{X}$.
- The netlist $\mathcal{H}$ is constructed as a heterogeneous graph with two types of edges: the net edge connecting a net's drive pin and one of its sink pins, and the cell edge connecting one of a cell's input pins and its output pin.
- We first use a GNN and CNN to extract features from netlist and layout, respectively, and then concatenate them.

[4] Ziyi Wang et al. (2023). "Restructure-Tolerant Timing Prediction via Multimodal Fusion". In: *DAC*.

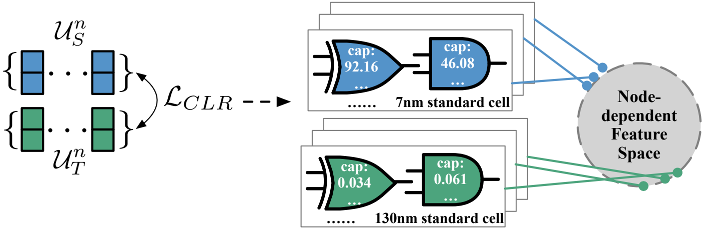
Disentangle Intertwined Features



- Each netlist contains two parts of information: the functionality information encoded in the design specification and the standard cell information.
- For any path feature $\boldsymbol{u}$, we further adopt two MLPs to disentangle the equal-sized node-dependent features $\boldsymbol{u}^n$ and design-dependent features $\boldsymbol{u}^d$ by:

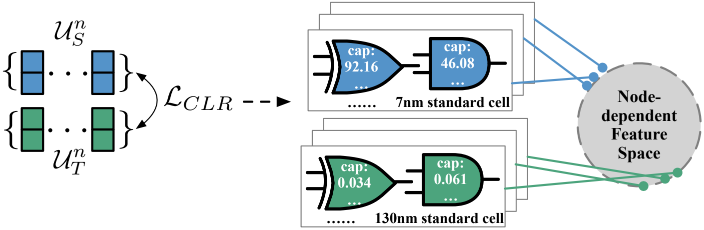
$$\boldsymbol{u}^n = \mathrm{MLP}_n(\boldsymbol{u}) \in \mathbb{R}^{m/2}, \quad \boldsymbol{u}^d = \mathrm{MLP}_d(\boldsymbol{u}) \in \mathbb{R}^{m/2}.$$

Align Node-based Features



- **Motivation:** The netlist on the same node should share the same standard cells, including the gate structures and their characteristics. On the other hand, the node-dependent features should be distinguishable for netlists in different nodes.

Node-based Contrastive Loss

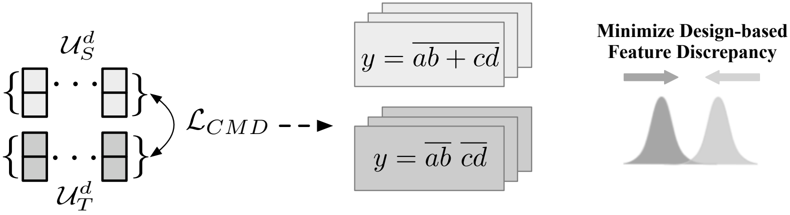


- Denote the set of all the node-dependent features as $\mathcal{A} = \mathcal{U}_S^n \cup \mathcal{U}_T^n$. Given any node-dependent feature set $\mathcal{U}^n$ ($\mathcal{U}_S^n$ or $\mathcal{U}_T^n$), the contrastive loss for the feature set can be defined as:

$$\mathcal{L}_{Set}(\mathcal{U}^n) = \sum_{\boldsymbol{u}\in\mathcal{U}^n} \frac{-1}{|\mathcal{U}^n| - 1} \sum_{\boldsymbol{m}\in\mathcal{U}^n\setminus\{\boldsymbol{u}\}} \frac{\exp(\boldsymbol{u}\cdot\boldsymbol{m}/\tau)}{\sum_{\boldsymbol{a}\in\mathcal{A}\setminus\{\boldsymbol{u}\}} \exp(\boldsymbol{u}\cdot\boldsymbol{a}/\tau)},$$

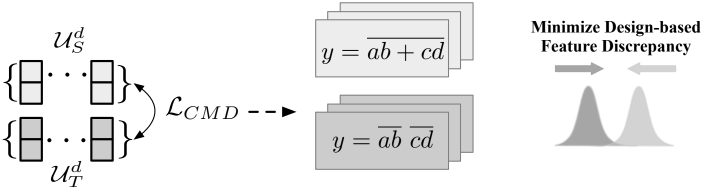
$$\mathcal{L}_{CLR} = \frac{1}{|\mathcal{U}_S^n|}\mathcal{L}_{Set}(\mathcal{U}_S^n) + \frac{1}{|\mathcal{U}_T^n|}\mathcal{L}_{Set}(\mathcal{U}_T^n).$$

Align Design-based Features



- **Motivation:** The design-dependent features represent the abstract logical functionality of each netlist. For each design, we can opt for different technology nodes for synthesis.

Design-based Discrepancy Loss

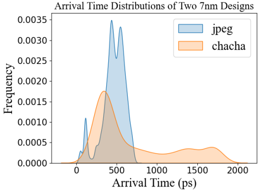


- To align the design-dependent features, we optimize the Central Moment Discrepancy (CMD) between the feature sets from different nodes, which can be formulated as:

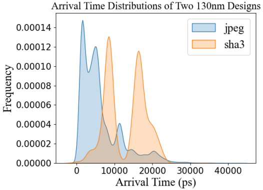
$$\mathcal{L}_{CMD}(\mathcal{U}_S^d, \mathcal{U}_T^d) = \frac{1}{b-a}\left\|\mathbb{E}(\mathcal{U}_S^d) - \mathbb{E}(\mathcal{U}_T^d)\right\| + \sum_{k=2}^{\infty} \frac{1}{|b-a|^k}\left\|c_k(\mathcal{U}_S^d) - c_k(\mathcal{U}_T^d)\right\|,$$

where $[a, b]$ is the interval that bounds $\mathcal{U}_S^d$ and $\mathcal{U}_T^d$, $\mathbb{E}(\cdot)$ denotes the expectation and $c_k(\cdot)$ is the k-th order moment.

Bayesian-based Timing Prediction



(a)



(b)

- To predict highly variable timing information and prevent overfitting, we propose a Bayesian-based timing prediction model that can be formulated as:

$$\log p(y|\mathcal{G}', \mathcal{N}) = \log \int p(y|\mathcal{G}', \boldsymbol{W})p(\boldsymbol{W}|\mathcal{N})d\boldsymbol{W},$$

where $\mathcal{N}$ represents the overall distribution for all the timing paths and $\boldsymbol{W}$ denotes the model parameters.

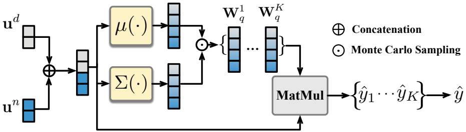
Evidence Lower Bound

- With a variational posterior distribution $q(\boldsymbol{W}|\mathcal{G}')$ only conditioning on single timing path input, we can derive the evidence lower bound (ELBO) by:

$$
\begin{aligned}
\log p(y|\mathcal{G}', \mathcal{N}) &= \log \int p(y|\mathcal{G}', \boldsymbol{W}) p(\boldsymbol{W}|\mathcal{N}) d\boldsymbol{W} \\
&= \log \int p(y|\mathcal{G}', \boldsymbol{W}) \frac{p(\boldsymbol{W}|\mathcal{N})}{q(\boldsymbol{W}|\mathcal{G}')} q(\boldsymbol{W}|\mathcal{G}') d\boldsymbol{W} \\
&= \log \mathbb{E}_q \left[p(y|\mathcal{G}', \boldsymbol{W}) \frac{p(\boldsymbol{W}|\mathcal{N}))}{q(\boldsymbol{W}|\mathcal{G}')} \right] \\
&\geq \mathbb{E}_q \left[\log p(y|\mathcal{G}', \boldsymbol{W}) \right] - \mathrm{KL}(q(\boldsymbol{W}|\mathcal{G}') || p(\boldsymbol{W}|\mathcal{N}))).
\end{aligned}
$$

where the first term is the log-likelihood with the variational posterior, while the second term is the KL divergence between the variational posterior and the prior distribution.

Approximated by Gaussian Distribution



- We use Gaussian distribution to approximate the variational posterior:

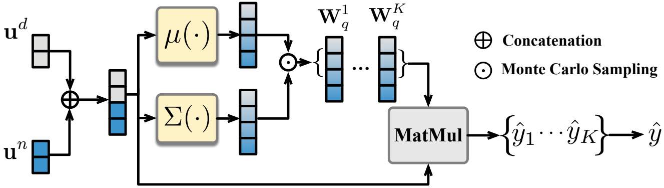
$$\boldsymbol{W}_q \sim N(\mu([\boldsymbol{u}^n, \boldsymbol{u}^d]), \Sigma([\boldsymbol{u}^n, \boldsymbol{u}^d])).$$

- In addition, the prior distribution is modeled as:

$$\boldsymbol{W}_p \sim N(\mu(\tilde{\boldsymbol{u}}(\mathcal{N})), \Sigma(\tilde{\boldsymbol{u}}(\mathcal{N}))),$$

where $\tilde{\boldsymbol{u}}(\mathcal{N}) \in \mathbb{R}^m$ is a dummy timing path feature that represents the true distribution of all the timing paths within the whole technology node $\mathcal{N}$.

Training Objective



- To construct a representative $\tilde{\boldsymbol{u}}$, we can simply use the mean of all the node-dependent features to represent the node information.
- For the design-related information, we can collect all the design-dependent features in both the source preceding technology node and the target advanced technology node.
- The final ELBO objective is formulated as:

$$\mathcal{L}_{ELBO}(y, \mathcal{G}', \mathcal{N}) = \frac{1}{K}\sum_{i=1}^{K} \log p(y|\mathcal{G}', \mathbf{W}_q^i) - \mathrm{KL}(q(\mathbf{W}|\mathcal{G}')||p(\mathbf{W}|\mathcal{N})).$$

Experimental Results

Benchmark Statistics

Table: Statistics of the dataset (edp stands for endpoint, e_n and e_c denote net edge and cell edge, respectively)

Benchmark		tech node	#pin	#edp	$\#e_n$	$\#e_c$
		Input information				
train	smallboom	7*nm*	694441	61764	488052	423344
	jpeg	130*nm*	1527166	39783	1150173	749294
	linkruncca	130*nm*	186546	17796	151617	84393
	spiMaster	130*nm*	99507	4739	75718	44917
	usbf_device	130*nm*	48104	4777	37557	21706
test	arm9	7*nm*	44469	2500	33065	29287
	chacha	7*nm*	35687	1986	25117	23083
	hwacha	7*nm*	1357798	61313	985057	922085
	or1200	7*nm*	1165114	172401	844443	658961
	sha3	7*nm*	794720	60323	552021	485596
Avg	train	7*nm*&130*nm*	511153	25772	380623	264731
	test	7*nm*	679558	59705	487941	423802

- Four 130*nm* netlists and one 7*nm* netlist for training, and five 7*nm* netlists for test.
- All designs are from Freecores and Chipyard.
- Cadence Genus for synthesis and Cadence Innovus for placement, timing optimization, routing and static timing analysis.

Comparison with SOTA timing prediction works

Table: The evaluation results on 7*nm* netlist data.

Baseline	DAC23-AdvOnly		DAC23-SimpleMerge		DAC23-ParamShare		DAC23-PT-FT		Ours	
	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime	R^2 score	runtime
arm9	0.603	2.546	-2.069	2.546	0.567	2.546	0.837	2.546	**0.864**	2.621
chacha	0.624	1.188	-1.983	1.188	0.568	1.188	0.726	1.188	**0.890**	1.234
hwacha	0.170	5.229	-2.203	5.229	0.499	5.229	0.818	5.229	**0.828**	5.400
or1200	0.156	14.257	-6.037	14.257	0.240	14.257	0.209	14.257	**0.682**	14.793
sha3	0.425	1.690	-4.741	1.690	0.195	14.257	0.284	1.690	**0.785**	1.725
average	0.396	4.982	-3.407	4.982	0.414	4.982	0.575	4.982	**0.810**	5.154

- Our method outperforms all the baselines with a significant margin.
- Only training with limited 7nm data achieves poor performance, indicating the necessity of massive training data belonging to the same distribution as the test data.
- Our method is effective in handling the distribution shifts in the 130*nm* and 7*nm* data and is capable of transferring the knowledge to different technology nodes.

Ablation Studies

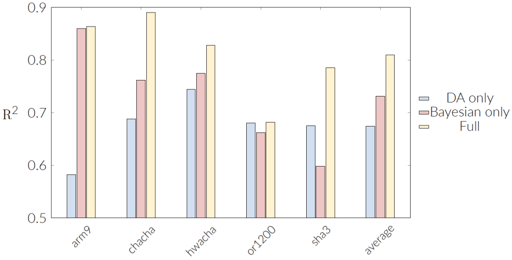


Table: Ablation study on the number of 130*nm* designs. J, L S, and U denote jpeg, linkruncca, spiMaster, and usbf_device, respectively.

J	L	S	U	arm9	chacha	hwacha	or1200	sha3	average
✓				0.496	0.394	0.649	0.363	0.631	0.507
✓	✓			0.312	0.773	0.470	0.616	0.599	0.554
✓	✓	✓		0.564	0.821	0.804	0.531	0.673	0.679
✓	✓	✓	✓	**0.864**	**0.890**	**0.828**	**0.682**	**0.785**	**0.810**

- Both modules are effective.
- These two modules lead to different improvements on different designs.

- When we increase the number of 130*nm* data, the timing prediction performance improves consistently.
- Our method is effective in transferring the knowledge in different nodes.

Conclusion

Conclusion

- We propose a novel transfer learning framework that leverages abundant data from previous technology node to enhance learning on the target technology node.
- Our method first disentangles the timing path features into node- and design-dependent parts and aligns them separately.
- We use a Bayesian machine learning-based model to predict the arrival time of each timing path, which can handle its high variability and generalize to new designs in the test set.

THE CHIPS TO SYSTEMS CONFERENCE

61

SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Thanks!