DATE 23
DESIGN, AUTOMATION & TEST IN EUROPE
17 – 19 April 2023 · Antwerp, Belgium
The European Event for Electronic System Design & Test

Fast and Accurate Wire Timing Estimation Based on Graph Learning

Yuyang Ye[1], Tinghuan Chen[2,3], Yifan Gao[1], Hao Yan[1], Bei Yu[2], Longxing Shi[1]

[1]Southeast University
[2]Chinese University of Hong Kong
[3]Chinese University of Hong Kong, Shenzhen

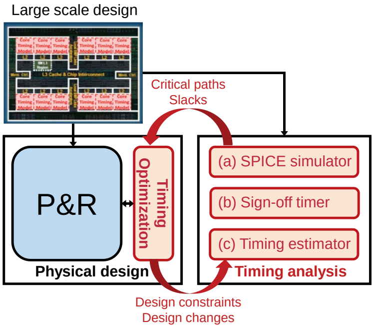
Apr. 18, 2023

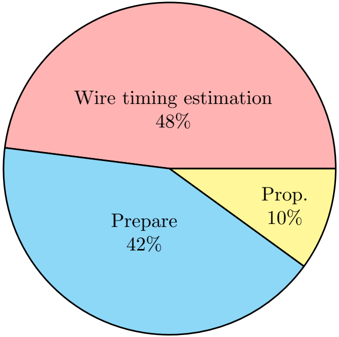
Introduction

Wire Timing Estimation: Fast? or Accurate?

As the design gets closer to tape-out, a more accurate wire timing estimation is required to guide timing optimization.

As designs become larger and larger, a faster wire timing estimation is a necessity to speed up STA.





Interaction between physical design and timing analysis [1].

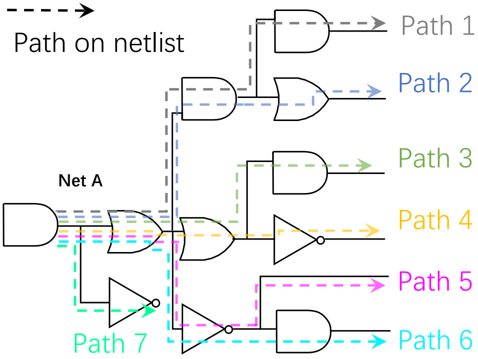
Runtime breakdown for Opentimer on a million-gate circuit using 40 CPUs [2].

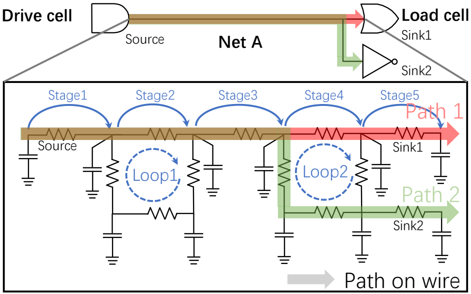
[1] Fast and Accurate Wire Timing Estimation on Tree and Non-Tree Net Structures

[2] GPU-Accelerated Static Timing Analysis

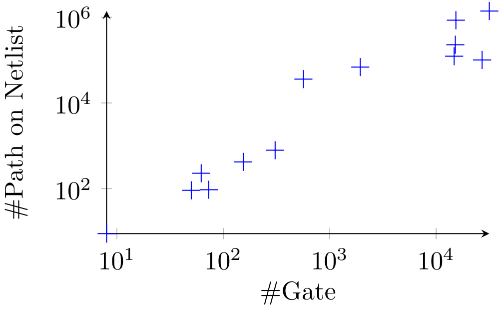
Introduction: Paths on the Netlist and Wire

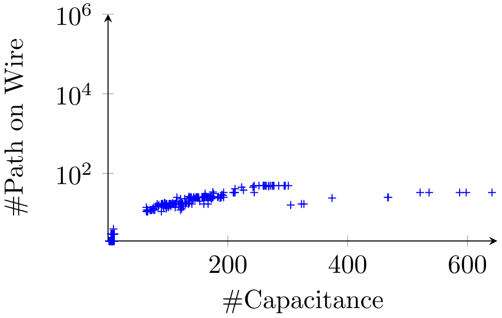
- An example of paths on netlists is shown at left. There are 7 paths on the netlist with 11 gates.
- An example of paths on nets is shown at right. There are 2 paths on the net with 11 capacitances.





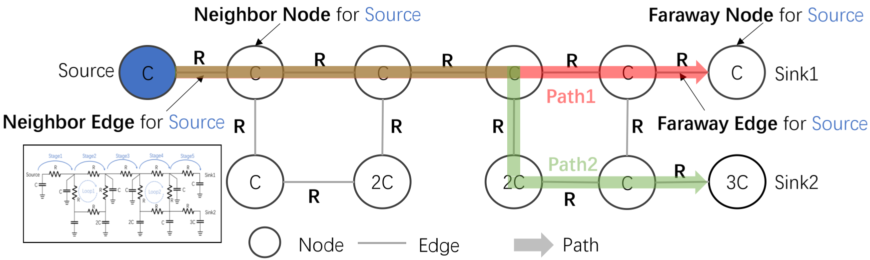
- The numbers of paths are more than 1 million with just 10k gates.
- The maximum path number of paths on these nets is just 49, and most of the nets are composed of 10-30 paths.
- Thus, The limited number of wire paths opens a door for the graph learning method to estimate wire timing effectively while considering path information.





RC Network from Graph View

- A complex RC net $\rightarrow$ RC graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$;
- A capacitance $\rightarrow$ A node v_i in $\mathcal{V}$;
- A resistance connected between node v_i and v_j $\rightarrow$ An edge e_{ij} in $\mathcal{E}$;
- A wire path from the path source *Source* to the target sink *Sink*1 $\rightarrow$ A sub-graph q in $\mathcal{P}$ consists of all nodes and edges visited



Problem Formulation

Firstly, we give some basic definitions:

- **Definition 1 (Wire Path):** The timing path of a wire, which is from the source to the target sink.
- **Definition 2 (Wire Slew):** The time required for a signal of high-to-low or low-to-high transition on a wire is captured from the signal waveform and defined as fall/rise slew.
- **Definition 3 (Wire delay):** The time required for a signal that propagates from the wire source to the target wire sink.

Then, **Problem 1 (Wire timing estimation)** is formulated as follows:

- Given an RC net with parasitics and net structure
- Capture the information of each path, each capacitance, each resistance, net structure and their relationships effectively and estimate the **wire slew** and **wire delay** of the **wire path** based on these information.

Proposed Method

Overall Flow

In our work, Problem 1 can be handled with two steps: In step 1, we propose a graph learning method GNNTrans, including three different modules to generate wire path representation for each wire path through collecting information of nets; In step2, we apply Multilayer Perceptronlayers (MLPs) based on the generated representations to fast and accurately estimate wire slew and wire delay.

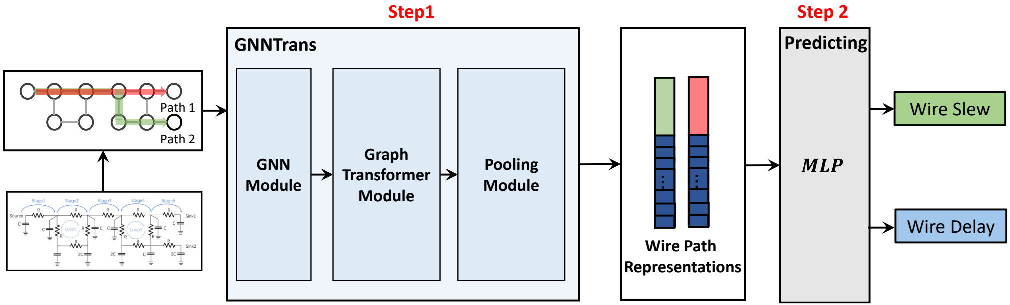
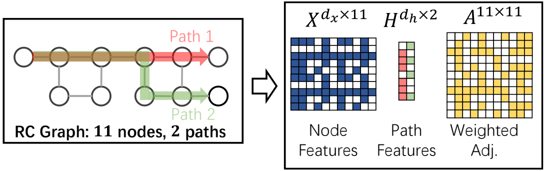


Table: Raw node and path features used.

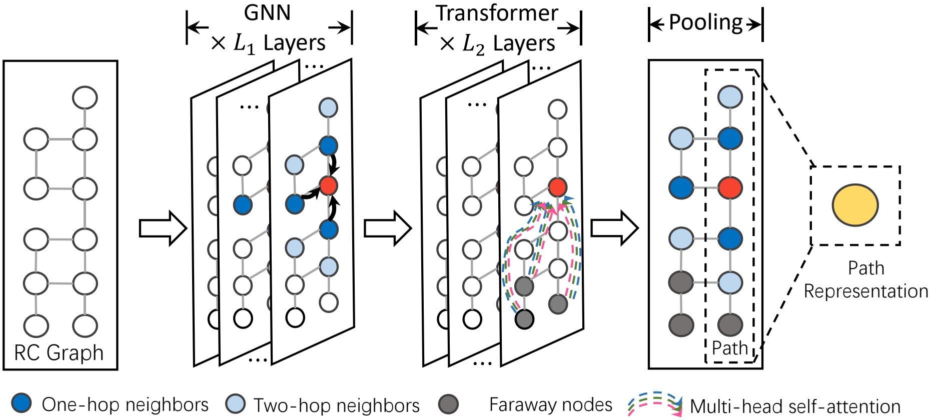
Type	Name	Description
Node	capacitance value	values of capacitance
	num of input nodes	number of input nodes
	num of output nodes	number of output nodes
	tot input cap	total input capacitance
	tot output cap	total output capacitance
	num of connect. res	number of connected resistance
	tot input res	total input resistance
	tot output res	total output resistance
	downstream cap	Elmore downstream capacitance
	stage delay	Elmore stage delay
Path	input slew	input transition time
	dir. of drive cell	drive strength of drive cell
	func. of drive cell	functionality of drive cell
	dir. of a load cell	drive strength of load cell
	func. of load cell	functionality of load cell
	ceff of load cell	effective capacitance of load cell
	Elmore delay	wire path Elmore delay
	D2M delay	wire path D2M delay

The RC net graph $\mathcal{G}$ is represented with:

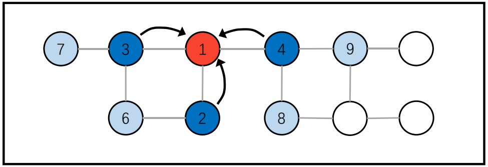
- Node feature matrix $\boldsymbol{X}$ for each capacitance
- Path feature matrix $\boldsymbol{H}$ for each wire path
- Weighted adj. matrix $\boldsymbol{A}$ for each resistance
- Label matrix for real wire slew and delay

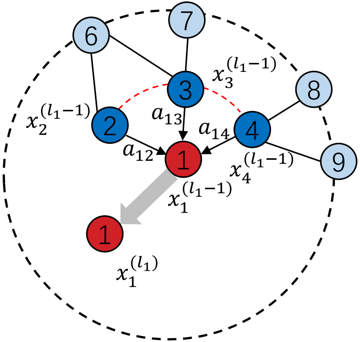


GNNTrans mainly consists of three modules: standard GNN, graph transformer and pooling. After GNNTrans, we can get path representations for each wire path.



To learn the RC net graph's local structural information, we update a node's representations by aggregating information from its neighbors with graph connectivity in GNN module. In Equation (1), it demonstrates the details where $\boldsymbol{W}_1^{(\ell_1)}$ and $\boldsymbol{W}_2^{(\ell_1)}$ denote the learnable matrices. ReLU is a nonlinear function.

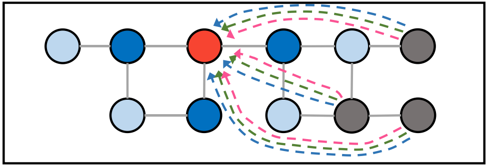


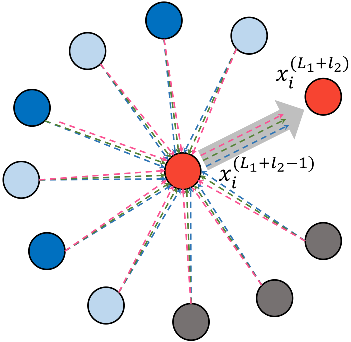


$$\boldsymbol{x}_i^{(\ell_1)} = \mathrm{ReLU}(\boldsymbol{W}_1^{(\ell_1)}\boldsymbol{x}_i^{(\ell_1-1)} + \boldsymbol{W}_2^{(\ell_1)} a_{iu} \sum_{u \in \mathcal{N}(v_i)} \boldsymbol{x}_u^{(\ell_1-1)}). \tag{1}$$

GNNTrans: Graph Transformer Module

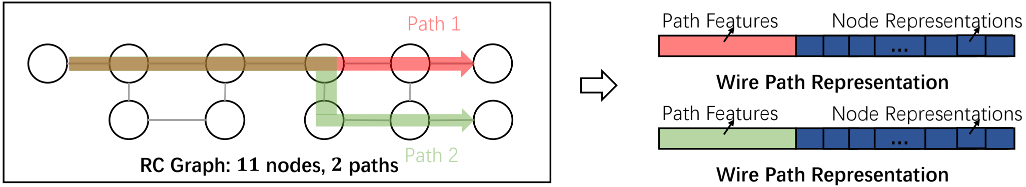
To learn the RC net graph's global net information (relationships between capacitances and resistances) without over-smoothing issues, we update a node's representations by a multi-head self-attention mechanism in graph transformer module. In Equation (2), it demonstrates the details where $\boldsymbol{W}_3^{(\ell_2)}$, $\boldsymbol{W}_V^{(k,\ell_2)}$ are learnable linear transformation matrices. $\|$ denotes concatenating operation.





$$\boldsymbol{x}_i^{(L_1+\ell_2)} = \boldsymbol{x}_i^{(L_1+\ell_2-1)} + \boldsymbol{W}_3^{(\ell_2)} \|_{k=1}^{\mathcal{K}} \sum_{u \in \mathcal{V}} \tilde{\boldsymbol{a}}_{iu}^{(k,\ell_2)} \left(\boldsymbol{W}_V^{(k,\ell_2)} \boldsymbol{x}_u^{(L_1+\ell_2-1)} \right). \tag{2}$$

To generate the wire path representations, we select and combine the node representations $\boldsymbol{X}^{(L_1+L_2)}$: $\{\boldsymbol{x}_i^{(L_1+L_2)}, \forall i \in \mathcal{V}\}$ after graph learning with original wire path features $\boldsymbol{H}$: $\{\boldsymbol{h}_q, \forall q \in \mathcal{P}\}$ in the pooling module. In Equation (3), it demonstrates the details where $\mathcal{V}_q$ is the node set of wire path q and N_q is the number of nodes on wire path q.



$$\boldsymbol{f}_q = \left(\frac{1}{N_q} \sum_{v_i \in \mathcal{V}_q} \boldsymbol{x}_i^{(L_1+L_2)}\right) \| \boldsymbol{h}_q. \tag{3}$$

Predicting and Path Delay Calculation

Based on the wire path representations $\boldsymbol{F}$: $\{\boldsymbol{f}_q, \forall q \in \mathcal{P}\}$, we use a multilayer perceptron layer *MLP* to predict the wire slew and delay under SI mode. Trainable parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ in the multilayer perceptron layer *MLP* are introduced.

$$S_{\mathrm{q}} = MLP(\boldsymbol{\theta} \mid \boldsymbol{f}_q), \tag{4}$$

$$D_{\mathrm{q}} = MLP(\boldsymbol{\phi} \mid \boldsymbol{f}_q, S_{\mathrm{q}}). \tag{5}$$

where S_{q} and D_{q} are the wire slew and delay estimation results of wire path q.

Results

Experimental Settings

- Synopsys StarRC extracts RC parasitics, and the golden timing report is generated by Synopsys PrimeTime SI mode with TSMC16nm technology.
- CPU Device: a 72-core 2.6GHz Linux machine with 1024 GB memory.
- GPU Device: 4 NVIDIA Tesla V100 GPUs.
- Benchmarks: 18 opencore circuits [1].

Table: Benchmark statistics.

	Benchmark	#Cells	#Nets (Non-tree)	#FFs	#CPs
Train	PCI_BRIDGE	1234	1598 (279)	310	456
	DMA	10215	10898 (1963)	1956	1475
	B19	33785	34399 (8906)	3420	5093
	SALSA	52895	57737 (16802)	7836	9648
	RocketCore	90859	93812 (38919)	16784	12475
	VGA_LCD	56194	56279 (20527)	17054	8761
	ECG	84127	85058 (31067)	14,018	13189
	TATE	184601	185379 (51037)	31,409	27931
	JPEG	219064	231934 (73915)	37,642	36489
	NETCARD	316137	317974 (76924)	87,317	46713
	LEON3MP	341000	341263 (81687)	108,724	50716
	Total	1390111	1075068 (402026)	326470	212766
Test	WB_DMA	40962	40664 (9493)	718	9619
	LDPC	39377	42018 (10257)	2048	7613
	DES_PERT	48289	48523 (9534)	2983	10976
	AES-128	113168	90905 (42657)	10686	24973
	TV_CORE	207414	189262 (53147)	40681	33706
	NOVA	141990	139224 (36482)	30494	39341
	OPENGFX	219064	231934 (62395)	37,642	47831
	Total	810264	782530 (223965)	125252	221890

[1]OpenCores, http://opencores.org

Wire Slew/Delay Estimation Accuracy of Non-tree Nets

Table: Estimation accuracy of non-tree nets (R^2 score).

Benchmark	Wire Slew/Delay Estimation Accuracy of Non-tree Nets (R^2 score)					
	DAC20	GCNII	GraphSage	GAT	Trans.	GNNTrans
WB_DMA	0.721/0.693	0.894/0.846	0.912/0.907	0.907/0.872	0.851/0.804	**0.987/0.979**
LDPC	0.714/0.705	0.871/0.829	0.904/0.893	0.881/0.872	0.817/0.781	**0.991/0.985**
DES_PERT	0.703/0.662	0.906/0.871	0.918/0.872	0.897/0.851	0.824/0.807	**0.984/0.975**
AES-128	0.684/0.651	0.824/0.819	0.846/0.829	0.832/0.824	0.807/0.791	**0.979/0.962**
TV_CORE	0.607/0.594	0.738/0.709	0.819/0.806	0.791/0.748	0.795/0.769	**0.969/0.957**
NOVA	0.664/0.631	0.795/0.781	0.834/0.829	0.819/0.802	0.783/0.774	**0.976/0.971**
OPENGFX	0.568/0.537	0.781/0.759	0.827/0.816	0.792/0.773	0.812/0.803	**0.962/0.959**
Average	0.666/0.639	0.830/0.802	0.866/0.850	0.845/0.820	0.813/0.790	**0.978/0.970**

- The average R^2 scores of GNNTrans reach 0.978 and 0.970, which outperforms GCNII by 0.148/0.168, GraphSage by 0.112/0.120, and GAT by 0.133/0.150.
- Compared Transformer, our method achieves gains of 0.165/0.180 on average.

Wire Slew/Delay Estimation Accuracy of All Nets

Table: Estimation accuracy of all nets (R^2 score)

Benchmark	Wire Slew/Delay Estimation Accuracy of All Nets (R^2 score)					
	DAC20	GCNII	GraphSage	GAT	Trans.	GNNTrans
WB_DMA	0.823/0.791	0.915/0.909	0.944/0.921	0.932/0.916	0.912/0.875	**0.999/0.994**
LDPC	0.815/0.797	0.908/0.863	0.925/0.917	0.913/0.907	0.862/0.859	**0.995/0.991**
DES_PERT	0.837/0.822	0.924/0.913	0.927/0.899	0.902/0.899	0.875/0.861	**0.997/0.990**
AES-128	0.802/0.760	0.879/0.867	0.883/0.872	0.845/0.824	0.867/0.854	**0.987/0.982**
TV_CORE	0.795/0.782	0.821/0.810	0.844/0.837	0.831/0.824	0.889/0.876	**0.989/0.986**
NOVA	0.783/0.710	0.854/0.847	0.872/0.865	0.845/0.831	0.876/0.871	**0.984/0.980**
OPENGFX	0.769/0.729	0.835/0.827	0.864/0.851	0.840/0.829	0.897/0.869	**0.982/0.979**
Average	0.803/0.770	0.877/0.862	0.894/0. 880	0.873/0.861	0.882/0.866	**0.990/0.986**

- Our method can achieve 0.990 and 0.986 accuracy on average in wire slew and delay estimation.

Path Delay Estimation Accuracy

Table: Path arrival time estimation accuracy, including R^2 score / MAE (ps), and runtime (s) comparison. "MAE" represents maximum absolute error. PlanA (L_1=25, L_2=5), PlanB (L_1=20, L_2=10), and PlanC (L_1=15, L_2=15) are GNNTrans with 3 different configurations, which helps test our work in different ways.

Benchmark	Path Delay Estimation Accuracy: R^2 score and MAE(ps)					Runtime(s)			
	PrimeTime	Piror Work	Our Work			STA-SI	Our Work		
	STA-SI	DAC20	PlanA	PlanB	PlanC	Full	Gate	Wire	Total
WB_DMA	1.0000/0.00	0.746/42.45	**0.999/0.57**	0.997/0.59	0.972/1.52	276.7	136.2	**25.1**	161.3
LDPC	1.0000/0.00	0.722/58.21	**0.998/0.64**	0.996/0.67	0.981/0.83	365.9	200.4	**32.9**	233.3
DES_PERT	1.0000/0.00	0.709/37.32	**0.999/0.43**	0.997/0.71	0.983/1.05	386.3	186.7	**27.4**	214.1
AES-128	1.0000/0.00	0.654/71.27	0.954/5.32	0.984/2.32	**0.990/1.14**	593.7	340.5	**56.7**	397.2
TV_CORE	1.0000/0.00	0.527/127.58	0.928/8.56	0.976/4.27	**0.981/3.94**	614.6	400.6	**60.2**	460.8
NOVA	1.0000/0.00	0.604/84.61	0.967/2.64	0.979/1.25	**0.985/0.91**	1133.8	491.2	**87.3**	578.5
OPENGFX	1.0000/0.00	0.574/100.67	0.931/6.18	0.969/3.68	**0.975/2.54**	1185.4	567.3	**97.6**	664.9
Average	1.0000/0.00	0.648/74.59	0.968/3.48	**0.985/1.93**	0.981/1.70	650.91	331.84	**55.31**	387.16

- The R^2 scores using different plans reach 0.968, 0.985, and 0.981 on average.
- The average MAEs using different plans are 3.48ps, 1.93ps and 1.70ps.
- The wire timing estimator costs 55.7s on average for different designs scaling from 40k to 200k nets.

Conclusion

Conclusion

- The limited number of wire paths opens a door for the graph learning method to estimate wire timing effectively while considering path information
- GNNTrans can encode wire paths into path representations containing whole net information, including local structures and global relationships.
- Wire timing estimator based on GNNTrans is accurate meanwhile fast.

THANK YOU!