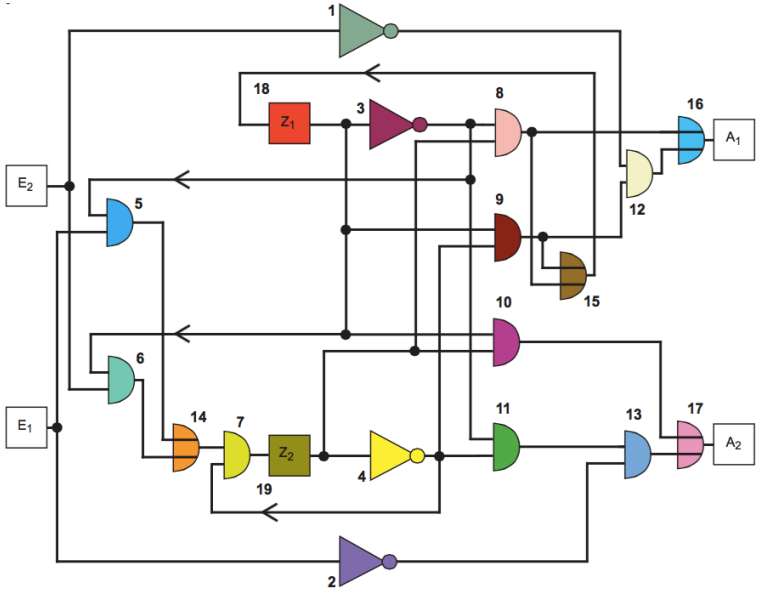
香港中文大學
The Chinese University of Hong Kong

Machine Learning in EDA

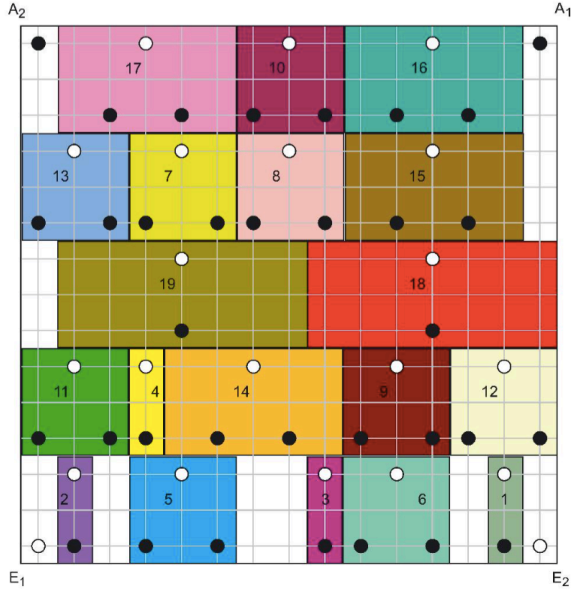
Bei Yu

Department of Computer Science and Engineering
The Chinese University of Hong Kong

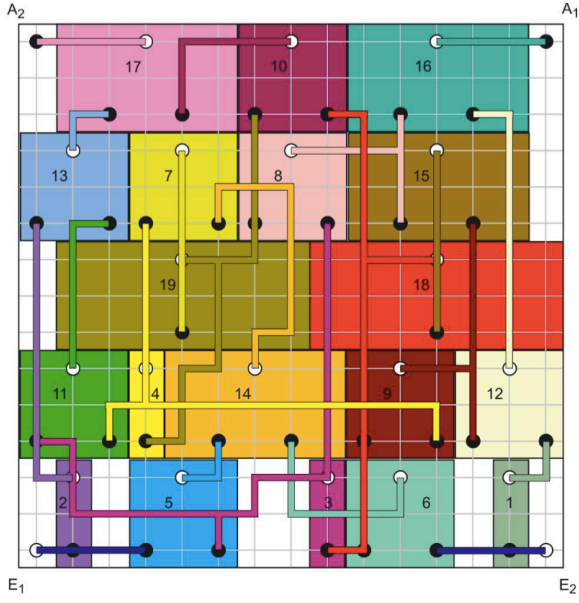
EDA Toy Example [Jens Vygen,2006]



EDA Toy Example [Jens Vygen,2006]

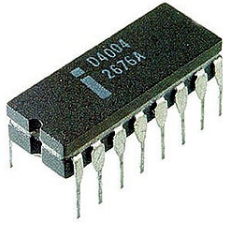


EDA Toy Example [Jens Vygen,2006]

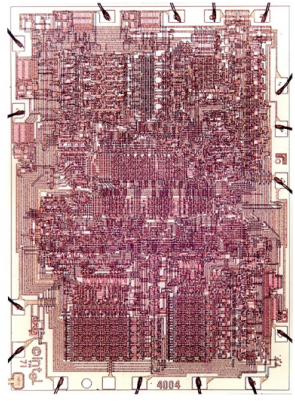


The Evolution of Computer Hardware

When was the first Microprocessor?



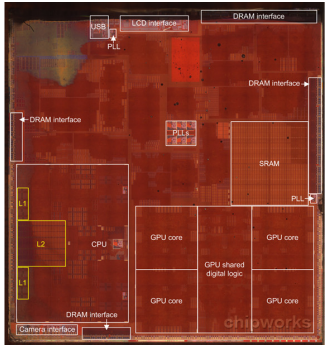
(a)



(b)

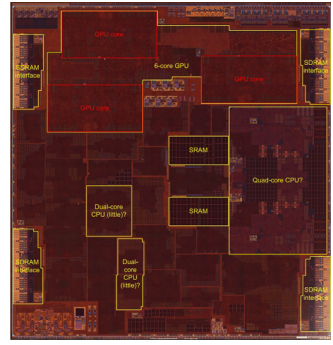
1971, Intel 4004.

Scaling of Apple SOC



Apple A7 (2013)

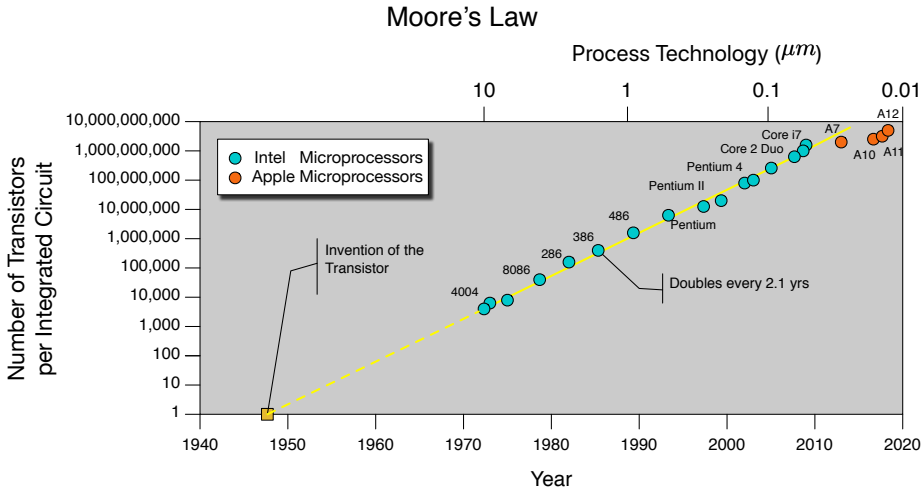
- 1,000,000,000 Transistors
- 102mm^2 die size
- 1.3GHz

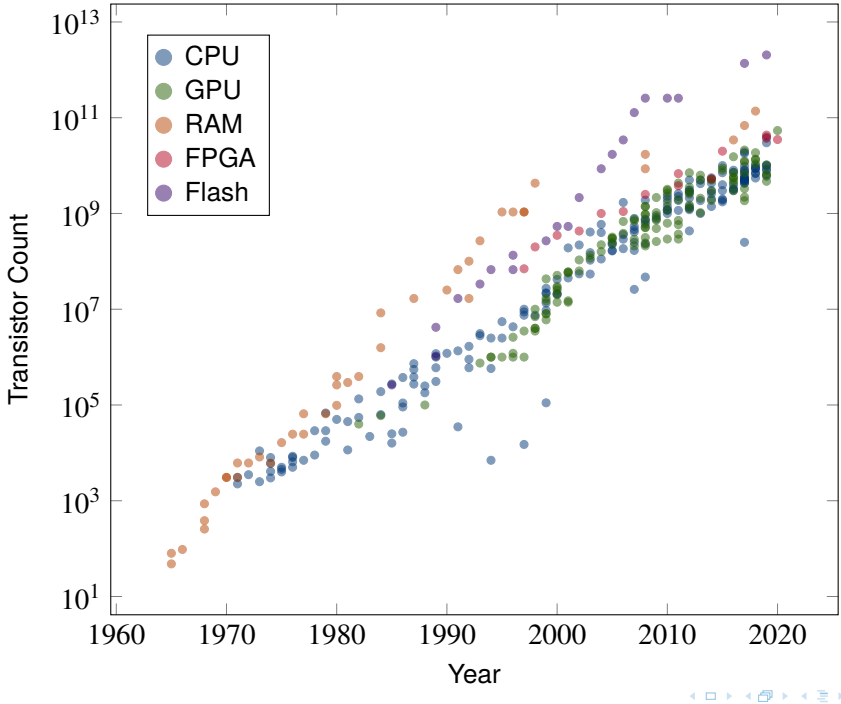


Apple A10 (2016)

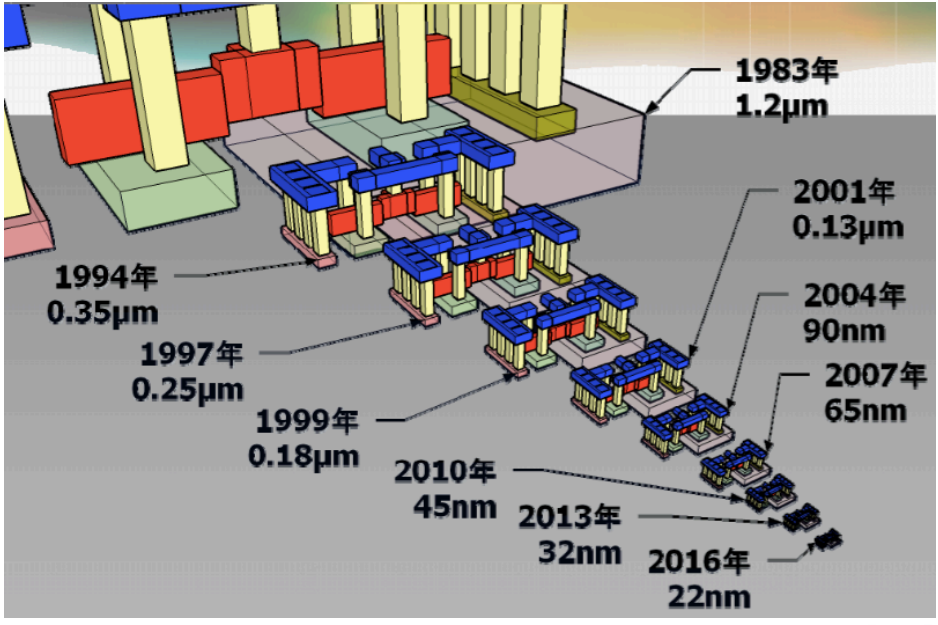
- 3,300,000,000 Transistors
- 125mm^2 die size
- 2.34GHz

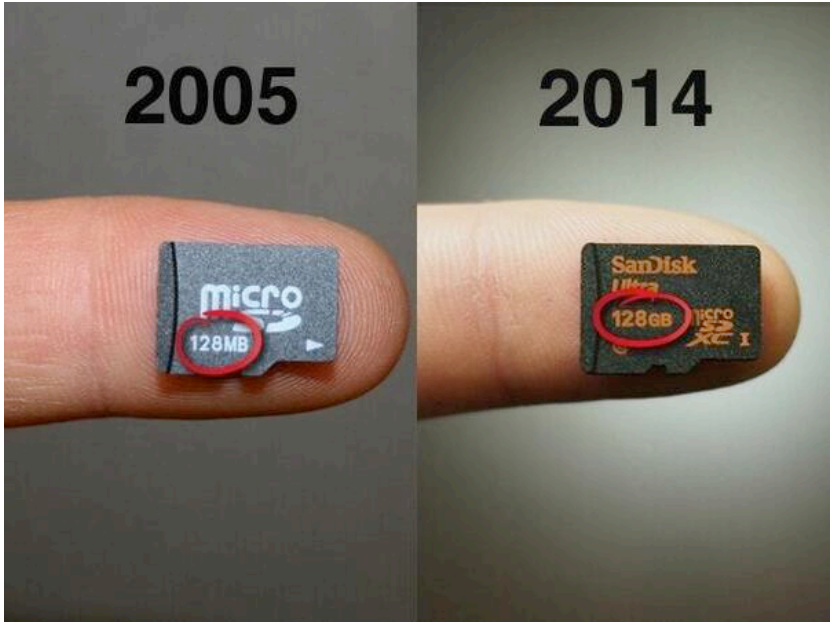
Moore's Law to Extreme Scaling



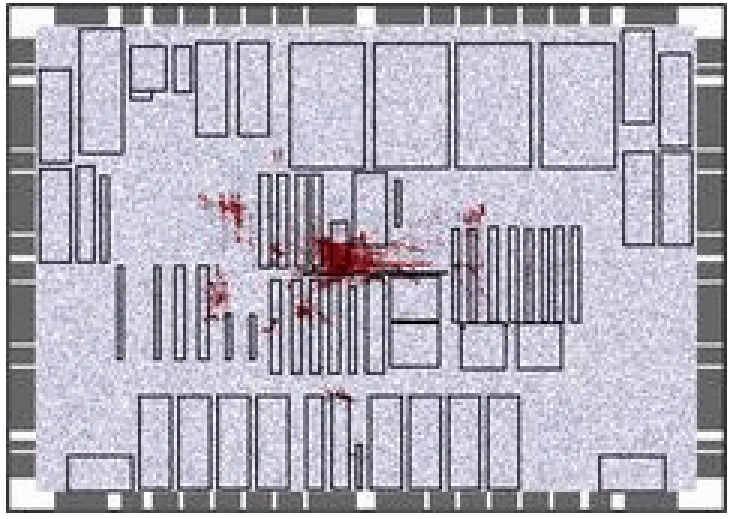

CPU
GPU
RAM
FPGA
Flash
Transistor Count
10^{13}
10^{11}
10^{9}
10^{7}
10^{5}
10^{3}
10^{1}
1960
1970
1980
1990
2000
2010
2020
Year

An Inverter Example



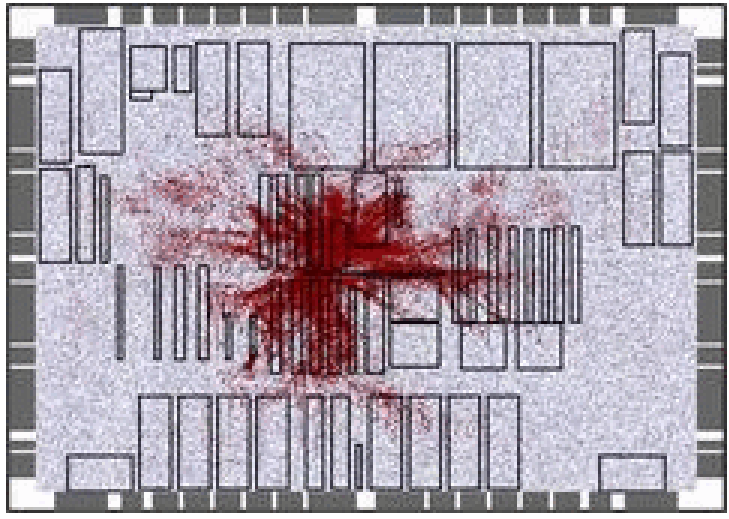

2005
2014
micro
128MB
SanDisk
128GB

Challenge: Larger and Larger Scale



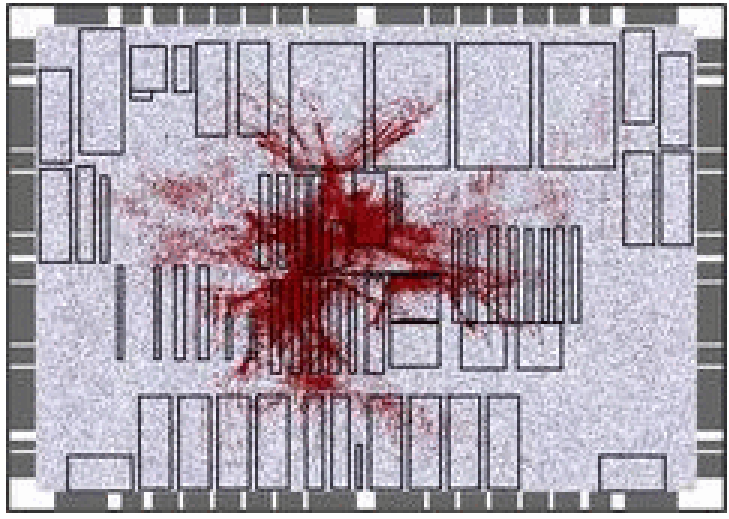
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



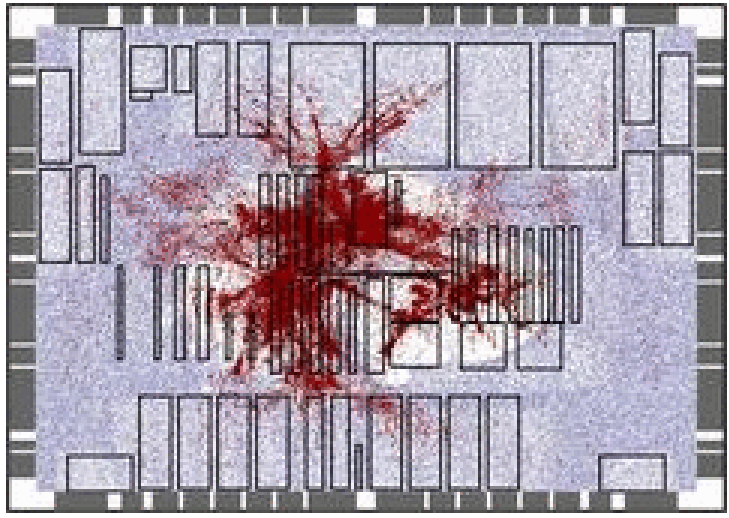
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



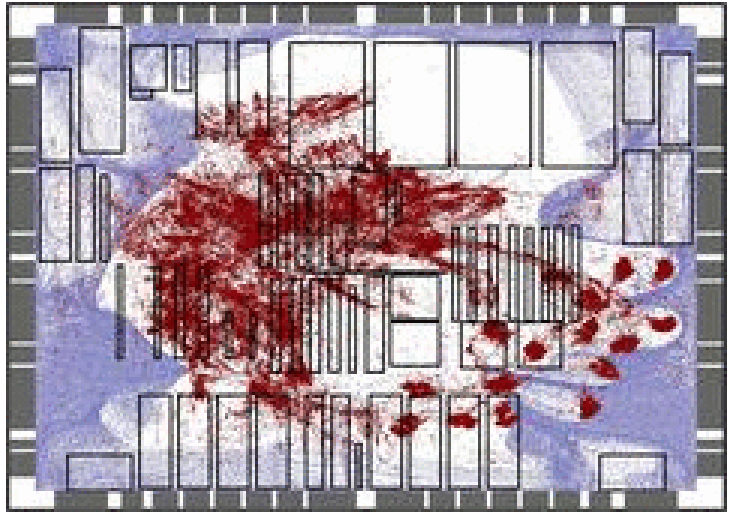
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



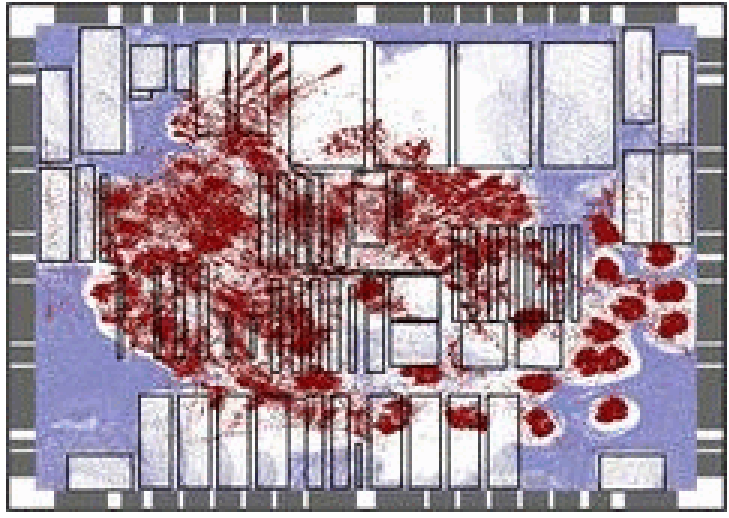
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



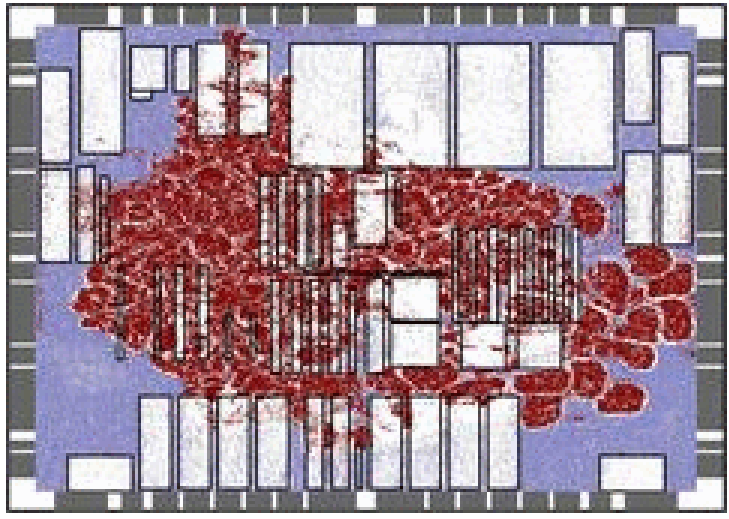
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



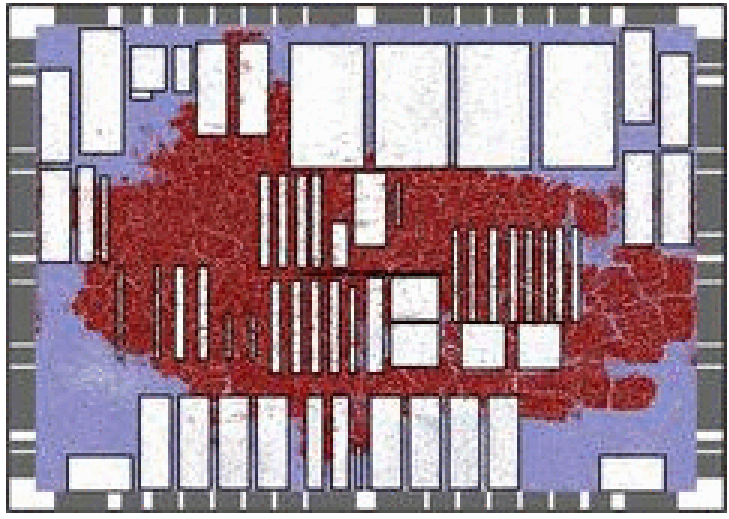
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



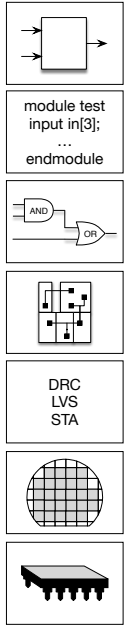
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Complicated Design Flow



System Specification

Architectural Design

Functional Design and Logic Design (RTL)

Logic Synthesis

Physical Design

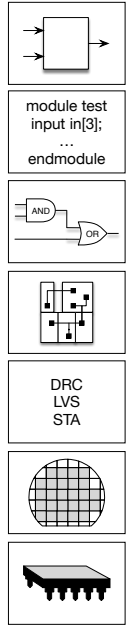
Physical Verification and Signoff

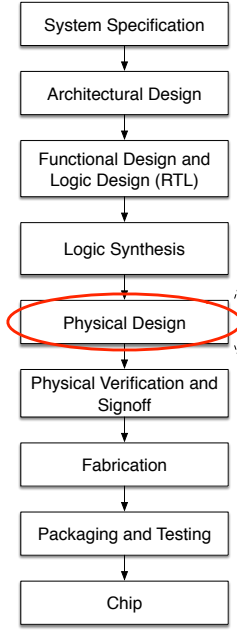
Fabrication

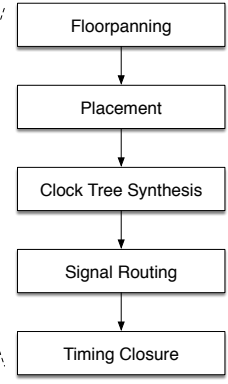
Packaging and Testing

Chip

Challenge: Complicated Design Flow



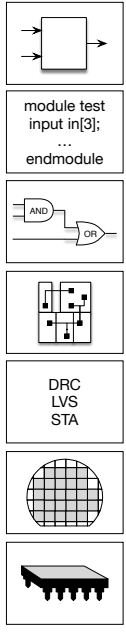




Outline

Outline

Challenge: Complicated Design Flow



System Specification

Architectural Design

Functional Design and Logic Design (RTL)

Logic Synthesis

Physical Design

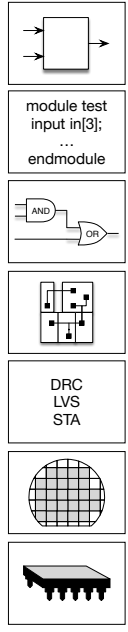
Physical Verification and Signoff

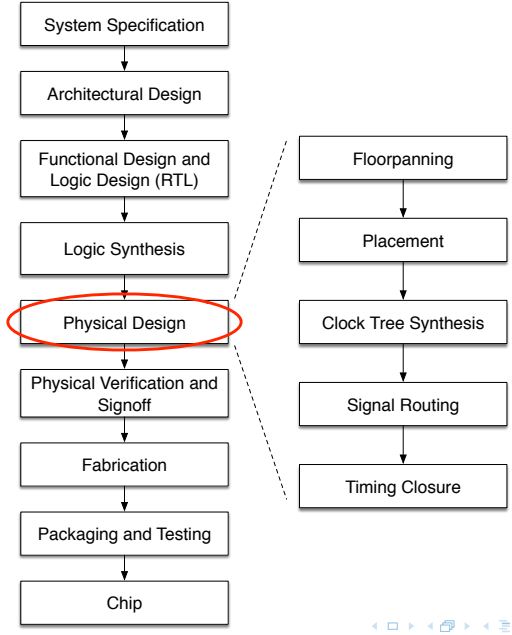
Fabrication

Packaging and Testing

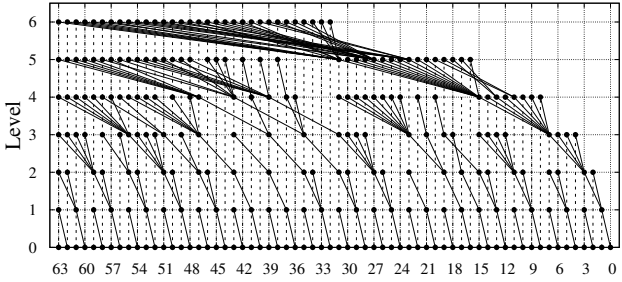
Chip

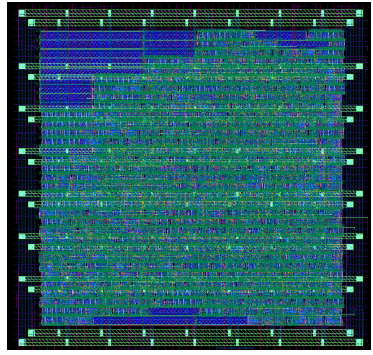
Challenge: Complicated Design Flow





Gaps Between Design Stages



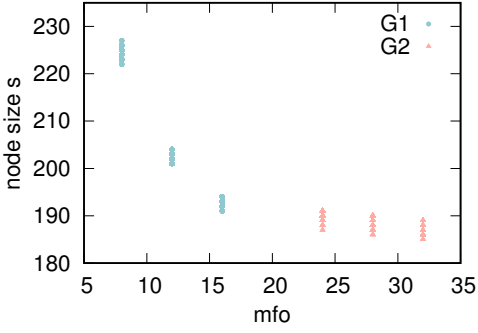


Case Study: Adder Design

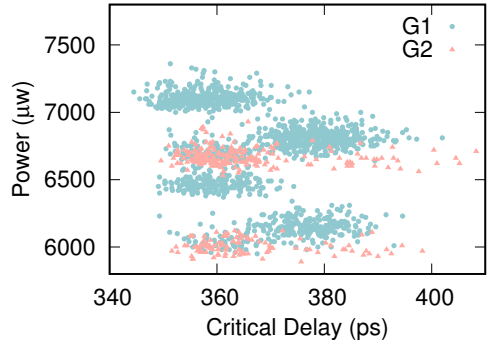
- Logic synthesis v.s. physical synthesis
- Constraints mapping between two synthesis stages is difficult.

Why Smart Sampling ?

Front-End Team Perspective:



Back-End Team Perspective:

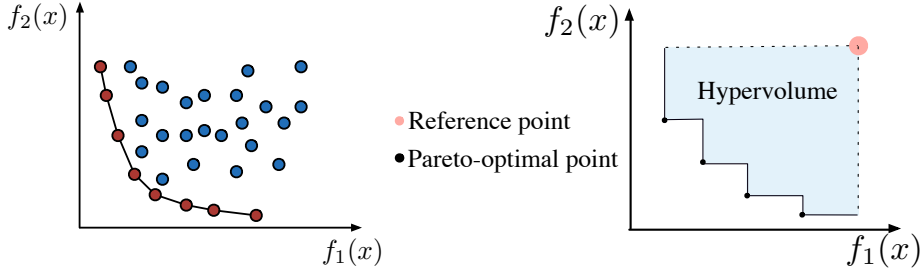


- Run design tools with all solutions is time-consuming.
- For 3K solutions, running time is $3000 \times 5 = 15$K mins.
- What we care: Pareto Frontier Curve

Pareto Frontier

Pareto Frontier

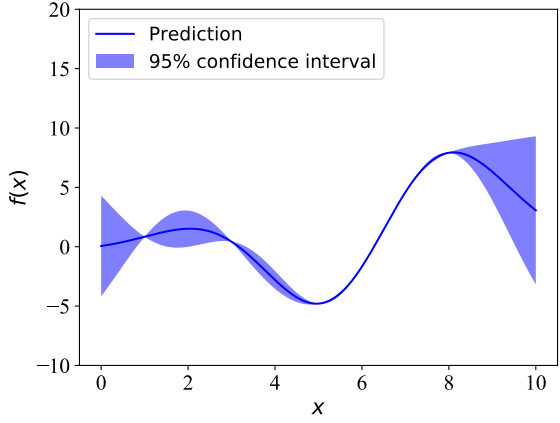
- All the points are not dominated by any other point.
- Evaluation: Hyper-volume.
  - Size of the region bounded by the Pareto frontier and reference point.
  - Each dimension of reference point is the maximum value on that dimension.



Active Learning Flow [TCAD'19]

Regression

- Gaussian process model;
- A prediction consists of a mean and a variance;
- Off-the-shelf library for implementation.



Active Learning Flow [TCAD'19]

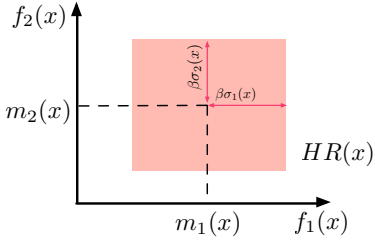
Uncertainty

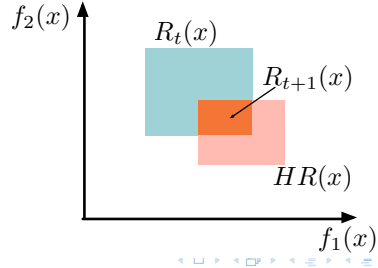
- Given the prediction (m, σ), a hyper-rectangle is defined as

$$HR(x) = \{\mathbf{y} : m_i(x) - \beta\sigma_i(x) \leq y_i \leq m_i(x) + \beta\sigma_i(x)\}$$

- The uncertainty region is defined as:

$$R_{t+1}(x) = R_t(x) \cap HR(x)$$

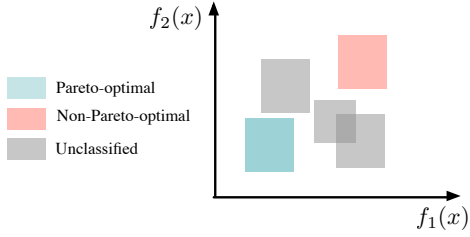


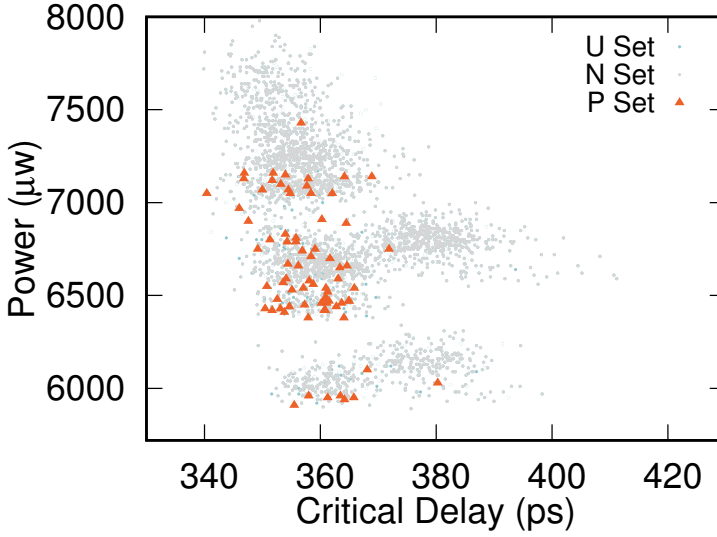


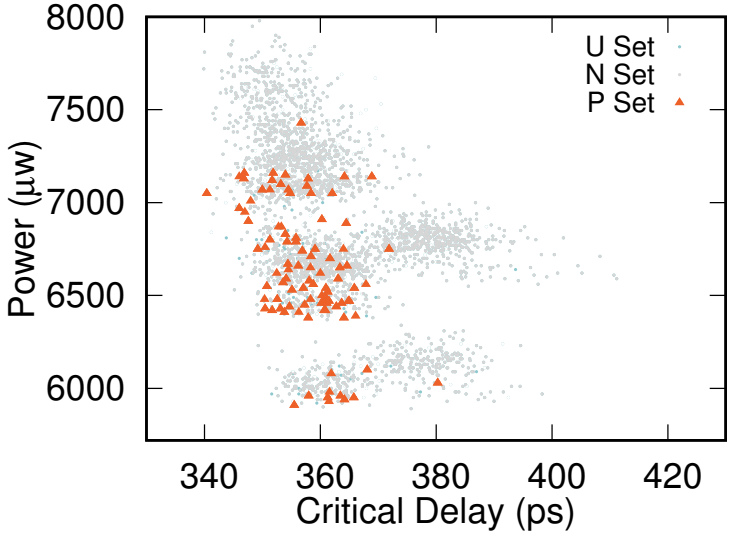
Active Learning Flow [TCAD'19]

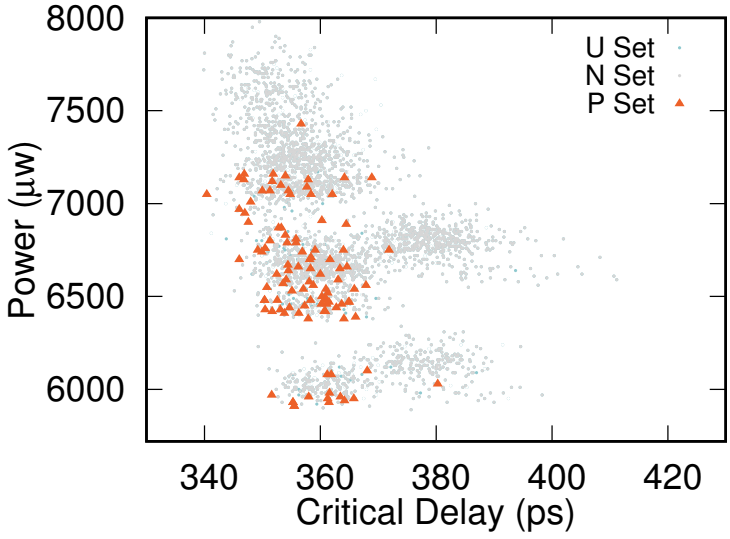
Classification

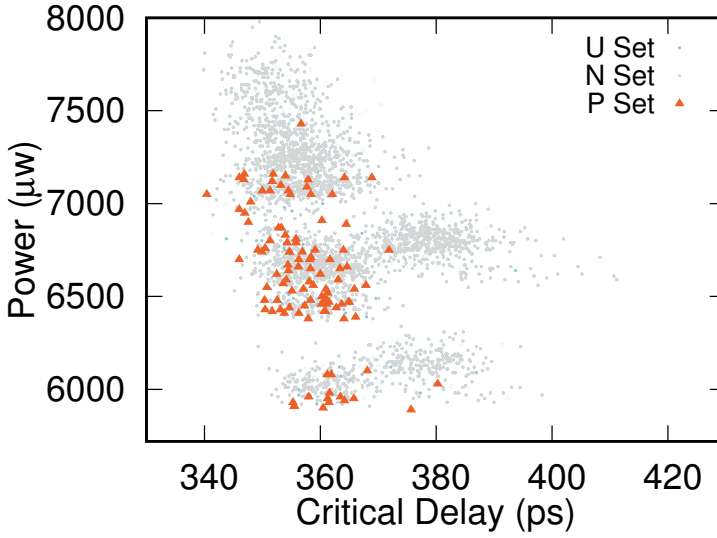
$$x \in \begin{cases} P, \text{ if } \max(R_t(x)) \leq \min(R_t(x')) + \delta, \\ N, \text{ if } \max(R_t(x')) \leq \min(R_t(x)) + \delta, \\ U, \text{ otherwise.} \end{cases}$$

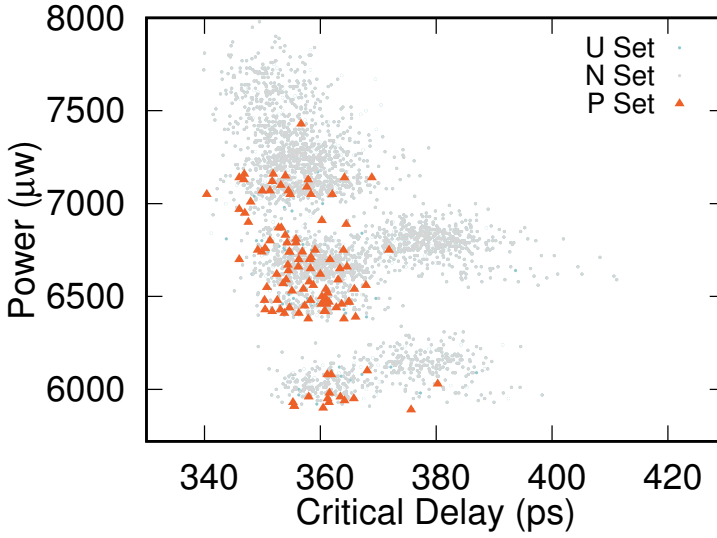


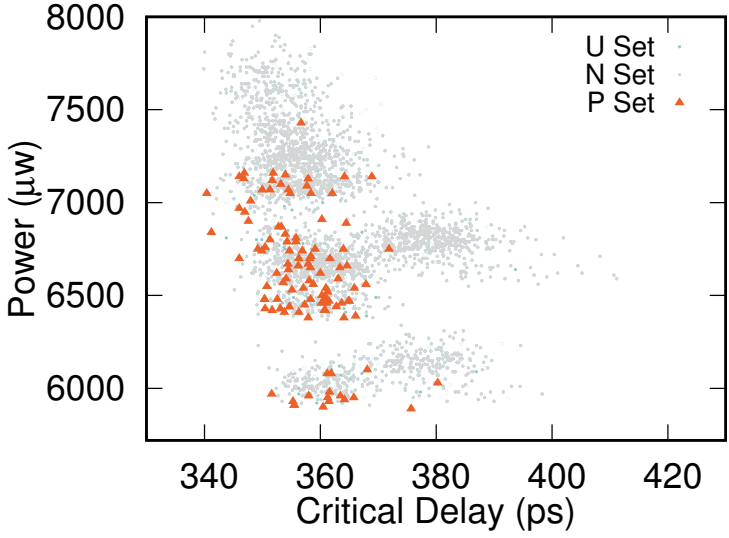

8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set


8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set

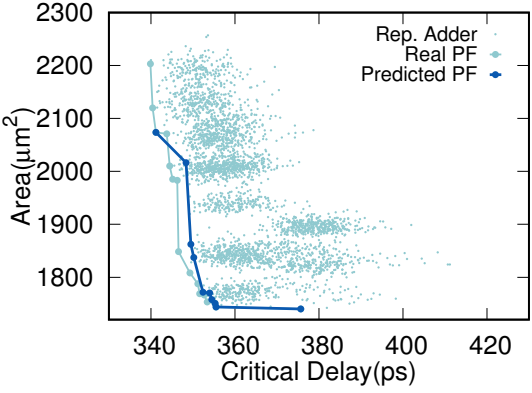

8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set

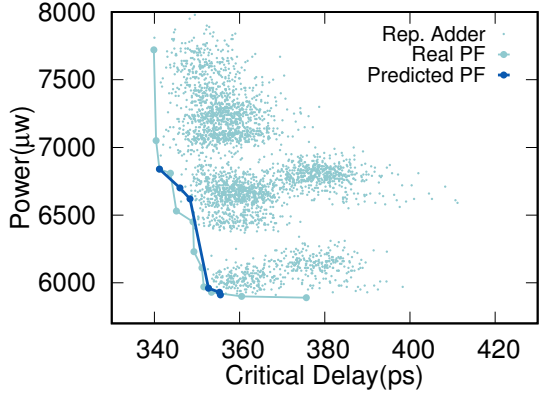

8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set


8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set

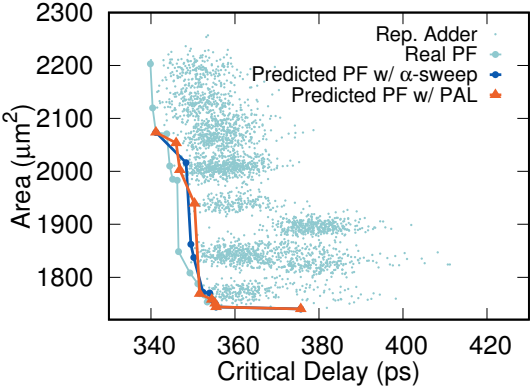

8000
7500
7000
6500
6000
Power (µw)
340
360
380
400
420
Critical Delay (ps)
U Set
N Set
P Set

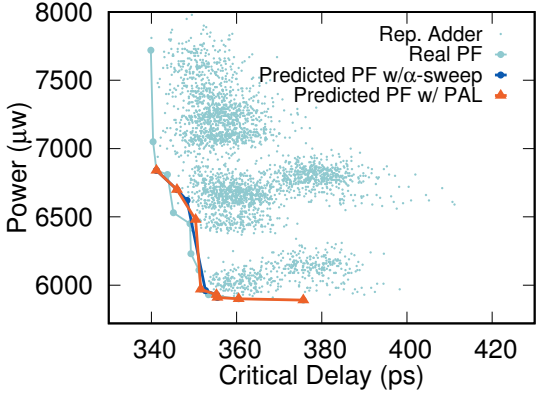
Pareto-Frontier Results





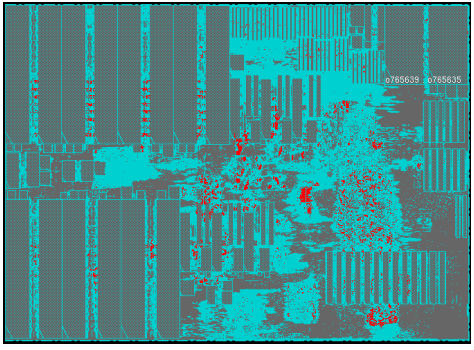
Pareto-Frontier Results





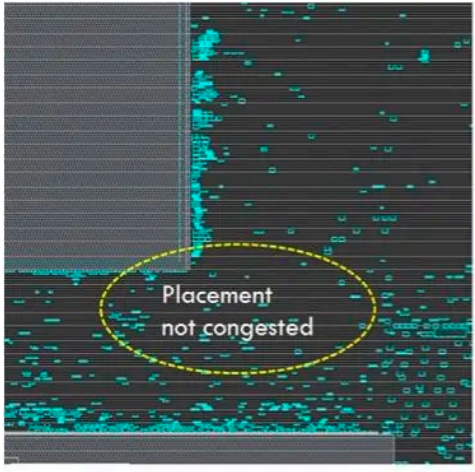
Outline

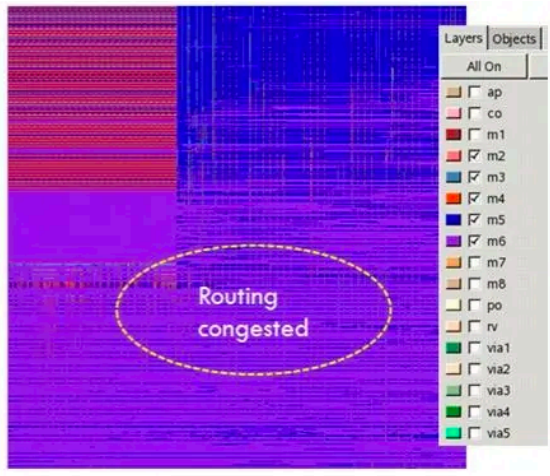
Challenge: Larger and Larger Scale



- Back-end is time consuming
- Accurate connectivity should be predictable
- Better estimation means efficient design-to-market budget

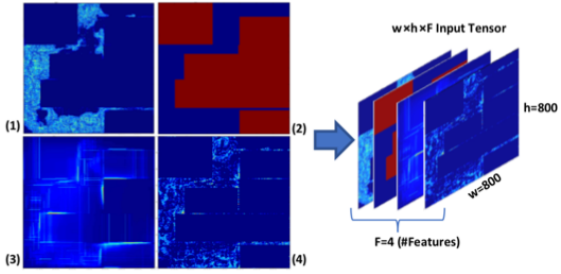
Complicated Relationship in Placement & Route



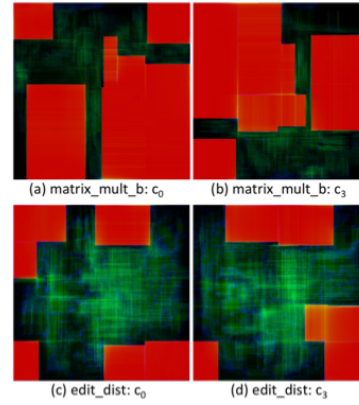


RouteNet [ICCAD'18]∗

Features Extraction



Input tensor constructed by stacking 2D features:
(1) Pin density, (2) macro (3) long-range RUDY, (4) RUDY pins

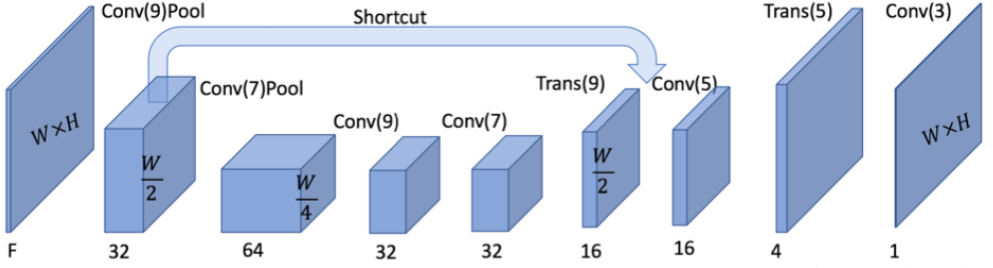


Input features for #DRV prediction.
Red: macro region
Green: global long-range RUDY
Blue: global RUDY pins

∗Zhiyao Xie et al. (2018). "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network". In: *Proc. ICCAD*.

RouteNet [ICCAD'18]∗

Proposed Model - Hotspot Detection



Filter size indicated in ()

$$Y_{i_{mn}}^{clip} = min(Y_{i_{mn}}, c)$$

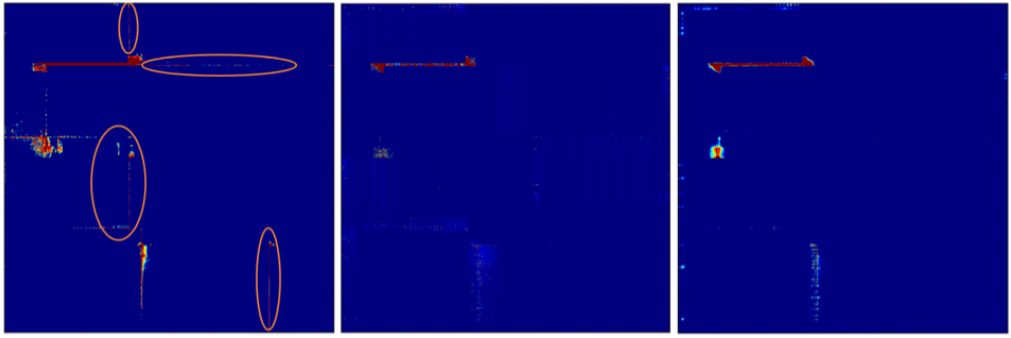
$$Loss = \sum_{i=1}^{N} \sum_{m=1}^{w} \sum_{n=1}^{h} ||f_{hotspot}(X_{i_{mn}}) - Y_{i_{mn}}^{clip}||_2 + \lambda||W||_2$$

Pixel-wise loss function

∗Zhiyao Xie et al. (2018). "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network". In: *Proc. ICCAD*.

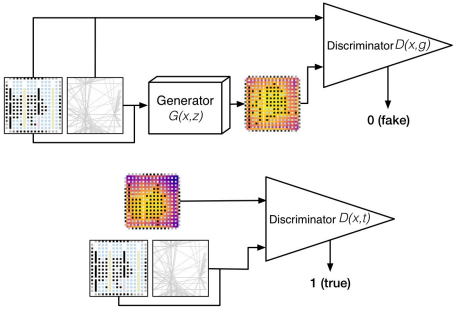
RouteNet [ICCAD'18]∗

DRC Hotspot Detection Evaluation



∗Zhiyao Xie et al. (2018). "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network". In: *Proc. ICCAD*.

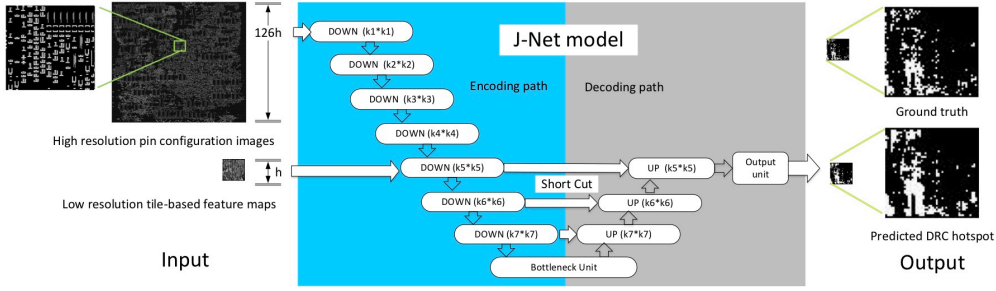
Conditional GAN [DAC'19]†



- The inputs of this network include $image_{place}$ and $image_{connect}$;
- The target image is the routing heat map image $image_{route}$;
- It only uses the post-placement information without routing information.

†Cunxi Yu and Zhiru Zhang (2019). "Painting on Placement: Forecasting Routing Congestion Using Conditional Generative Adversarial Nets". In: *Proc. DAC*.

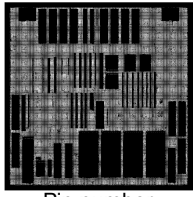
J-Net [ISPD'20]‡



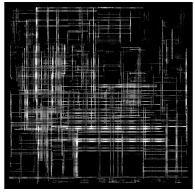
‡Rongjian Liang et al. (2020). "DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network". In: *Proc. ISPD*.

Our Study: Feature Extraction

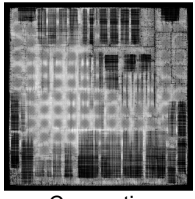
Capacity



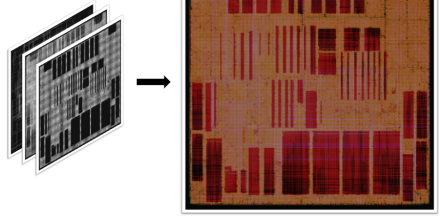
Pin number



Net density



Congestion

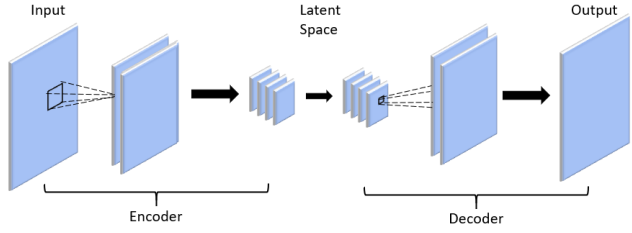


Feed features into different RGB channels

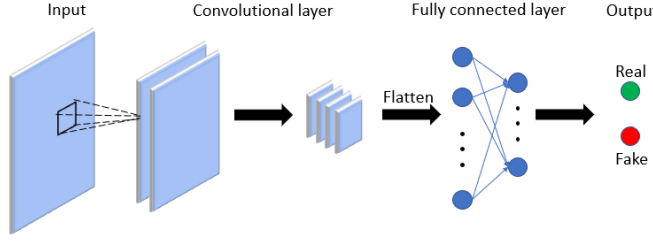
Our Study: Network Structure

Generator:

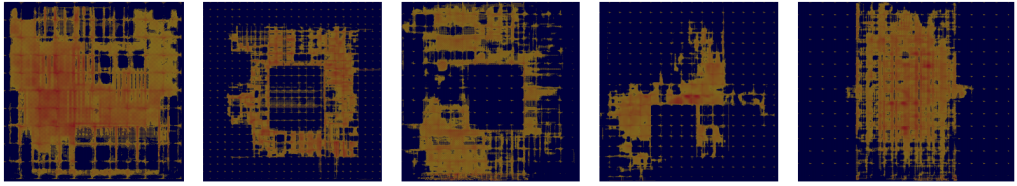
Predictor



Discriminator:



Predicted Congestion Comparison



(a) adaptec1 (b) adaptec3 (c) adaptec5 (d) bigblue3 (e) newblue2

(f) adaptec1 (g) adaptec3 (h) adaptec5 (i) bigblue3 (j) newblue2

Top row: predicted congestion map; Bottom row: actual congestion map.

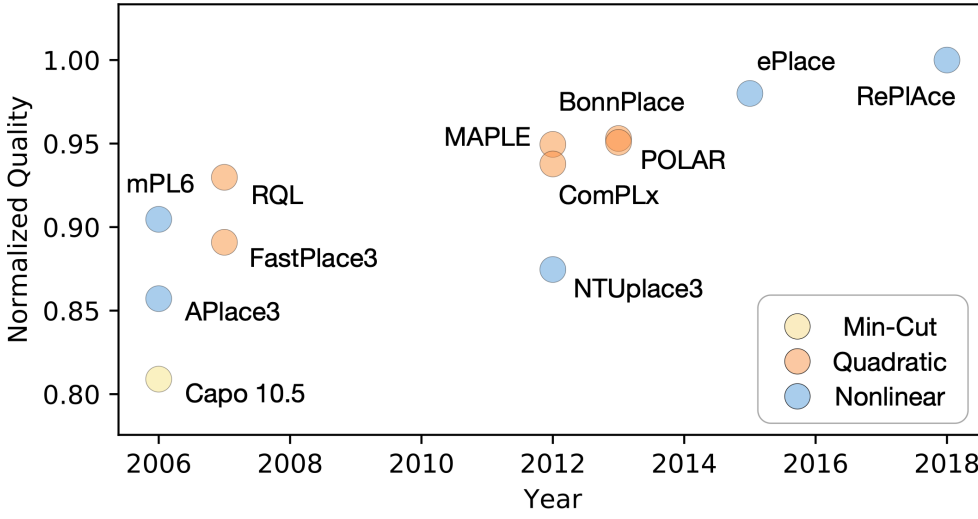
Plugged In Global Router [MLCAD'19]

	NTUgr2			NCTU-gr			NTHU-Route 2.0			Ours		
	TOF	WL	T(s)	TOF	WL	T(s)	TOF	WL	T(s)	TOF	WL	T(s)
a1	0	5.60	177.2	0	5.44	102.53	0	5.36	207.11	0	5.38	207.37
a3	0	13.41	157.7	0	13.11	154	0	13.15	225.84	0	13.10	263.89
a4	0	12.29	59.2	0	12.19	63.6	0	12.17	56.11	0	12.23	77.29
a5	0	16.03	520.4	0	15.95	381.71	0	15.53	549.98	0	15.64	611.39
b1	0	5.85	428.4	0	5.97	204.32	0	5.57	406.78	0	5.60	283.41
n2	0	7.66	27.6	0	7.59	35.73	0	7.59	30.82	0	7.59	30.65
n6	0	18.55	487.3	0	18.27	238.72	0	17.69	968.39	0	17.67	439.83
a2	0	5.36	39.8	0	5.27	36.02	2	5.23	93.4	0	5.24	51.23
n5	0	23.90	1220.1	0	23.46	281.47	18	23.14	721.52	0	23.21	399.4
b3	0	13.47	206	0	13.17	99.4	32	13.07	307.45	0	13.10	126.38
b2	2	9.42	6616.8	4	9.10	171.35	84	9.00	400.29	8	9.01	189.17
n1	38	4.87	14339.2	108	4.70	120.39	144	4.60	483.1	18	4.63	140.05
n4	148	13.55	16327.4	172	13.00	158.86	242	12.88	1032.89	172	12.90	449.65
b4	212	23.96	4478.6	512	23.17	277.64	266	22.78	2145.56	160	22.74	930.7
n3	31136	17.96	36325.5	37182	10.80	21053	n/a	n/a	>24 hrs	31050	10.70	2603.55
Total	31536	191.90	81411.2	37978	181.19	23378.74	>788	>167.78	>86400.00	31606	178.74	6803.96
Ratio	1.01	1.07	11.97	1.21	1.02	3.44	n/a	n/a	n/a	**1.00**	**1.00**	**1.00**

Zhonghua Zhou et al. (2019). "Congestion-aware Global Routing using Deep Convolutional Generative Adversarial Networks". In: *Proc. MLCAD*.

Outline

Recent Development of Placement



Typical Nonlinear Placement Algorithm

$$\min_{\mathbf{x},\mathbf{y}} \quad \mathrm{WL}(\mathbf{x},\mathbf{y}), \qquad \text{s.t.} \quad D(\mathbf{x},\mathbf{y}) \le t_d$$

Objective of nonlinear placement

$$\min \quad \underbrace{\left(\sum_{e\in E} \mathrm{WL}(e;\mathbf{x},\mathbf{y})\right)}_{\text{Wirelength}} + \lambda \underbrace{D(\mathbf{x},\mathbf{y})}_{\text{Density}}$$

Challenges of Nonlinear Placement

Low efficiency

- $>$3h for 10M-cell design

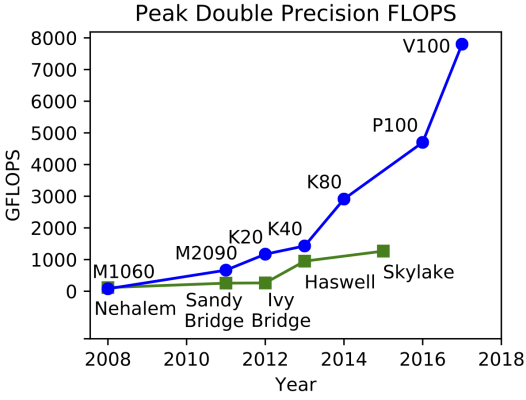
Limited acceleration

- Limited speedup, e.g. mPL, due to clustering

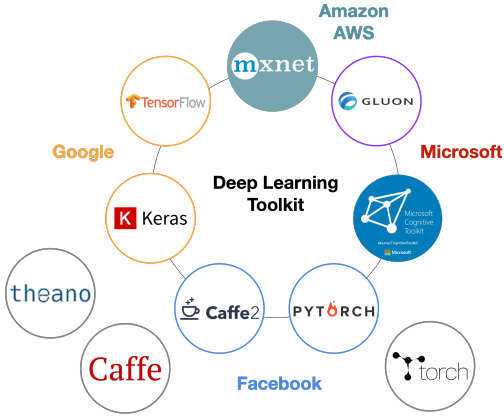
Huge development effort

- $>$1year for ePlace/RePlAce

Advances in Deep Learning Hardware/Software



Over **60x** speedup in neural network training since 2013



Deep learning toolkits

DREAMPlace Strategies [DAC'19]∗

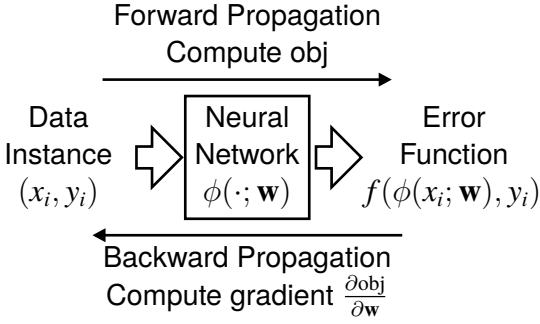
DREAMPlace Strategies

- Cast the non-linear placement problem into a neural network training problem.
- Leverage deep learning hardware (GPU) and software toolkit (e.g. `Pytorch`)
- Enable ultra-high parallelism and acceleration while getting the state-of-the-art results.

∗Yibo Lin et al. (2019). "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement". In: *Proc. DAC*.

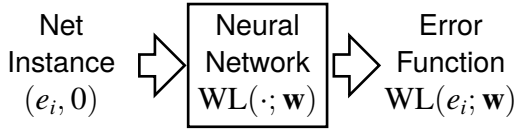
Analogy between NN Training and Placement

$$\min_{\mathbf{w}} \sum_{i}^{n} f(\phi(x_i; \mathbf{w}), y_i) + \lambda R(\mathbf{w})$$



Train a neural network

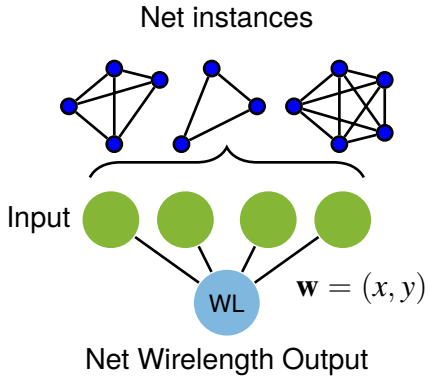
$$\min_{\mathbf{w}} \sum_{i}^{n} \text{WL}(\phi(x_i; \mathbf{w}), y_i) + \lambda D(\mathbf{w})$$



Solve a placement

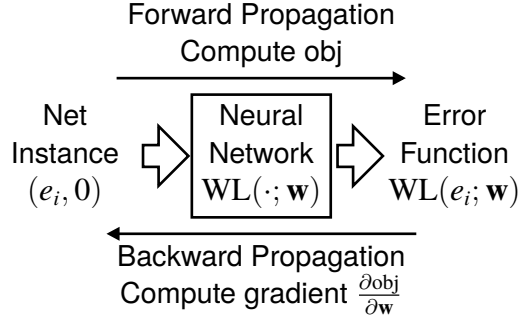
Analogy between NN Training and Placement

Casting the placement problem into neural network training



Train a neural network

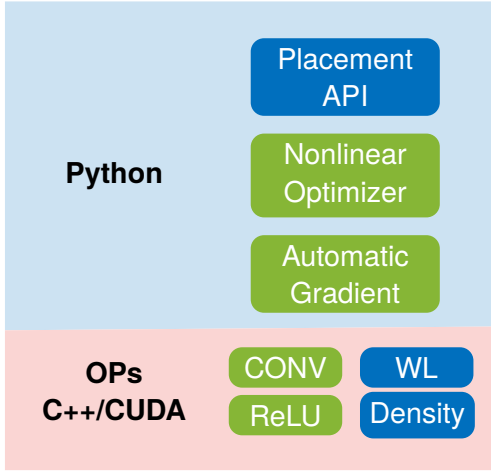
$$\min_{\mathbf{w}} \sum_{i}^{n} \mathrm{WL}(e_i; \mathbf{w}) + \lambda D(\mathbf{w})$$



Solve a placement

DREAMPlace Architecture

Leverage highly optimized deep learning toolkit PyTorch



Match RePlAce

Nesterov's Method

[TCAD'19,Cheng+]

†

†C. Cheng et al. (2019). "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement". In: *IEEE TCAD* 38.9, pp. 1717–1730.

Experimental Results

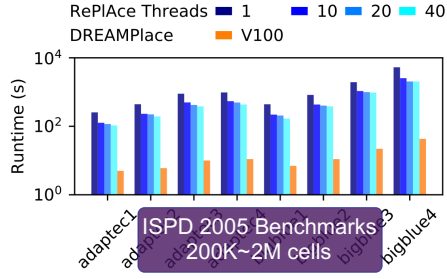
DREAMPlace
- CPU: Intel E5-2698 v4 @2.20GHz
- GPU: 1 NVIDIA Tesla V100
- Single CPU thread was used

RePlAce [TCAD'18,Cheng+]
- CPU: 24-core 3.0 GHz Intel Xeon
- 64GB memory allocated

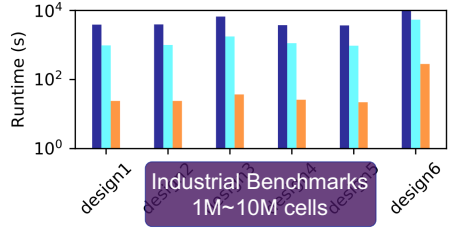
Same quality of results!

10M-cell design finishes within **5min c.f. 3h**

34x speedup



43x speedup



Future Directions

Outline

Integrating Active Learning

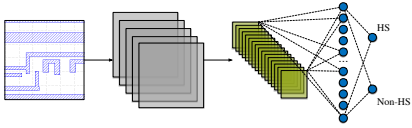
Integrating Deep Learning

Integrating Deep Learning Engine

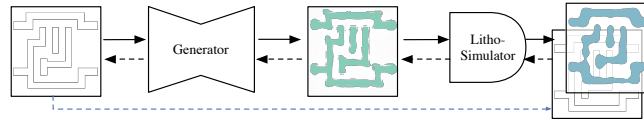
On Irregular Structure Learning: Graph Learning

Challenge: Irregular Structure Learning

- Verification [Yang et.al TCAD'2018]

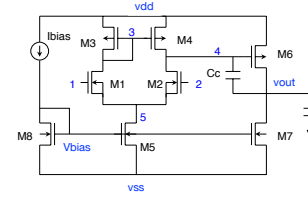


- Mask optimization [Yang et.al DAC'2018]



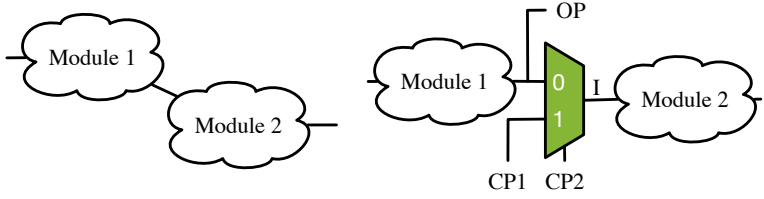
More Considerations

- Existing attempts still rely on regular format of data, like images;
- Netlists and layouts are naturally represented as graphs;
- Few DL solutions for graph-based problems in EDA.



Graph Structure Example: Test Points Insertion

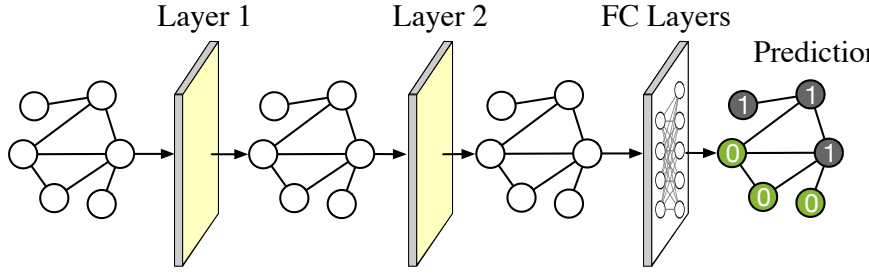
- Fig. (a): Original circuit with bad testability. Module 1 is unobservable. Module 2 is uncontrollable;
- Fig. (b): Insert test points to the circuit;
- (CP1, CP2) = (0, 1) $\rightarrow$ line I = 0; (CP1, CP2) = (1, 1) $\rightarrow$ line I = 1;
- CP2 = 0 $\rightarrow$ normal operation mode.



(a) (b)

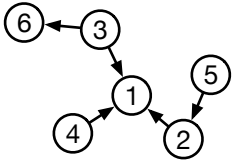
Node Classification [DAC'19]∗

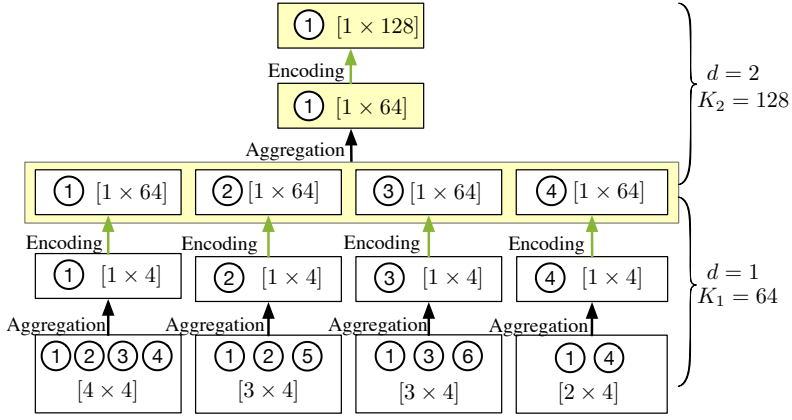
- Represent a netlist as a directed graph. Each node represents a gate.
- Initial node attributes: SCOAP values [Goldstein et. al, DAC'1980].
- Graph convolutional networks: compute node embeddings first, then perform classification.



∗Yuzhe Ma, Haoxing Ren, et al. (2019). "High Performance Graph Convolutional Networks with Applications in Testability Analysis". In: *Proc. DAC*, 18:1–18:6.

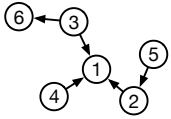
Node Embedding





Efficient Inference

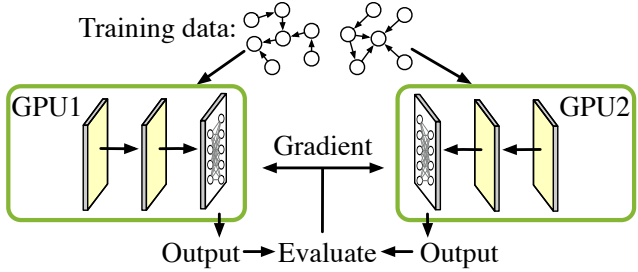
- Neighborhood overlap leads to duplicated computation $\rightarrow$ poor scalability.
- Transform weighted summation to matrix multiplication.
- Potential issue: adjacency matrix is too large.
- Fact: adjacency matrix is highly sparse! It can be stored using compressed format.



$$\boldsymbol{G}_d = \boldsymbol{A} \cdot \boldsymbol{E}_{d-1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 1 & w_1 & w_1 & w_1 & 0 & 0 \\ w_2 & 1 & 0 & 0 & w_1 & 0 \\ w_2 & 0 & 1 & 0 & 0 & w_2 \\ w_2 & 0 & 0 & 1 & 0 & 0 \\ 0 & w_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & w_1 & 0 & 0 & 1 \end{bmatrix} \end{array} \times \begin{bmatrix} \boldsymbol{e}_{d-1}^{(1)} \\ \boldsymbol{e}_{d-1}^{(2)} \\ \boldsymbol{e}_{d-1}^{(3)} \\ \boldsymbol{e}_{d-1}^{(4)} \\ \boldsymbol{e}_{d-1}^{(5)} \\ \boldsymbol{e}_{d-1}^{(6)} \end{bmatrix}$$

Efficient Training

- Adjacency matrix cannot be split as conventional way.
- A variant of conventional data-parallel scheme.
  - Each GPU process one graph instead of one "chunk";
  - Gather all to calculate the gradient.



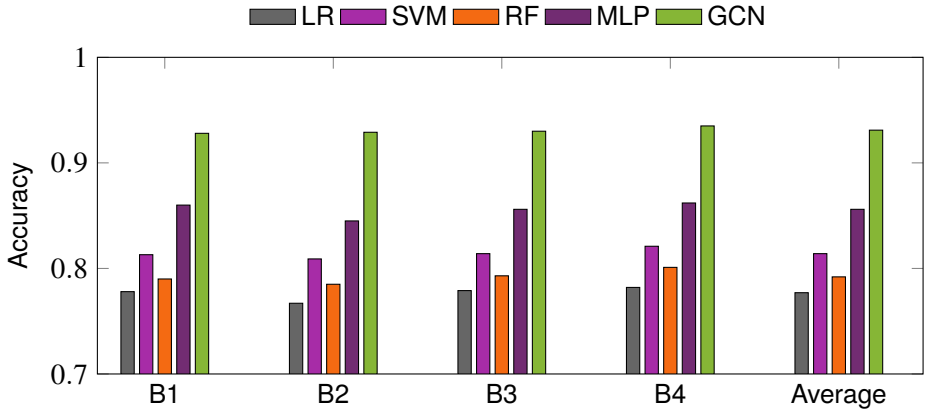
Benchmarks

- Industrial designs under 12nm technology node.
- Each graph contains > 1M nodes and > 2M edges.

Design	#Nodes	#Edges	#POS	#NEG
B1	1384264	2102622	8894	1375370
B2	1456453	2182639	9755	1446698
B3	1416382	2137364	9043	1407338
B4	1397586	2124516	8978	1388608

Classification Results Comparison

- Baselines: classical learning models with feature engineering in industry;
- GCN outperforms other classical learning algorithms.



Testability Results Comparison

- Without loss on fault coverage, 11% reduction on test points inserted and 6% reduction on test pattern count are achieved.

Design	Industrial Tool			GCN-Flow		
	#OPs	#PAs	Coverage	#OPs	#PAs	Coverage
B1	6063	1991	99.31%	5801	1687	99.31%
B2	6513	2009	99.39%	5736	2215	99.38%
B3	6063	2026	99.29%	4585	1845	99.29%
B4	6063	2083	99.30%	5896	1854	99.31%
Average	6176	2027	**99.32%**	**5505**	**1900**	**99.32%**
Ratio	1.00	1.00	**1.00**	**0.89**	**0.94**	**1.00**

Conclusion

- Active learning leverages the gap between two different design spaces
- Extention: parameter tuning?

- Enable Deep Learning in back-end phase
- Extension: Integrated in Global/Detailed Placement

- A GCN-based methodology is proposed for netlist representation;
- GCN shows superior performance to classical learning models in test points insertion problem;
- Extension: gate-sizing, or circuit reliability analysis?

Acknowledgment

CUHK colleagues:

- Prof. Evengeline F. Y. Young; Prof. Martin D. F. Wong

Industrial Liaison:

- Piyush Pathak, Frank Gennari, Ya-Chieh Lai, Jin Miao, Joydeep Mitra, Jhih-Rong Gao, Jian Kuang, Wen-Hao Liu, Zhuo Li, Charles J. Alpert (Cadence).
- Yi Zou, Jing Su (ASML); Minsik Cho (IBM); Subhendu Roy (Intel);
- Mark Ren, Brucek Khailany (nVIDIA)

Academic Liaison:

- David Z. Pan (UT Austin), Xuan Zeng, Fan Yang, Changhao Yan, Jianli Chen (Fudan);
- Shiyan Hu (University of Southampton); Wenjian Yu (Tsinghua University);
- Xin Li (Duke University), Song Chen (USTC); Wenxing Zhu (Fuzhou Univ);
- Bing Li, Ulf Schlichtmann (TUM); Yibo Lin (Peking University)

Thank You