芯片设计自动化与智能优化
Electronic Design Automation (EDA) and Intelligent Optimization

林亦波

Website: http://yibolin.com.cn

Email: yibolin@pku.edu.cn

Outline

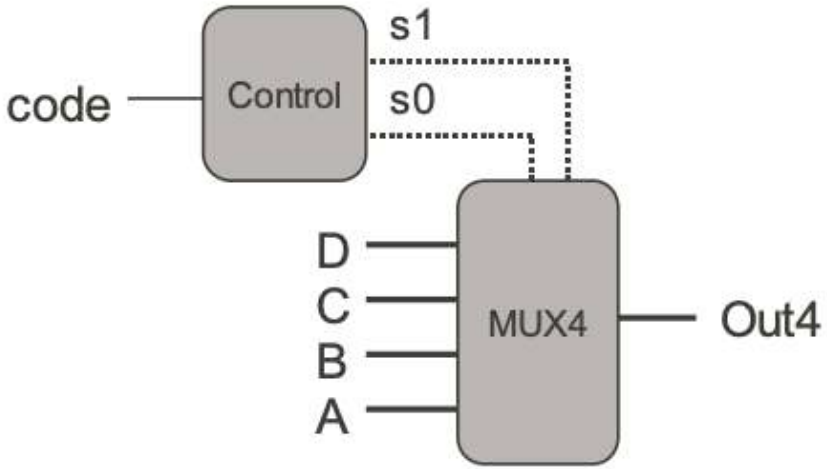
- What is EDA & IC design flow
- History of EDA
- Typical algorithms in EDA
  - Placement
  - Routing
- Summary

Hardware Description Language – HCL vs Verilog

- You have learnt to describe a hardware with codes

```
22 boolsig s0 's0_val'
23 boolsig s1 's1_val'
24 intsig code 'code_val'
25 intsig  A 'atoi(data_names[0])'
26 intsig  B 'atoi(data_names[1])'
27 intsig  C 'atoi(data_names[2])'
28 intsig  D 'atoi(data_names[3])'
29
30 ## HCL descriptions of the logic
31 bool s1 = code in { 2, 3 };
32
33 bool s0 = code in { 1, 3 };
34
35 int Out4 = [
36         !s1 && !s0 : A; # 00
37         !s1        : B; # 01
38         !s0        : C; # 10
39         1          : D; # 11
40 ];
```

HCL code

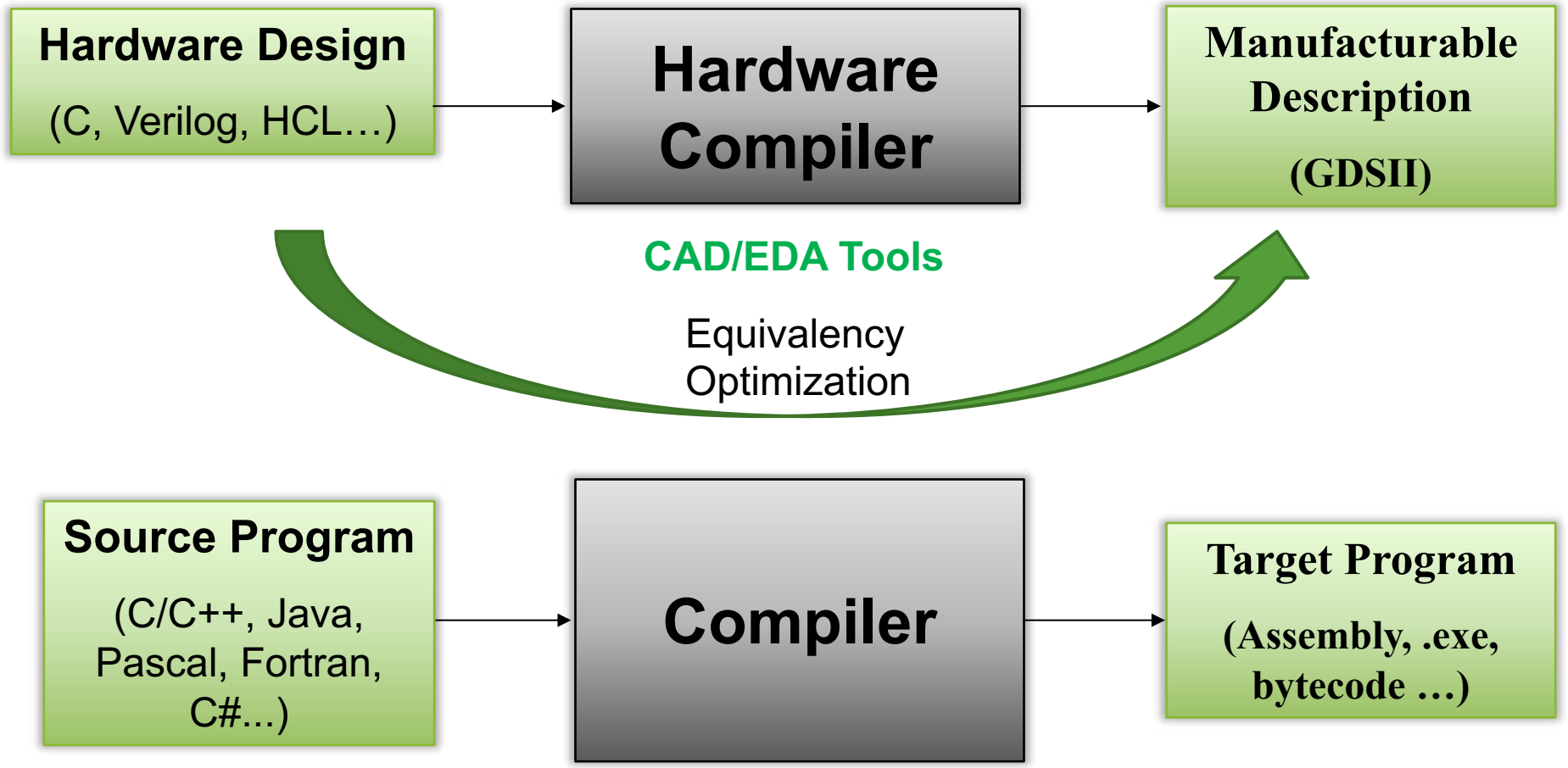


It is just the beginning…

```
1  module Control(code, s0, s1);
2  # Declare signals
3  input code;
4  output s0, s1;
5  # Describe logic
6  s0 = (code == 2) || (code == 3);
7  s1 = (code == 1) || (code == 3);
8  endmodule
9
10 module MUX4(A, B, C, D, s0, s1, Out4)
11 # Declare signals
12 ...
13 # Describe logic
14 if (!s1 && !s0) # 00
15   Out4 = A;
16 else if (!s1)   # 01
17   Out4 = B;
18 else if (!s0)   # 10
19   Out4 = C;
20 else            # 11
21   Out4 = 1;
22 end
23 endmodule
24
25 module Top(code, A, B, C, D, Out4)
26 # Declare signals
27 ...
28 # Describe logic
29 Control control_inst(.code(code), .s0(s0), .s1(s1));
30 MUX4 mux4_inst(.A(A), .B(B), .C(C), .D(D),
31   .s0(s0), .s1(s1), .Out4(Out4));
32 endmodule
```

Verilog code

IC Design Flow – Analogy to Software Compiler

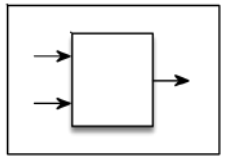


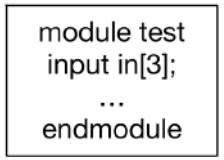
IC Design Flow – Hardware Compiler

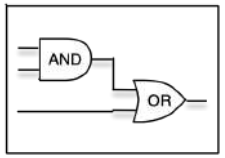
Fabless Design House

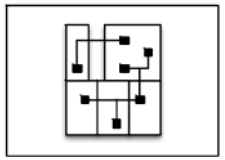
Fab/Foundry

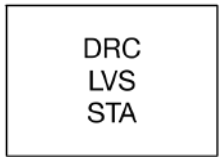
System Design | Logic Design | Backend Design

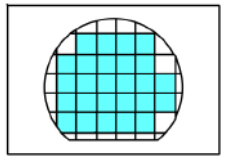


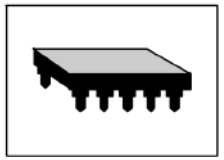












System Level Synthesis | Logic Synthesis | Physical Design | Physical Verification | Fabricate | Package Test

SystemC
SystemVerilog

Verilog
VHDL

```
module mux( input a,
    input b, input sel,
    output z, output zbar);
    assign z = sel ? b : a;
    assign zbar = z;
endmodule
```

Gate-level description

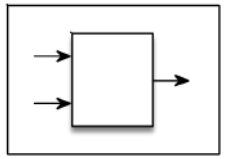
```
module and_gate( input a,
    input b, output z);
    wire zbar;
    NANDx2 inst1(a, b, zbar);
    INVx4 inst2(zbar, z);
endmodule
```

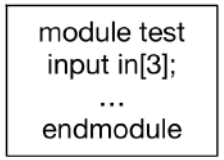
IC Design Flow – Hardware Compiler

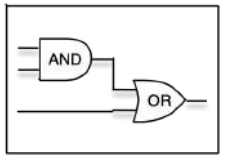
Fabless Design House

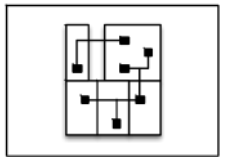
Fab/Foundry

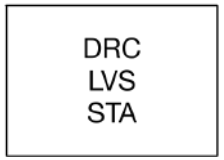
System Design | Logic Design | Backend Design

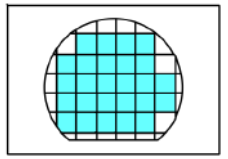


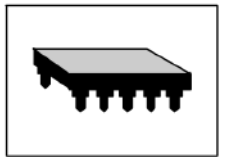




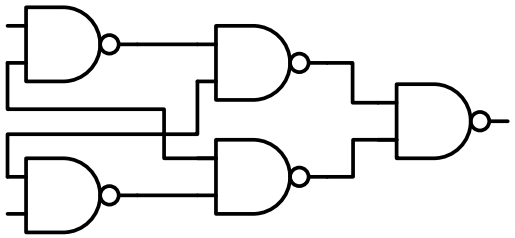


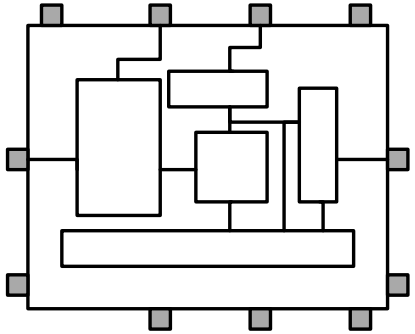


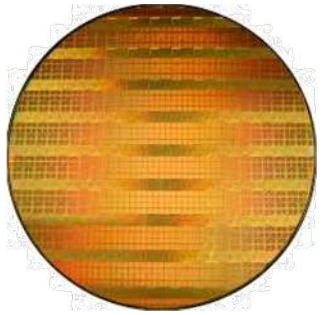




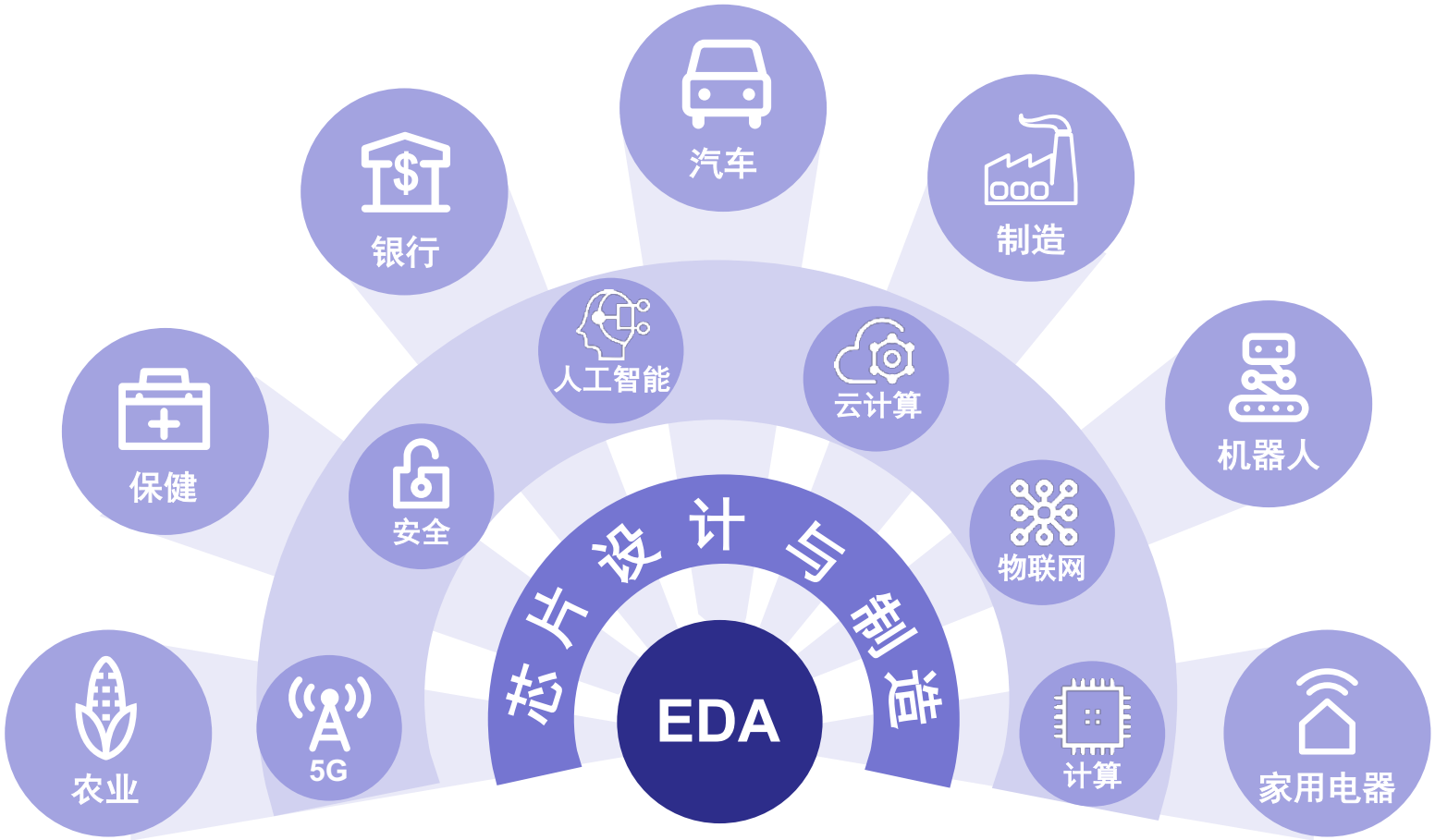
System Level Synthesis | Logic Synthesis | Physical Design | Physical Verification | Fabricate | Package Test



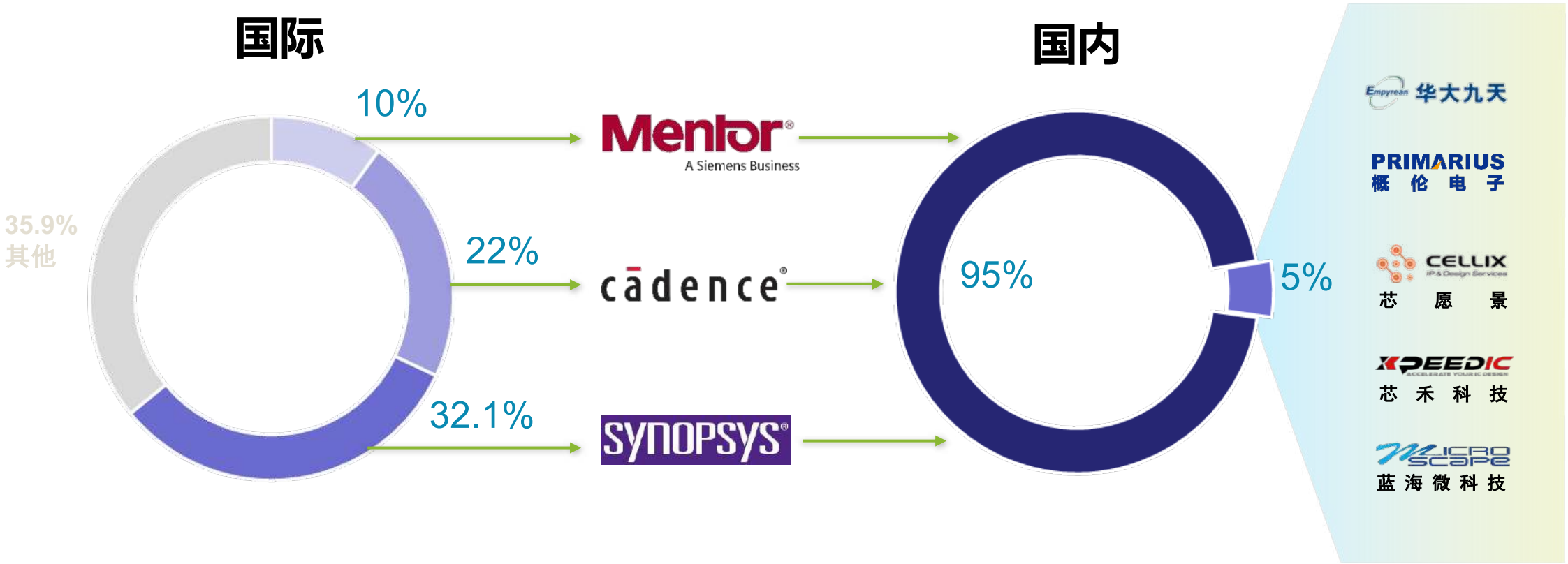




CAD/EDA – Foundation of Semiconductor Industry



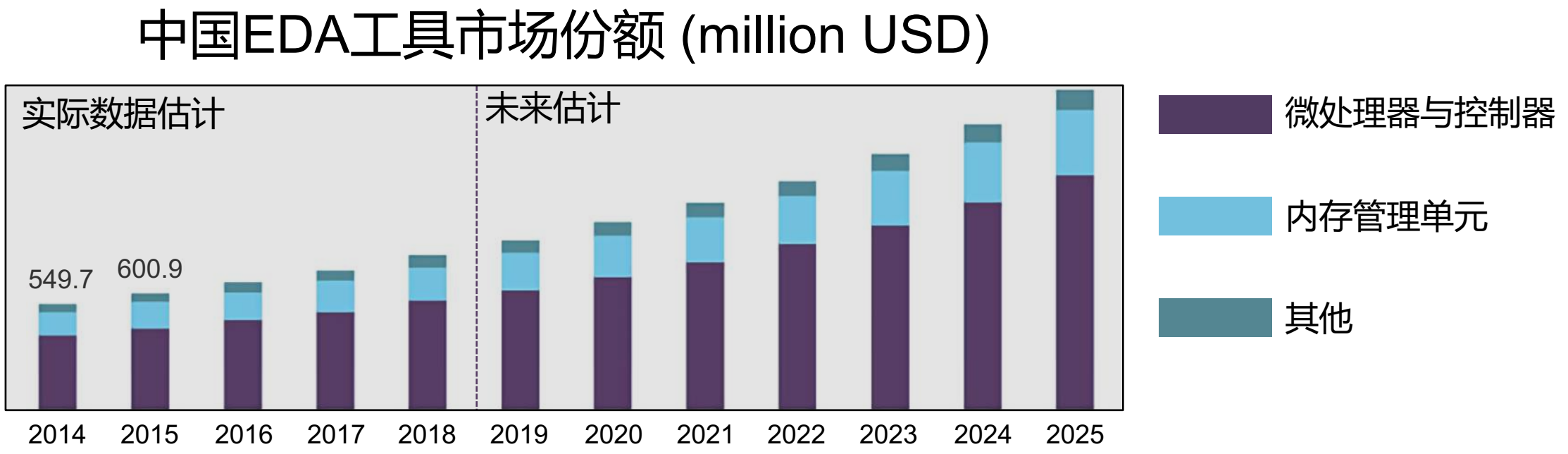
Industry and Revenue



数据来源：国海证券——芯片之母 EDA，下个国产化核心领域

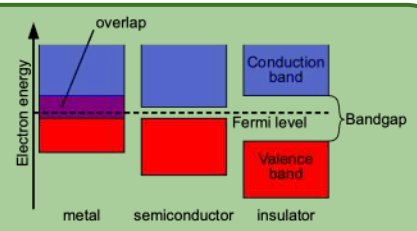
机遇与挑战：EDA的庞大国内市场

- 我国作为Synopsys和Cadence最大客户之一，学界与工业界广泛采用其EDA工具
- 我国每年需要花费数亿美元来购买EDA 工具的使用许可



History of EDA – Interdisciplinary Fusion

物理学

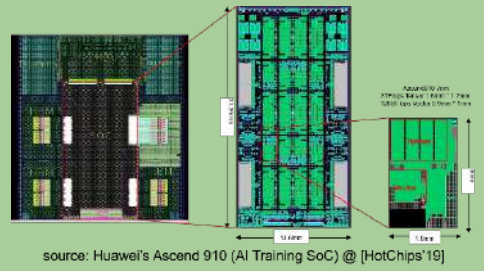


半导体能带理论

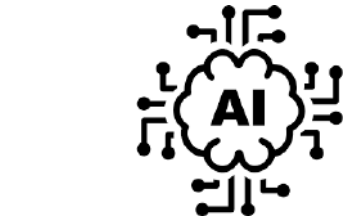
微电子学

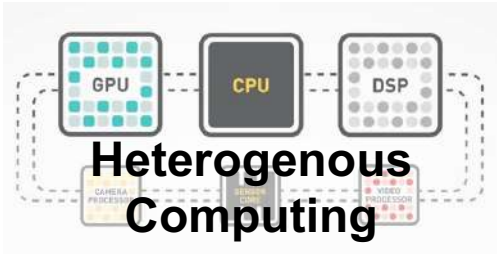
W. Shockley于1947年发明晶体管

J. Kilby于1958年发明集成电路



现代超大规模集成电路





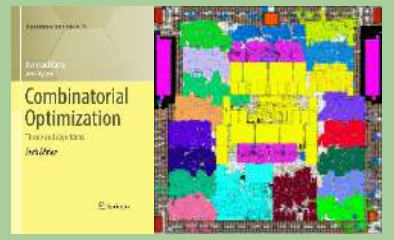
Heterogenous Computing

Next?

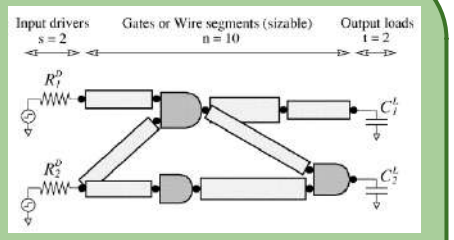
E. Kuh 葛守仁：60年代首批将数值计算用于电路设计

C.L. Liu 刘炯朗：将ad hoc EDA变革为算法驱动的EDA

计算机科学



波恩大学离散数学研究院BonnTools被全球芯片设计者使用



凸优化应用于大规模电路优化问题

应用数学

Fathers of CAD/EDA

Ernest Kuh
1928-2015
Dean@UC Berkeley

Students

- Massoud Pedram@USC
- Sanjit K. Mitra@Berkeley
- C.K. Cheng@UCSD
- Meiling Wang@Berkeley

Chung-Laung Liu
1934-2020
Professor@MIT, 1962-1972
A. Provost@UIUC, 1972-1998
President@NTHU, 1998-2002

Students

- Martin Wong@UIUC
- Andrew Yao@Princeton
- Shmuel Zaks@UIUC
- Jason Cong@UCLA
- …

Leading ad-hoc EDA to algorithm-driven EDA

https://academictree.org/etree/tree.php?pid=507028
https://academictree.org/evolution/tree.php?pid=826665

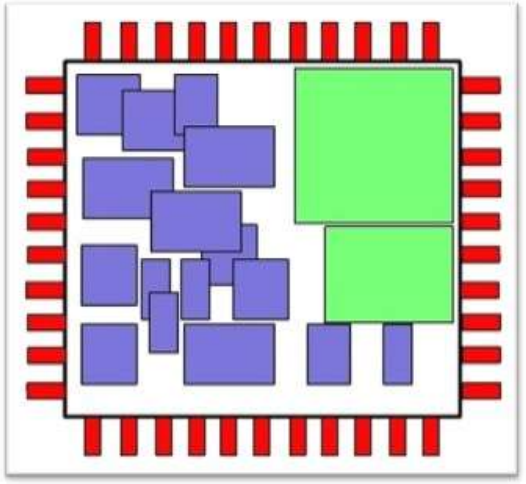
What Skills You Need to Work on EDA

- Formulate problems
  - Formalize a practical problem into a math/CS representation
  - Learn to solve practical problems with theoretical insights

Keep in mind…
In theory, we know everything, but nothing works.
In practice, everything works, but nobody knows why.

- **Algorithms and programming skills**
  - Typical ones to solve graph and numerical optimization
  - E.g., network flow, partitioning, dynamic programming, basic convex optimization

Widely-Existing Problems – Placement



Circuit Placement

Urban Planning

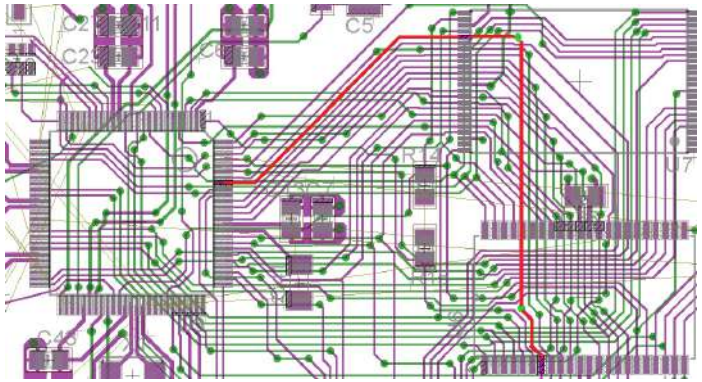
Warehouse Location

Storage Planning

Book Placement

Data Placement on Disk

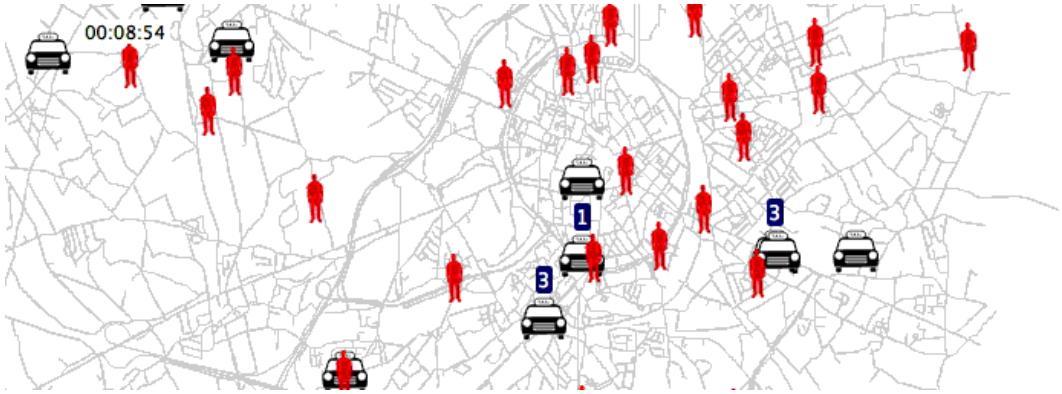
Widely-Existing Problems – Routing



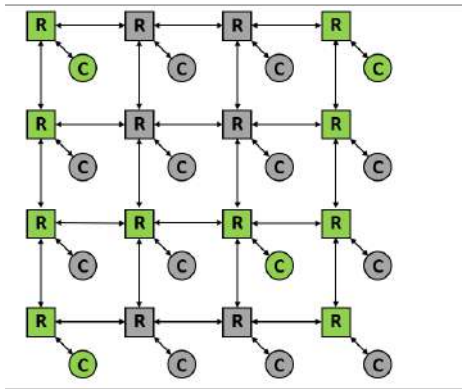
Circuit Routing

Road Planning

GPS Navigation



Vehicle Routing



Network-on-Chip Routing

With Skills in EDA Algorithms, You Can Work at…

EDA and hardware companies

…

Software companies

…

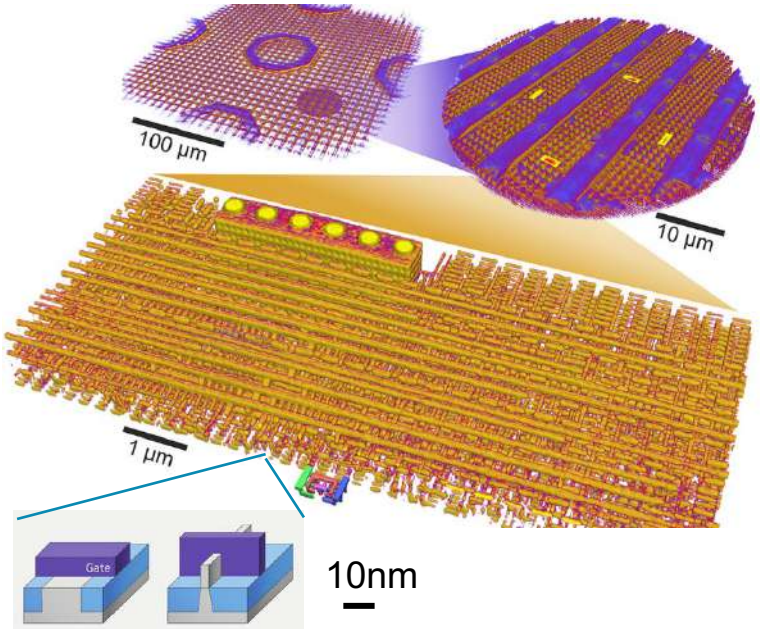
Challenges of EDA – A Rough Analogy

Smart City

Urban Planning : 1 m → 100 km

(Area of Beijing : 160 x 176 km^2)

Chip : Nanoscale City

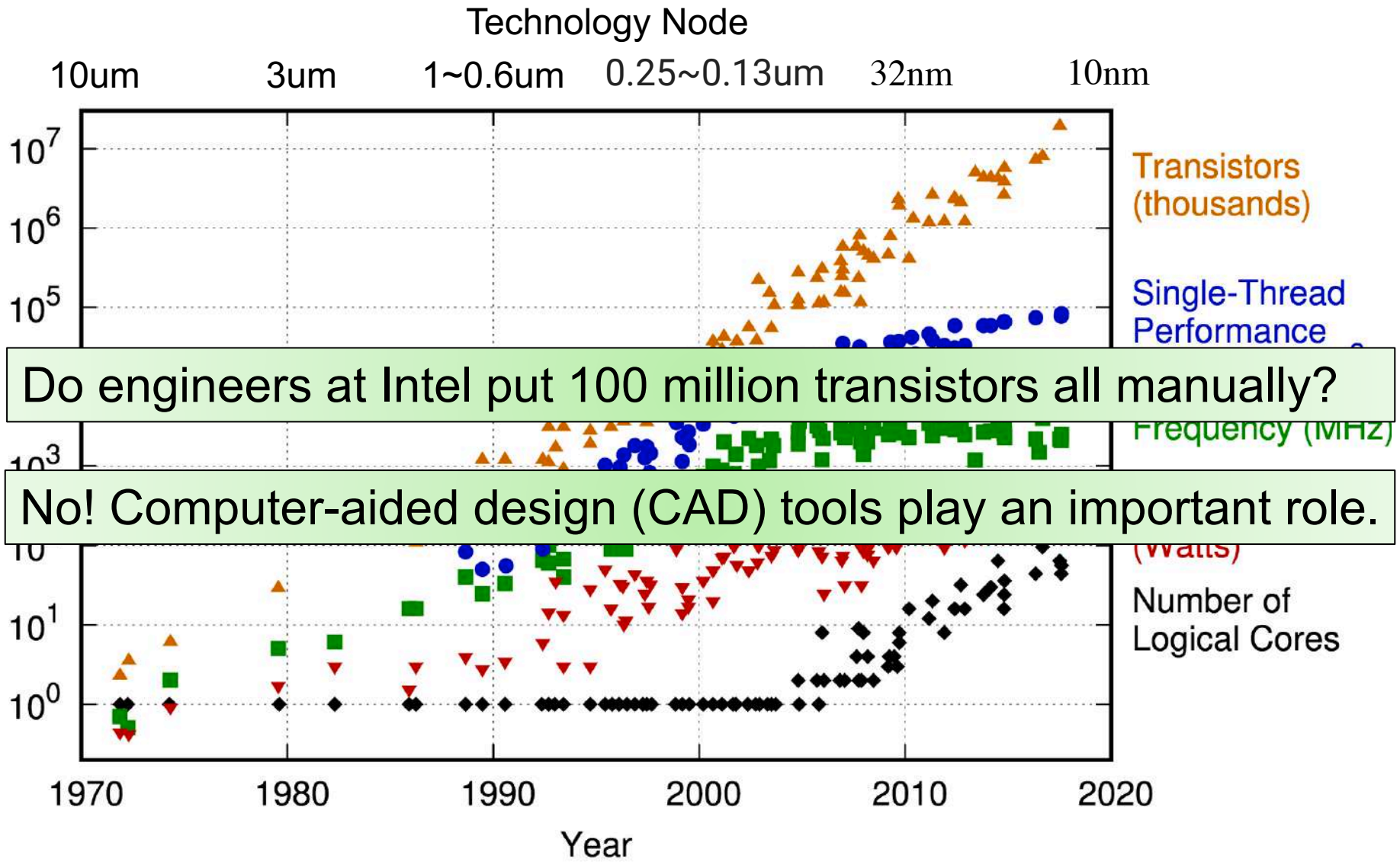


IC Design : 10 nm → 10 mm

How Many Transistors on a CPU?

Microprocessor	Year	Transistors
4004	1971	2,300
8080	1974	4,500
8086	1978	29,000
Intel286	1982	134,000
Intel386	1985	275,000
Intel486	1989	1,200,000
Intel Pentium	1993	3,100,000
Intel Pentium II	1997	7,500,000
Intel Pentium III	1999	9,500,000
Intel Pentium 4	2000	42,000,000
Intel Itanium 2	2003	220,000,000

Moore's Law



Do engineers at Intel put 100 million transistors all manually?

No! Computer-aided design (CAD) tools play an important role.

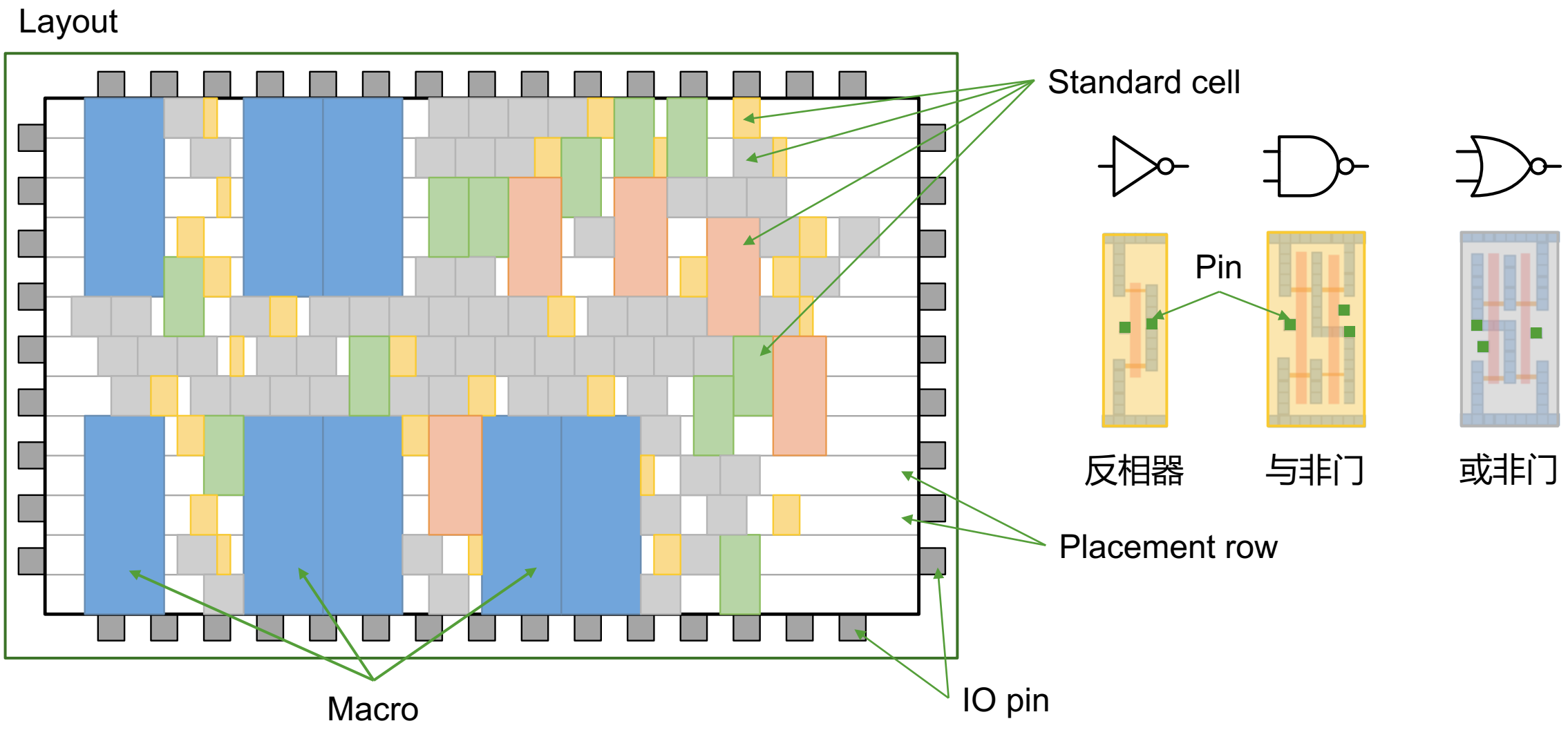
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Any questions?

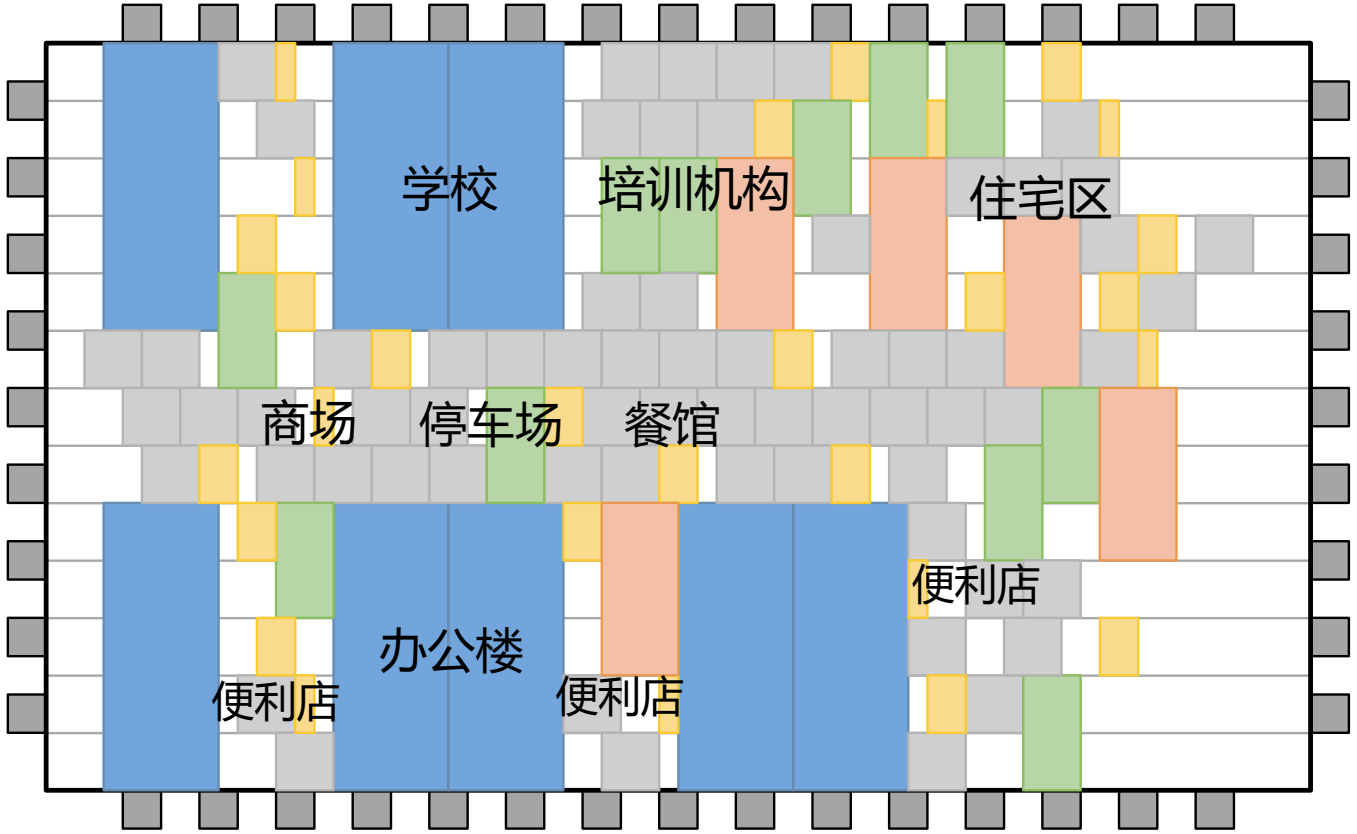
Outline

- What is EDA & IC design flow
- History of EDA
- **Typical algorithms in EDA**
  - Placement
  - Routing
- Summary

What is Placement



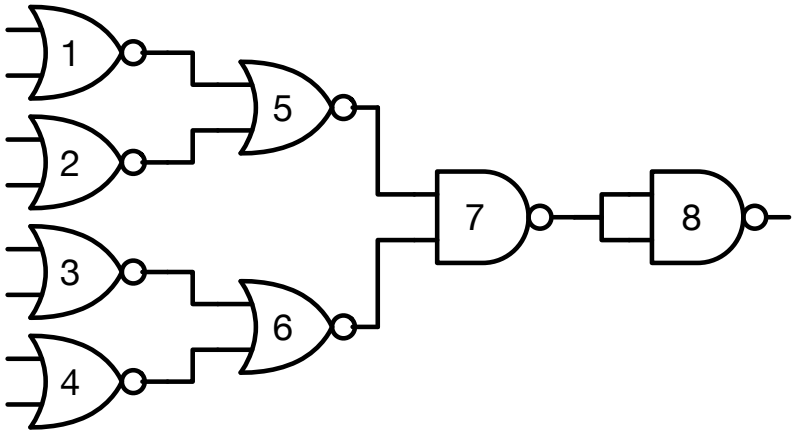
Analog to Urban Planning

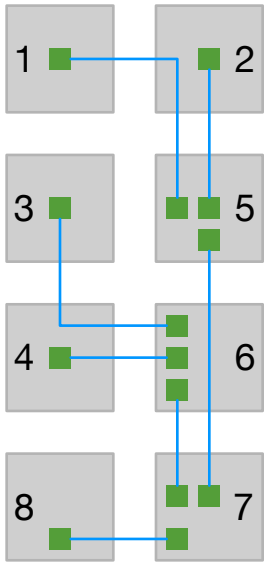


好的城市规划应使功能关联建筑之间的距离尽可能小

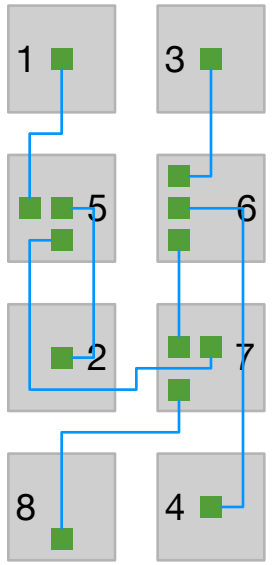
Metrics for Placement

- Wirelength: length of physical wires connecting cells





Wirelength=100



Wirelength=150

好的布局应使相连Cell之间的距离尽可能小

Problem Formulation for Placement

- Input
  - Blocks (standard cells and macros) B_1, … , B_n
  - Shapes and Pin Positions for each block B_i
  - Nets N_1, … , N_m
- Output
  - Coordinates (x_i , y_i) for block B_i.
  - No overlaps between blocks within a fixed layout area
- Objective
  - The total wirelength is minimized
- Other objectives: timing, routability, clock, buffering

How Difficult Placement is

#states: ~10^{123}

#states: ~10^{360}

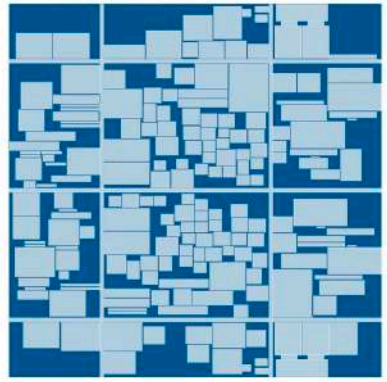
Google AlphaGo
Train **40 days** using **176 GPUs**

How Difficult Placement is

- Huge problem sizes：tens of millions of cells
- Huge solution space：larger than $1K \times 1K$ grids in a layout

#states: ~10^{123}

#states: ~10^{360}



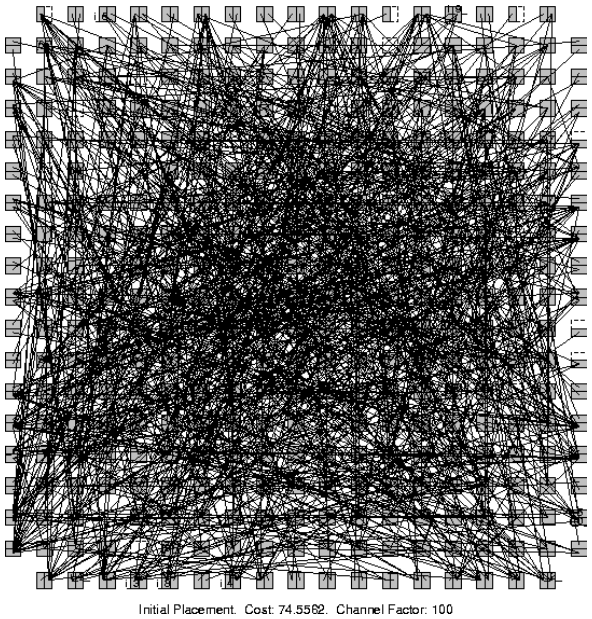
#states: >$10^{100,000}$

Google AlphaGo
Train **40 days** using **176 GPUs**

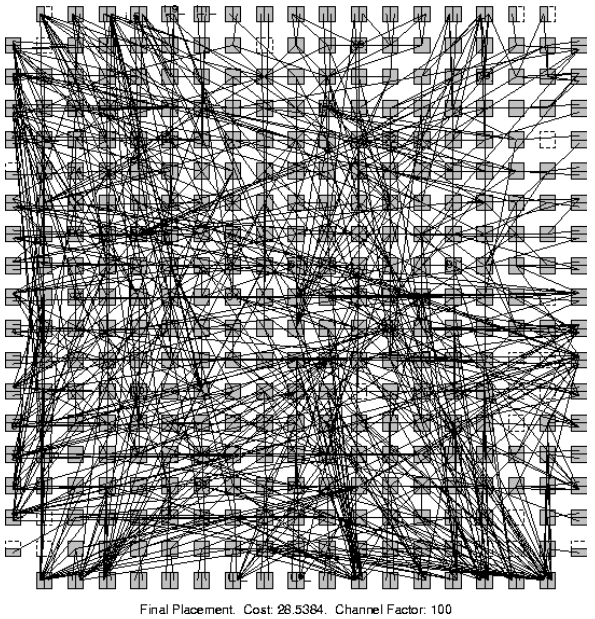
Good Placement vs Bad Placement

- **230** cells in FPGA (design *e64* in the MCNC benchmark suite)

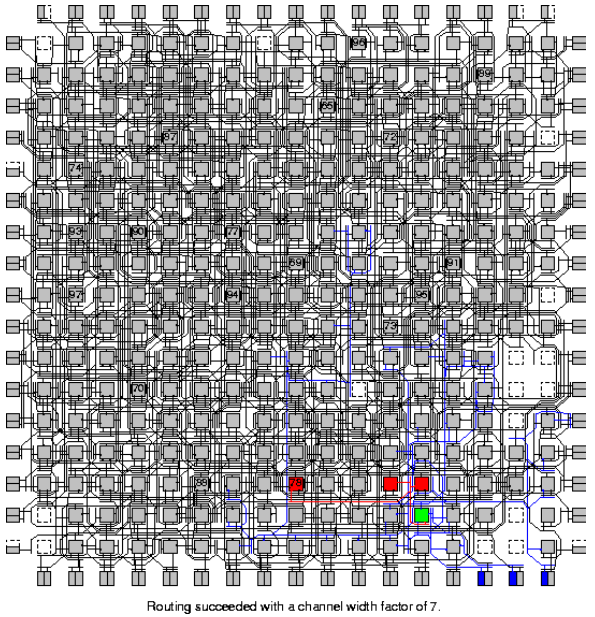
Random Initial	Final Solution	Routing Solution



WL = 5.47e+4

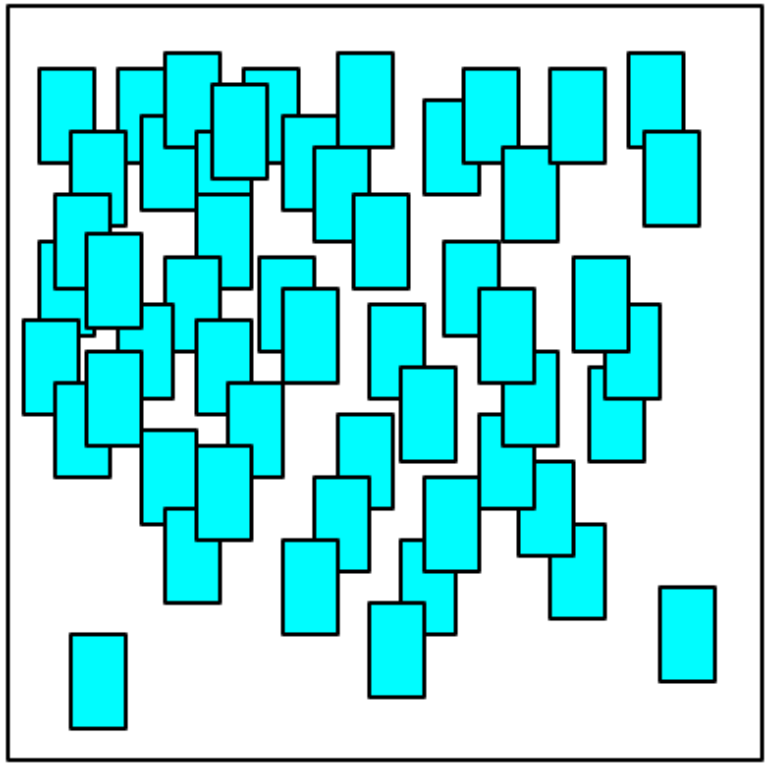


WL = 6.73e+3



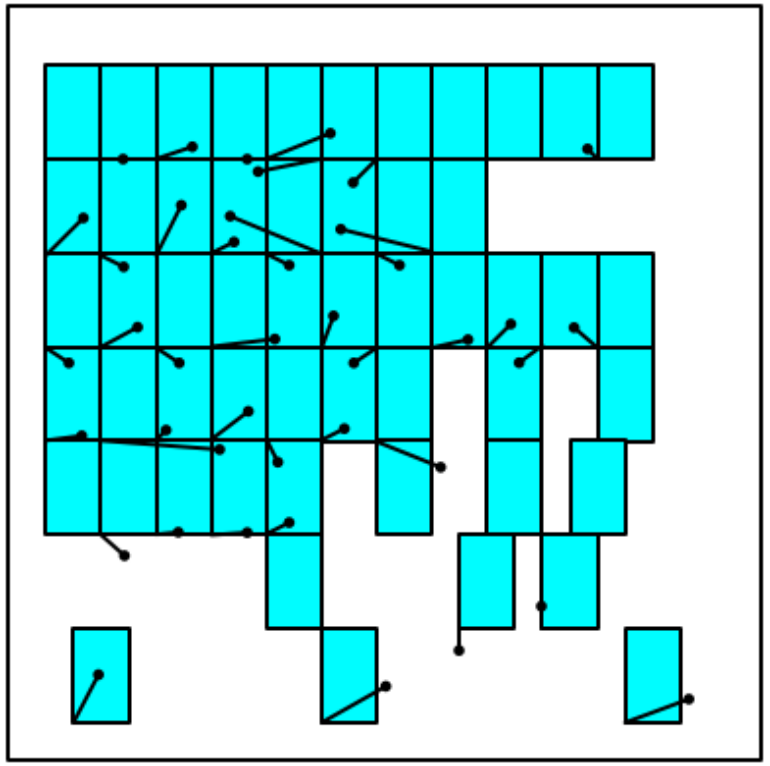
Typical Placement Flow

WL: 1.00e+6



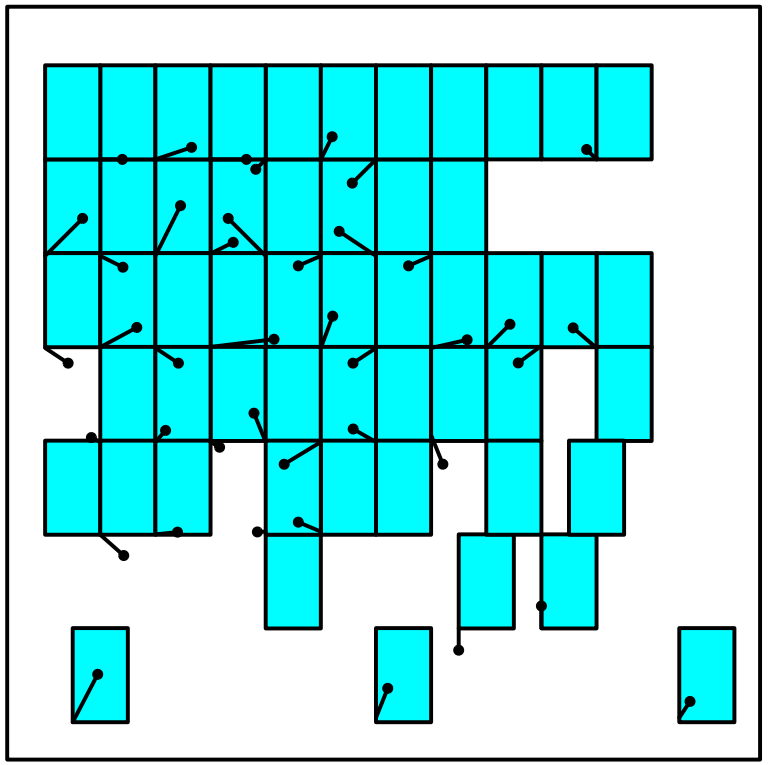
Global placement

WL: 1.05e+6



Legalization

WL: 1.02e+6



Detailed Placement

The History of Placement Algorithms

<1970-1980s	1980s-1990s
Partitioning	Simulated Annealing

- Breuer
- Dunlop & Kernighan
- Quadratic Assignment
- Cadence QPlace

- Timberwolf VPR
- Dragon

Low quality Low efficiency

The History of Placement Algorithms

<1970-1980s	1980s-1990s	1990s-2010s			>2010s	
Partitioning	Simulated Annealing	Min-Cut (Multi-level)	Analytic		Analytic	
			Quadratic	Nonlinear	Quadratic	Nonlinear
Breuer	Timberwolf VPR	FengShui	GORDIAN	APlace	POLAR	ePlace RePlAce
Dunlop & Kernighan	Dragon	Capo	BonnPlace	Naylor Synopsis	SimPL ComPLx	DREAMPlace
Quadratic Assignment		Capo +Rooster	mFar	NTUplace	MAPLE	
Cadence QPlace			Kraftwerk	mPL6		
			FastPlace			
			Warp3			

Low quality Low efficiency

Nonlinear Placement

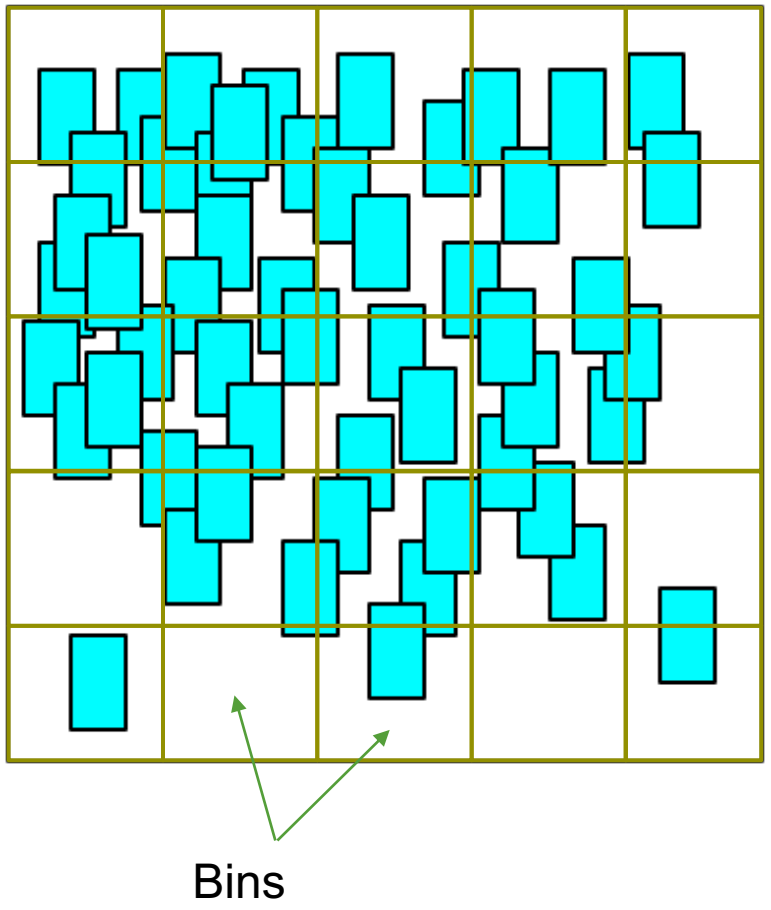
- Mathematical formulation
  - d_i denotes the density of bin i

$$\min_{\boldsymbol{x},\boldsymbol{y}} \; WL(\boldsymbol{x},\boldsymbol{y}),$$
$$s.t. \;\; d_b(\boldsymbol{x},\boldsymbol{y}) \le t_d, \forall b \in Bins$$

- Nonlinear placement objective
  - Lagrangian relaxation

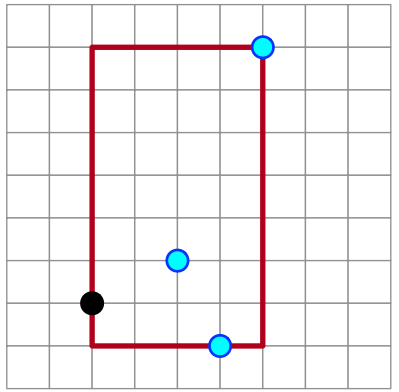
$$\min_{\boldsymbol{x},\boldsymbol{y}} \; WL(\boldsymbol{x},\boldsymbol{y}) + \lambda D(\boldsymbol{x},\boldsymbol{y})$$

Wirelength Density



Wirelength Smoothing

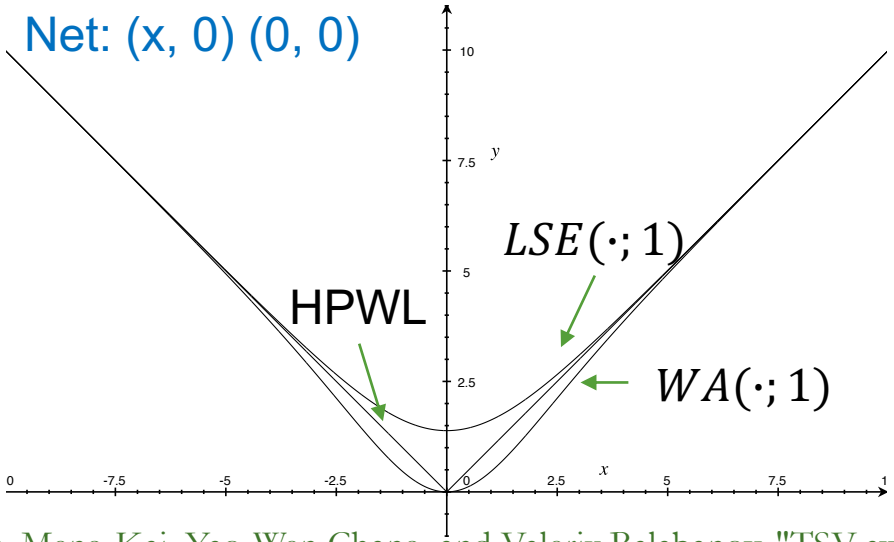
- $WL(\boldsymbol{x},\boldsymbol{y}) = \sum_{e\in E} WL_e(\boldsymbol{x},\boldsymbol{y})$
- $HPWL = max|x_i - x_j| + max\,|y_i - y_j|$
  - Equivalently $\left(\max_i x_i - \min_i x_i\right) + \left(\max_i y_i - \min_i y_i\right)$
- Log-sum-exp (LSE)
  - $LSE(\boldsymbol{x};\gamma) = \gamma \ln \sum_i e^{\frac{x_i}{\gamma}}$
  - $\max\{x_1,\cdots,x_n\} < LSE(\boldsymbol{x};\gamma) \le \max\{x_1,\cdots,x_n\} + \gamma\ln(n)$
  - $LSE(\boldsymbol{x};\gamma) \approx \max\{x_1,\cdots,x_n\}$
  - $-LSE(\boldsymbol{x};-\gamma) \approx \min\{x_1,\cdots,x_n\}$
  - $WL_e(\boldsymbol{x},\boldsymbol{y};\gamma) = \underbrace{\gamma(\ln \sum_{v_i\in e} e^{\frac{x_i}{\gamma}} + \ln \sum_{v_i\in e} e^{-\frac{x_i}{\gamma}}}_{x} + \underbrace{\ln \sum_{v_i\in e} e^{\frac{y_i}{\gamma}} + \ln \sum_{v_i\in e} e^{-\frac{y_i}{\gamma}})}_{y}$

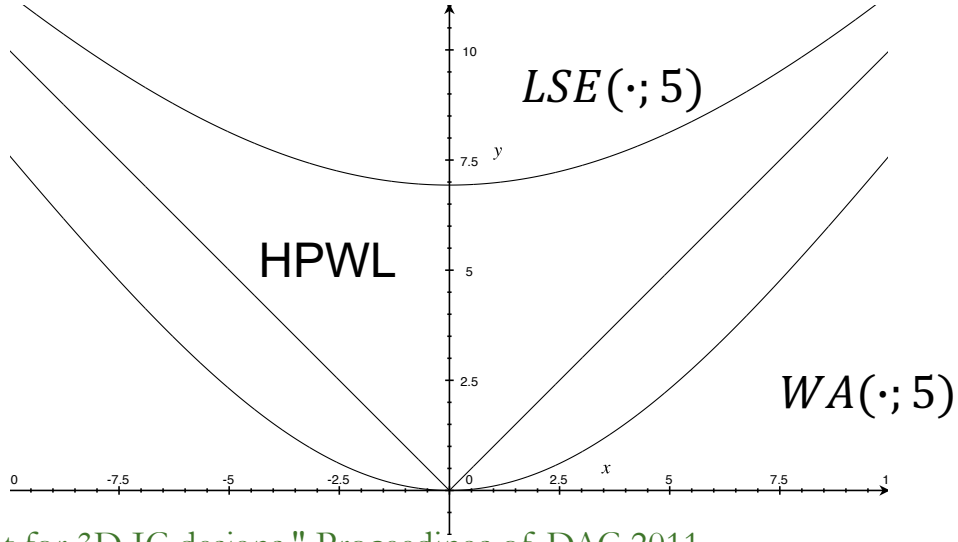


Wirelength Smoothing

- Weighted average (WA)
  - $WL_e(\boldsymbol{x}, \boldsymbol{y}; \gamma) = \underbrace{\left(\frac{\sum_{v_i \in e} x_i e^{x_i/\gamma}}{\sum_{v_i \in e} e^{x_i/\gamma}} - \frac{\sum_{v_i \in e} x_i e^{-x_i/\gamma}}{\sum_{v_i \in e} e^{-x_i/\gamma}}\right)}_{x} + \underbrace{\left(\frac{\sum_{v_i \in e} y_i e^{y_i/\gamma}}{\sum_{v_i \in e} e^{y_i/\gamma}} - \frac{\sum_{v_i \in e} y_i e^{-y_i/\gamma}}{\sum_{v_i \in e} e^{-y_i/\gamma}}\right)}_{y}$
- Larger γ → smoother, but less accurate

More recent work
DAC2019
BiG: Bivariant smoothing





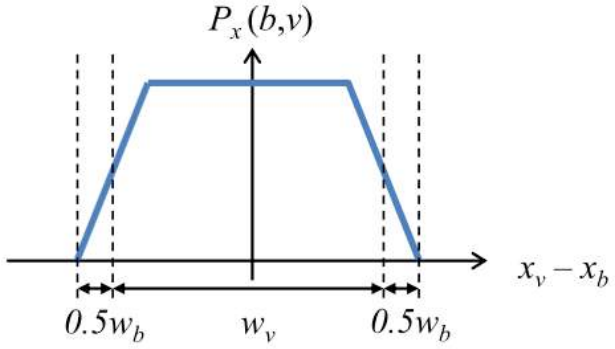
Hsu, Meng-Kai, Yao-Wen Chang, and Valeriy Balabanov. "TSV-aware analytical placement for 3D IC designs." Proceedings of DAC 2011.
Sun, Fan-Keng, and Yao-Wen Chang. "BiG: A bivariate gradient-based wirelength model for analytical circuit placement." Proceedings of DAC 2019.

Nonlinear Placement – NTUplace

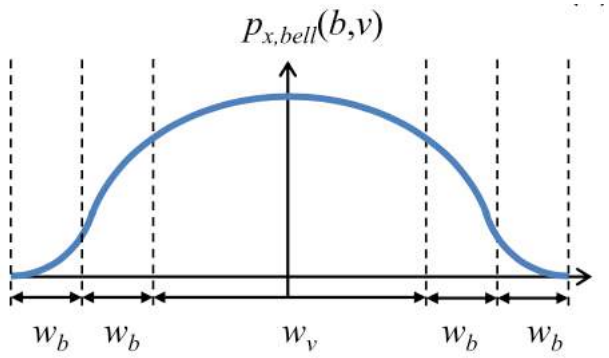
- Chen, Tung-Chieh, et al. "NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs." ISPD 2005
- Chen, Tung-Chieh, et al. "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints." IEEE TCAD 2008.
- Hsu, Meng-Kai, et al. "NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs." IEEE TCAD 2014
- Huang, Chau-Chin, et al. "NTUplace4dr: a detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints." IEEE TCAD 2017

Density Penalty

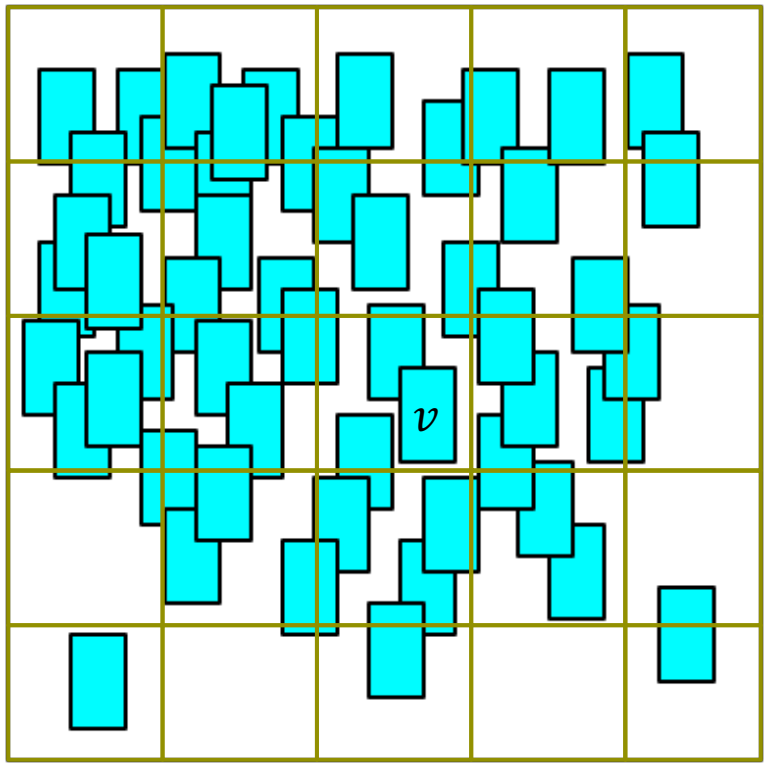
- Potential function for standard cells
  - $P_x(b, v)$ and $P_y(b, v)$ are the overlap functions between bin b and cell v
  - $D_b(\boldsymbol{x}, \boldsymbol{y}) = \sum_{v \in V} P_x(b, v) P_y(b, v)$



Non-smooth
Non-convex

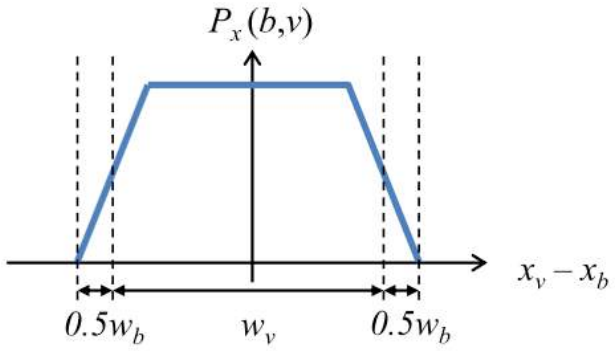


Bell-shape smoothing

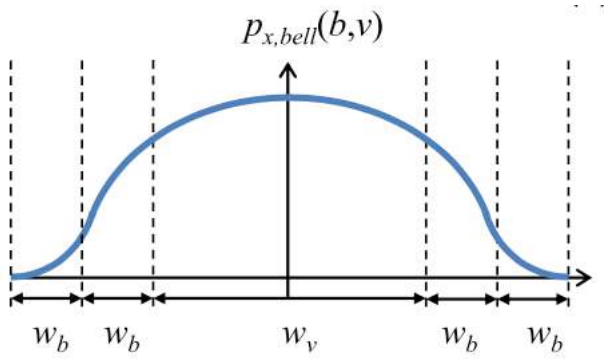


Density Penalty

- Potential function for standard cells
  - $P_x(b,v)$ and $P_y(b,v)$ are the overlap functions between bin b and cell v
  - $D_b(\boldsymbol{x},\boldsymbol{y}) = \sum_{v \in V} P_x(b,v) P_y(b,v)$

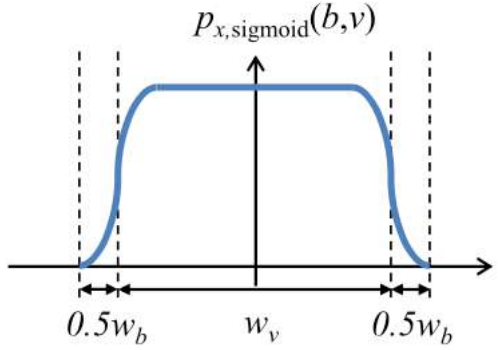


Non-smooth
Non-convex



Bell-shape smoothing

Sigmoid smoothing
[NTUplace4h, TCAD2014]



If $d_x \leq \frac{w_v}{2} + w_b$,

$$\widehat{P}_x(b,v) = 1 - ad_x^2$$

If $\frac{w_v}{2} + w_b \leq d_x \leq \frac{w_v}{2} + 2w_b$,

$$\widehat{P}_x(b,v) = b\left(d_x - \frac{w_v}{2} - 2w_b\right)^2$$

Otherwise,

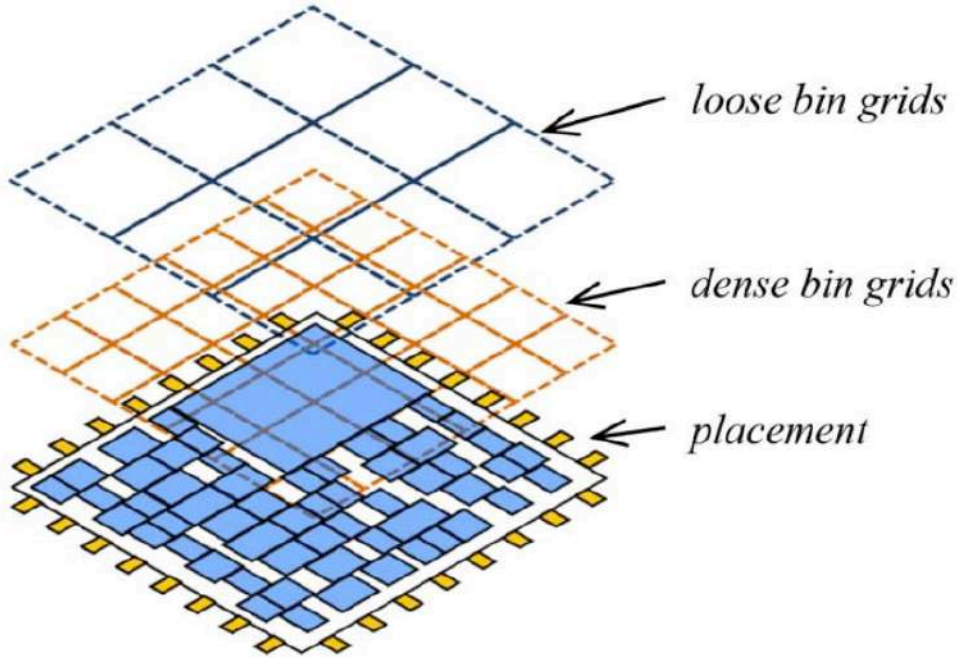
$$\widehat{P}_x(b,v) = 0$$

Density Penalty

- Potential function for standard cells
  - Smoothed potential function
  - $\widehat{D_b}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{v \in V} \widehat{P_x}(b, v) \widehat{P_y}\,(b, v)$
- $\min\limits_{\boldsymbol{x},\boldsymbol{y}} \; WL(\boldsymbol{x}, \boldsymbol{y}) + \lambda\, D(\boldsymbol{x}, \boldsymbol{y})$

$$\lambda \sum_{b} \left(\widehat{D_b}(\boldsymbol{x}, \boldsymbol{y}) - t_d\right)^2$$

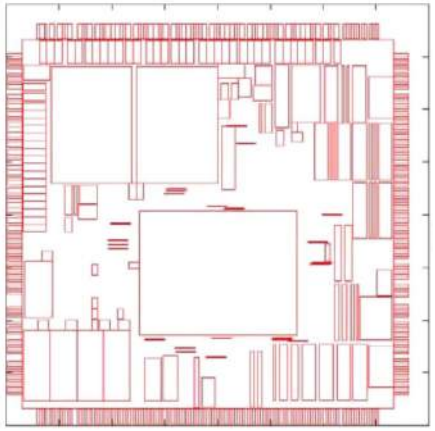
- Challenges
  - Gradient only has local view
  - Need multi-level bins



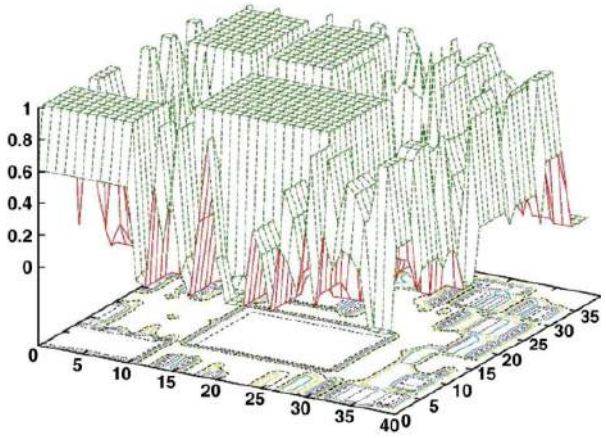
Multi-level bins

Density Penalty – Fixed Macros are Different

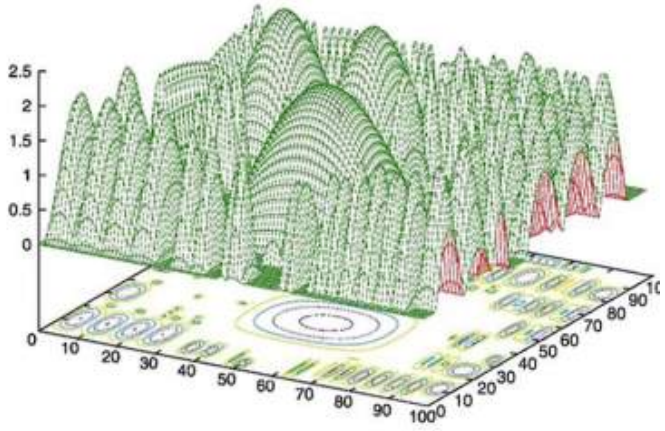
- Potential function for fixed macros
  - Bell-shape smoothing works well for standard cells
  - For fixed macros, $P'(x,y) = G(x,y) * P(x,y)$



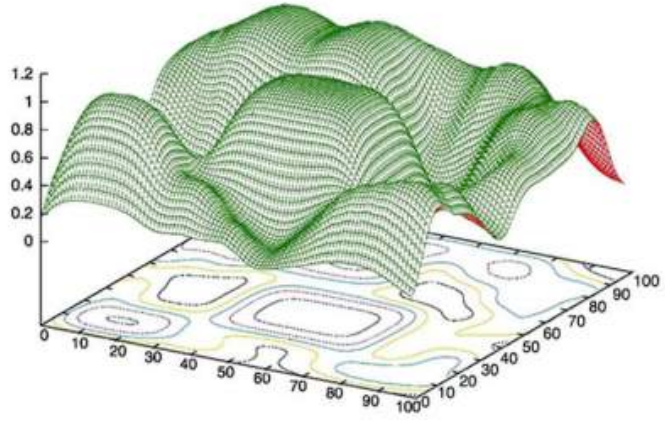
ISPD2005
adaptec2



Exact potential
$P(x,y)$



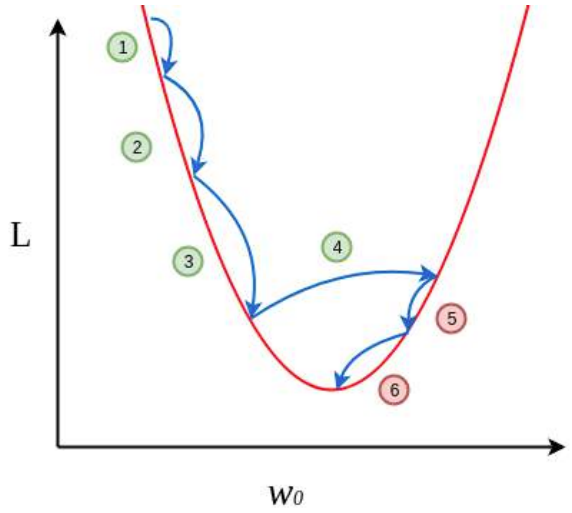
Bell-shape smoothing

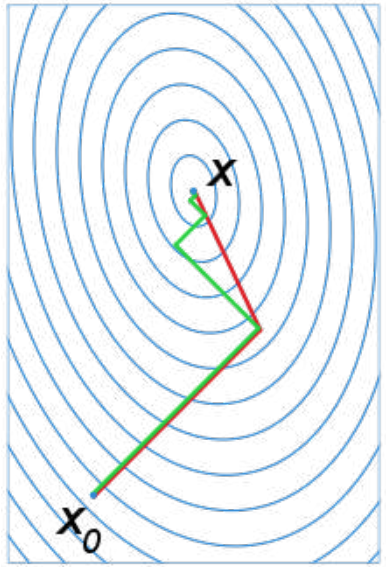


Gaussian smoothing
$P'(x,y)$

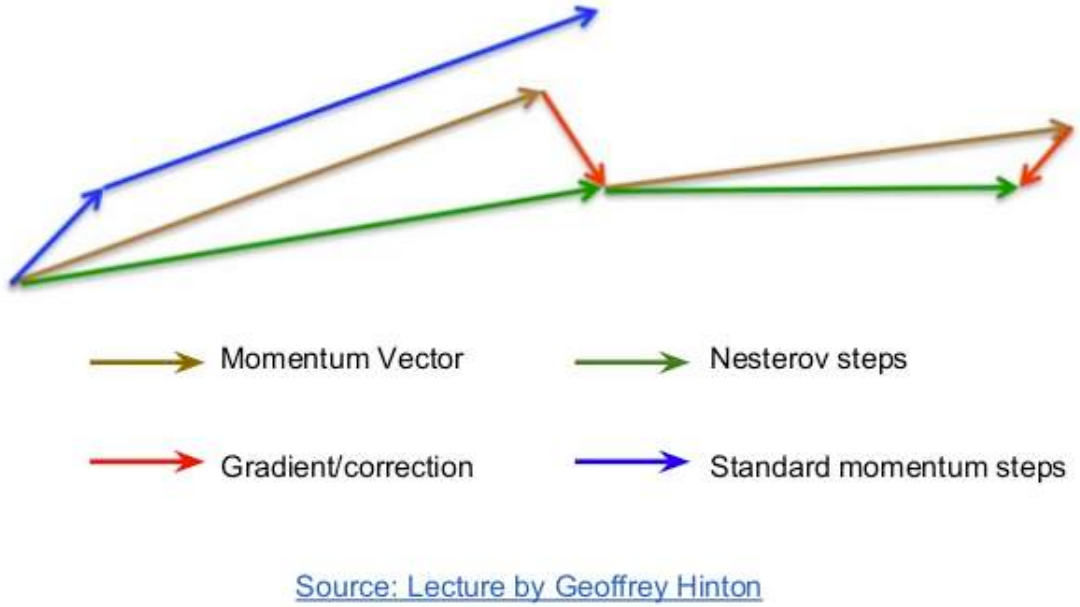
Gradient Descent Solvers for Nonlinear Optimization

- Conjugate gradient (CG) descent
  - Between steepest descent & Newton method
  - Avoid computation of Hessian





- Nesterov's accelerated gradient descent



Source: Lecture by Geoffrey Hinton

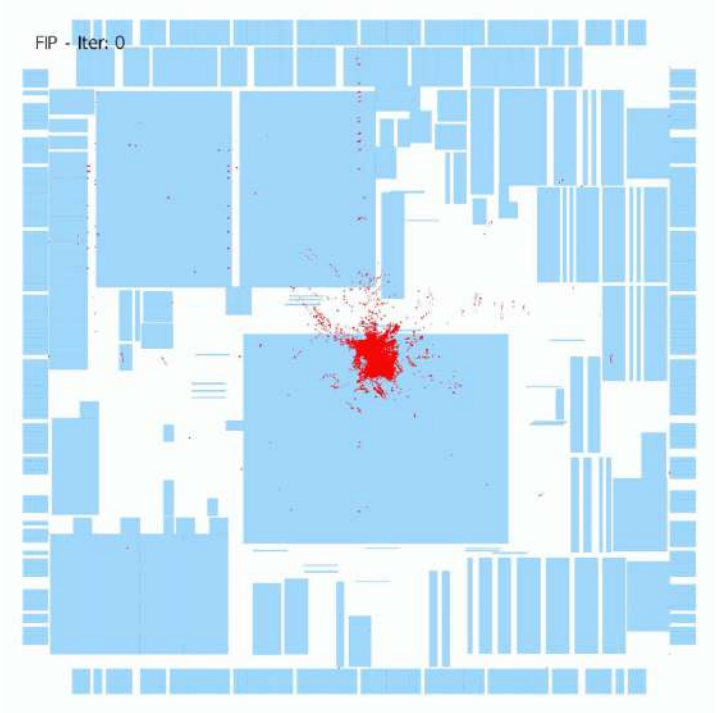
Our Contribution – DREAMPlace

- Placement Engine with Deep Learning Toolkit

Input
Gate-level netlist
Standard cell library

Output
Legal placement solution

Objective
Optimize wirelength, routability



Cell Spreading in Placement

Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

Nonlinear Placement

- Mathematical formulation

$$\min_{\boldsymbol{x},\boldsymbol{y}} \; WL(\boldsymbol{x},\boldsymbol{y}),$$
$$s.t. \;\; d_b(\boldsymbol{x},\boldsymbol{y}) \le t_d, \forall b \in Bins$$

- Nonlinear placement objective

$$\min_{\boldsymbol{x},\boldsymbol{y}} \; \underbrace{WL(\boldsymbol{x},\boldsymbol{y})}_{\text{Wirelength}} + \lambda \underbrace{D(\boldsymbol{x},\boldsymbol{y})}_{\text{Density}}$$

- Challenges
- Low efficiency
  - More than 3h for 10M-cell design
- Limited acceleration
  - Limited speedup, e.g., mPL, due to clustering
- Huge development effort
  - More than 1 year for ePlace/RePlAce

Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

DREAMPlace Solution

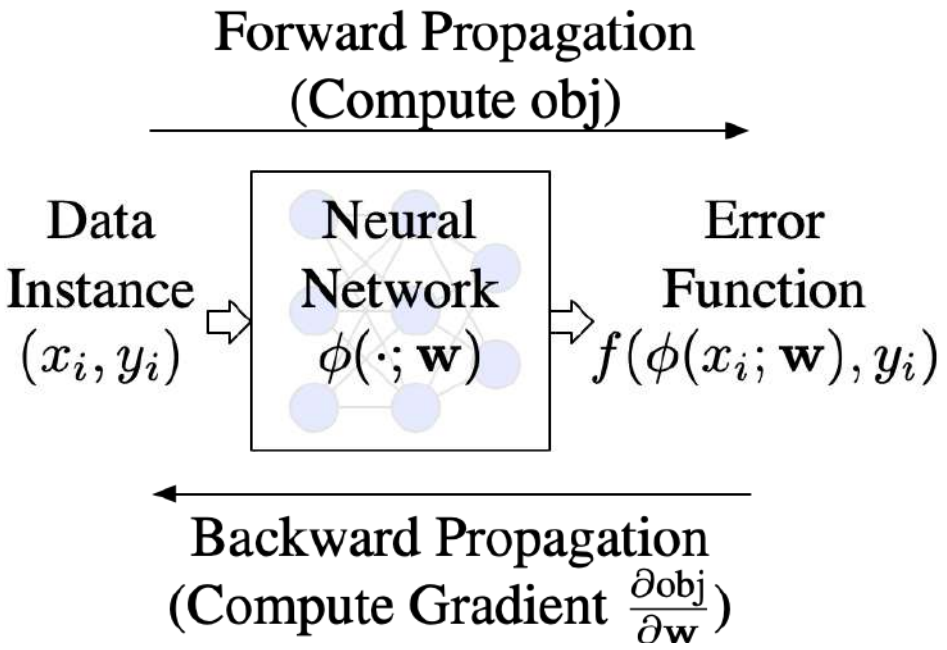
- We propose a novel analogy by casting the **nonlinear placement optimization** into a **neural network training** problem
- Greatly leverage deep learning hardware (GPU) and software toolkit (e.g., PyTorch)
- Enable ultra-high parallelism and acceleration while getting the state-of-the-art results

Best Paper Awards @ DAC & TCAD

Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

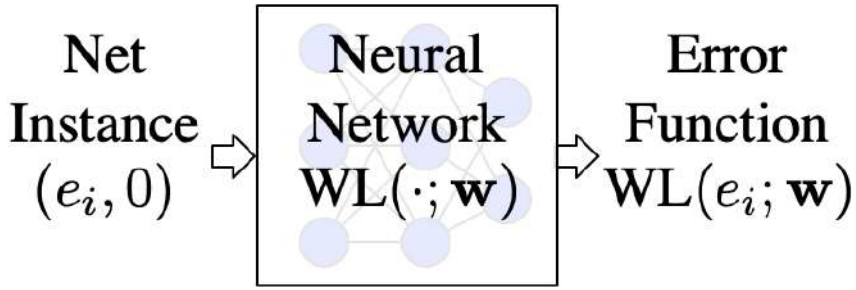
Analogy between Neural Network Training and Placement

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



Train a neural network

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

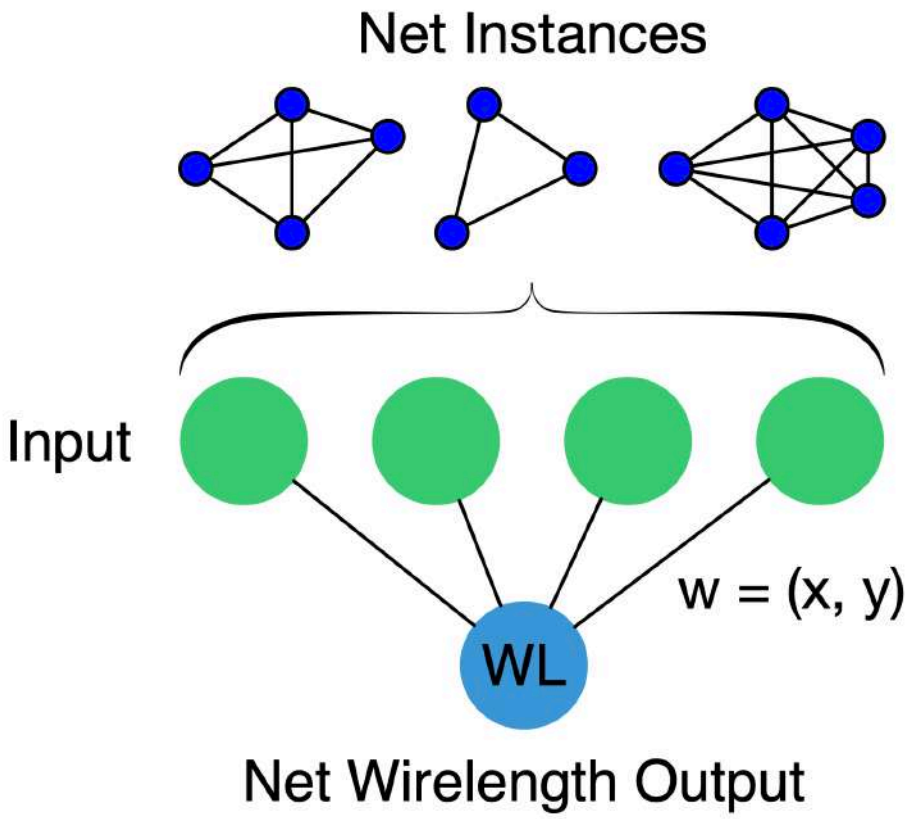


Solve a placement

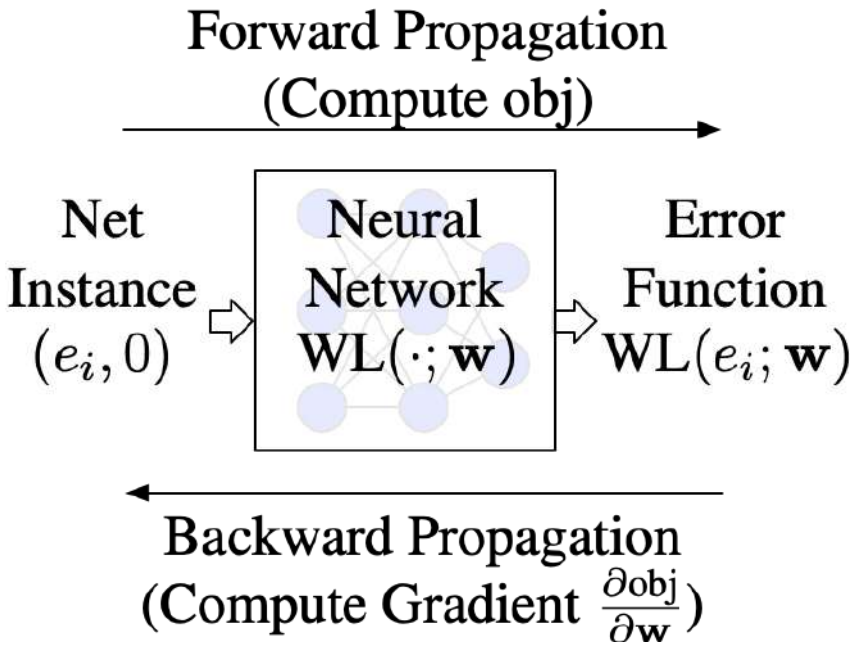
Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

Analogy between Neural Network Training and Placement

Casting the placement problem into neural network training



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

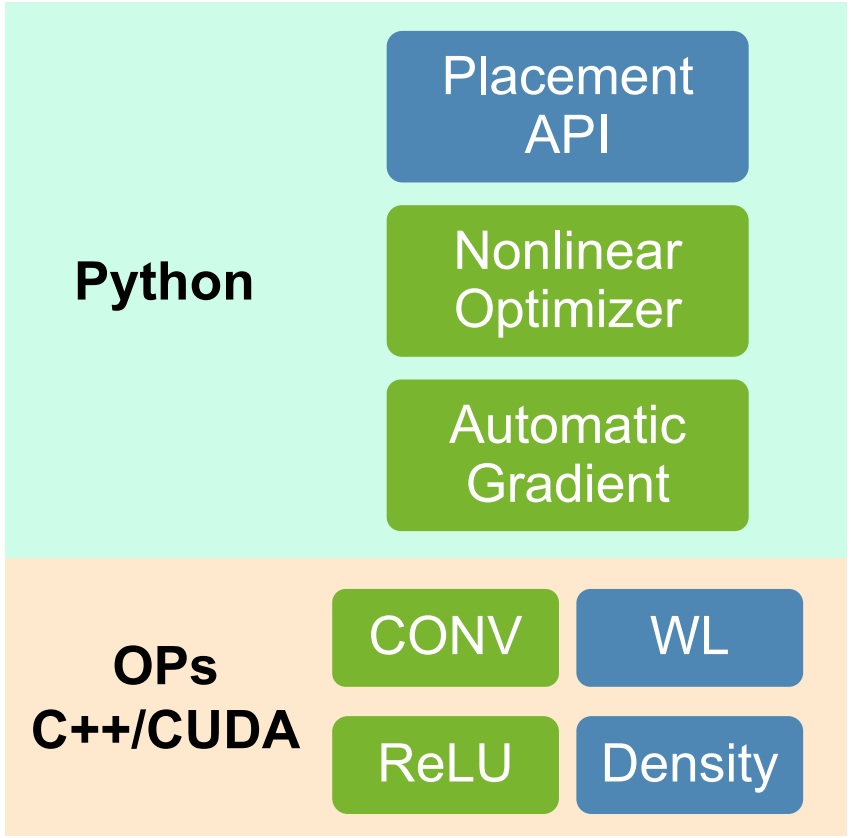


Train a neural network ⟺ **Solve a placement**

Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

Develop Placement Engine with Deep Learning Toolkit

- Leverage highly optimized deep learning toolkit PyTorch



Match RePlAce

Nesterov's Method

[TCAD'18,Cheng+]

Customized CUDA Kernels

Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement." DAC 2019 & TCAD 2020.

Experimental Results

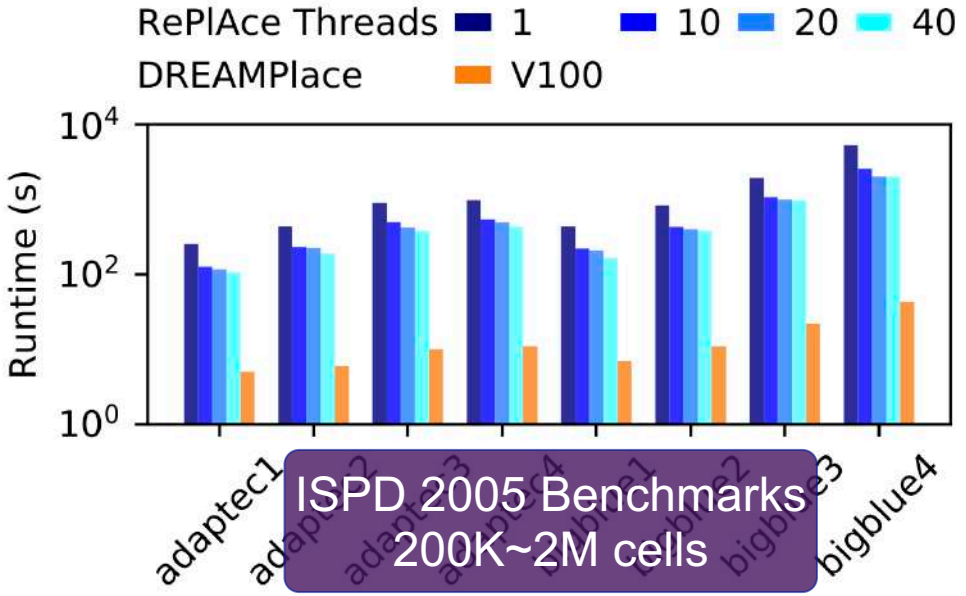
DREAMPlace
- CPU: Intel E5-2698 v4 @2.20GHz
- GPU: 1 NVIDIA Tesla V100
- Single CPU thread was used

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

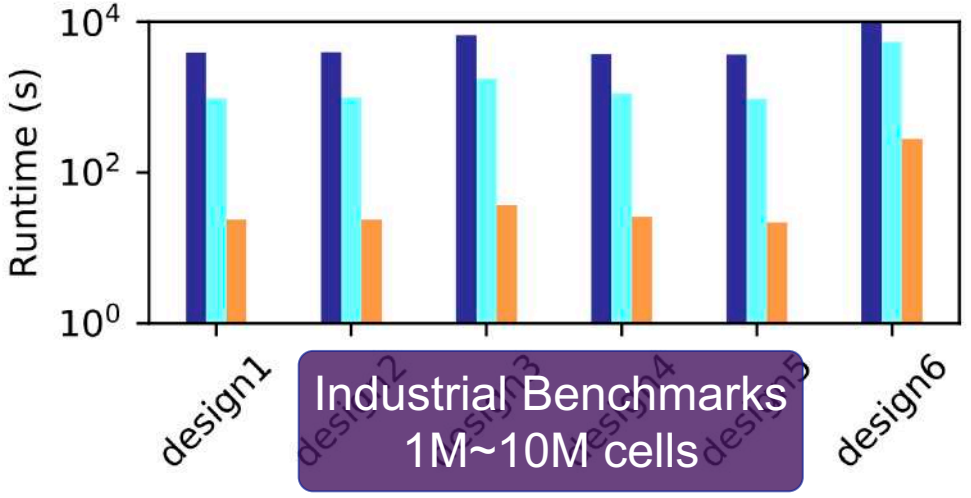
Same quality of results!

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

34x speedup



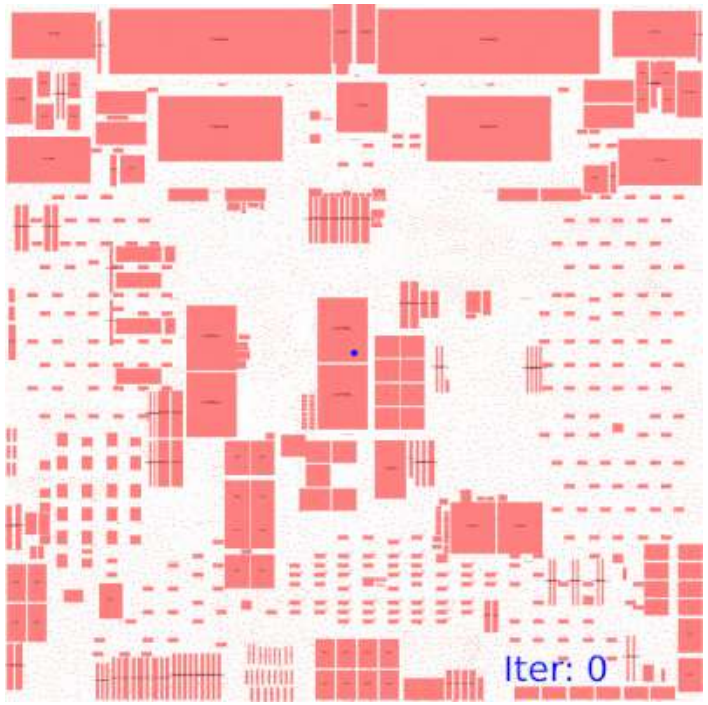
43x speedup

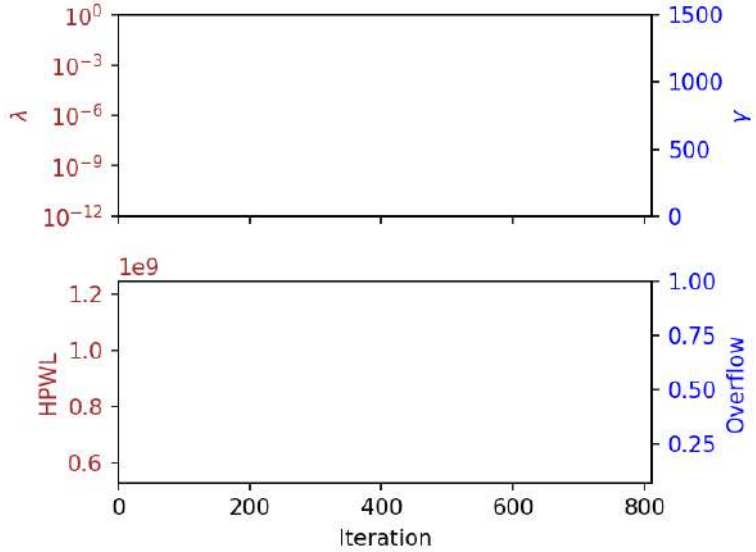


Lin, Yibo, et al. "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Mo

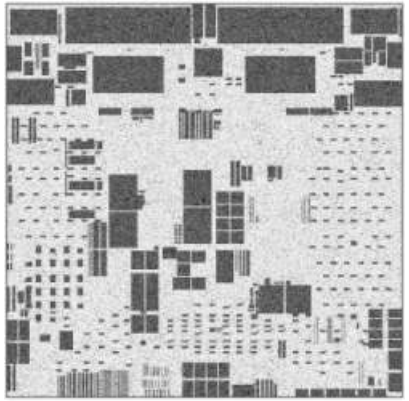
Bigblue4 (2M-Cell Design)

DREAMPlace impl. of the ePlace algorithm

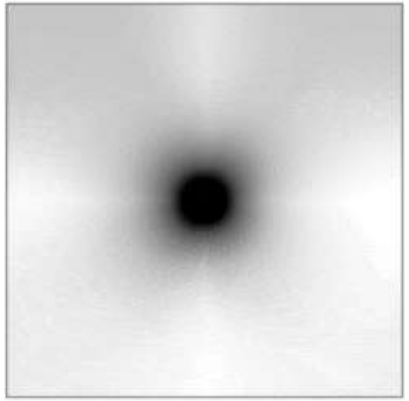




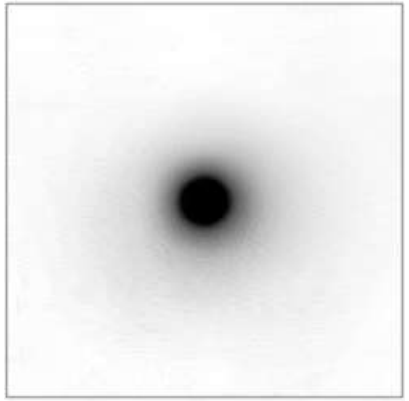
Placement Metrics



Density Map



Potential Map



Field Map

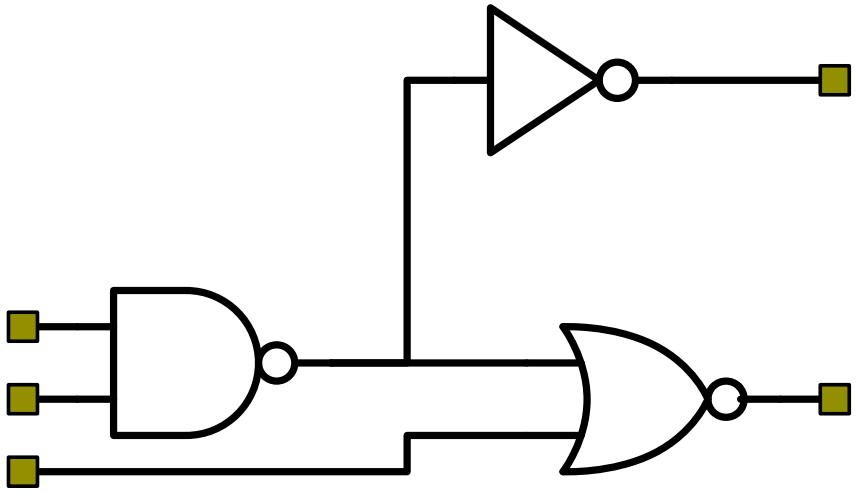
Nonlinear Placement – Summary

- Placement formulation
  - Minimization of two terms: wirelength and density
- Nonlinear placement
  - Wirelength smoothing: LSE and WA
  - Density smoothing: bell-shape function
  - Solve nonlinear optimization with gradient descent
- Open-source tools
  - DREAMPlace
  - https://github.com/limbo018/DREAMPlace
  - RePlAce
  - https://github.com/The-OpenROAD-Project/RePlAce

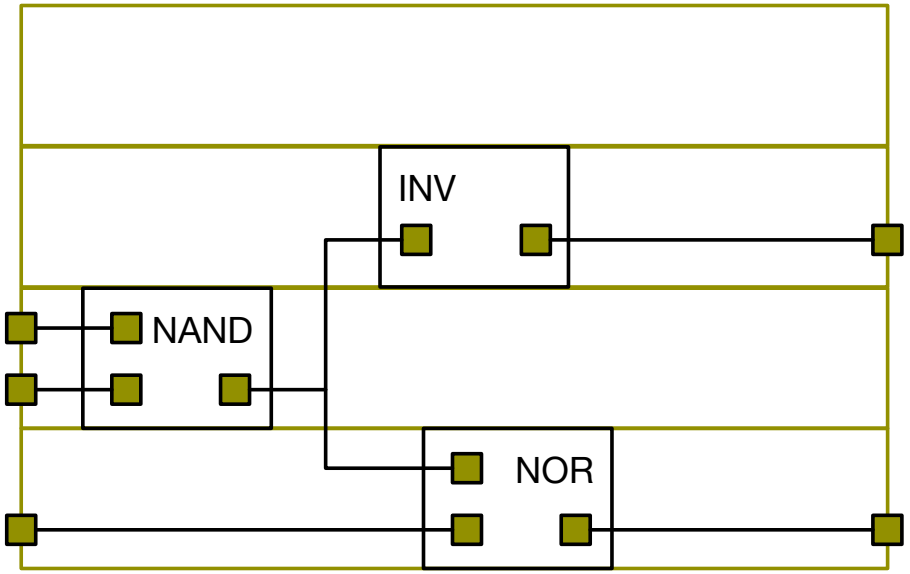
课后思考

- 北京大学计划筹建新校区，请你帮忙规划新校区的建筑分布
  - 已知宿舍楼50幢、食堂10个、教学楼10幢、绿地20块
  - 假设各建筑形状为矩形，且已知大小
  - 假设新校区形状如下图
  - 尝试利用Nonlinear Placement框架设计算法流程求解建筑位置
  - **要求1**：宿舍楼、教学楼和食堂尽可能靠近，且不同宿舍楼、教学楼尽量靠近不同的食堂
  - **要求2**：不同建筑不能重叠
  - **要求3**：写出优化目标、约束条件以及算法流程，并解释设计理由

What is Routing

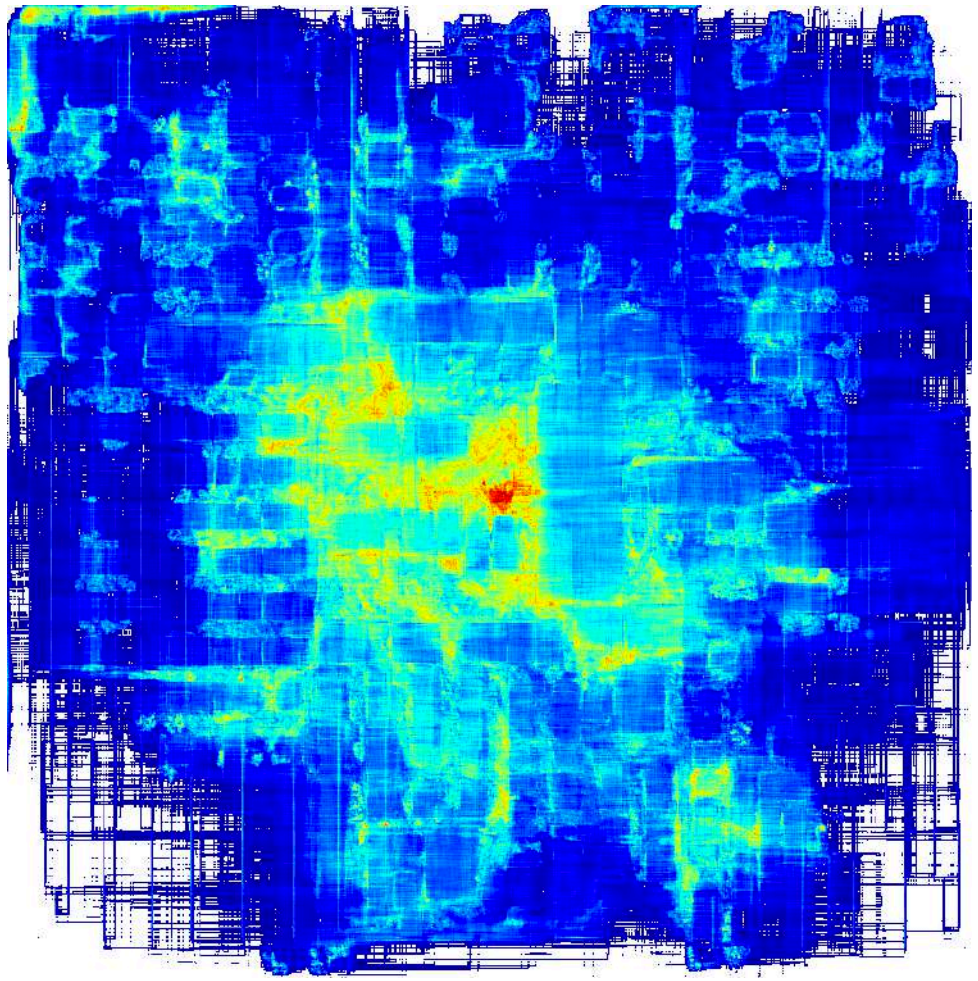


Netlist

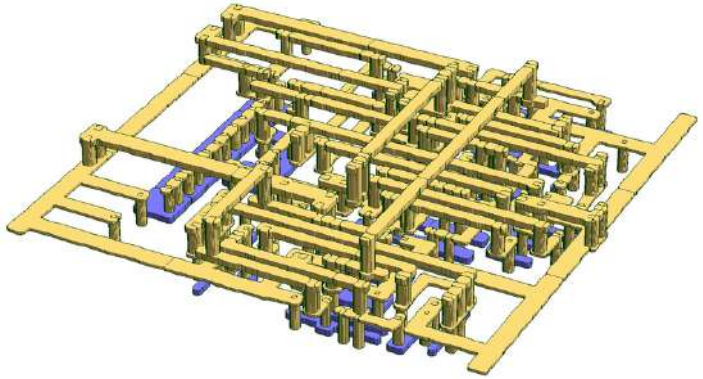


Placement and Routing

What is Routing



[Curtesy Umich]



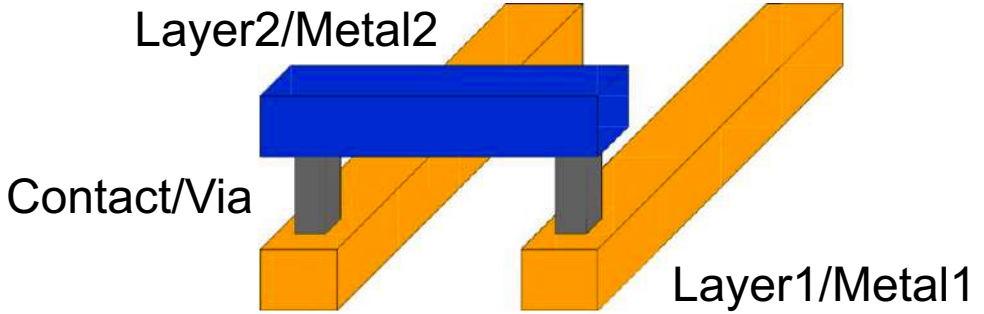
A zoom-in 3D view
[curtesy samyzaf]

Challenging problem

- 10+ metal layers
- Millions of nets
- May be highly congested
- Minimize wirelength

Routing Problem Formulation

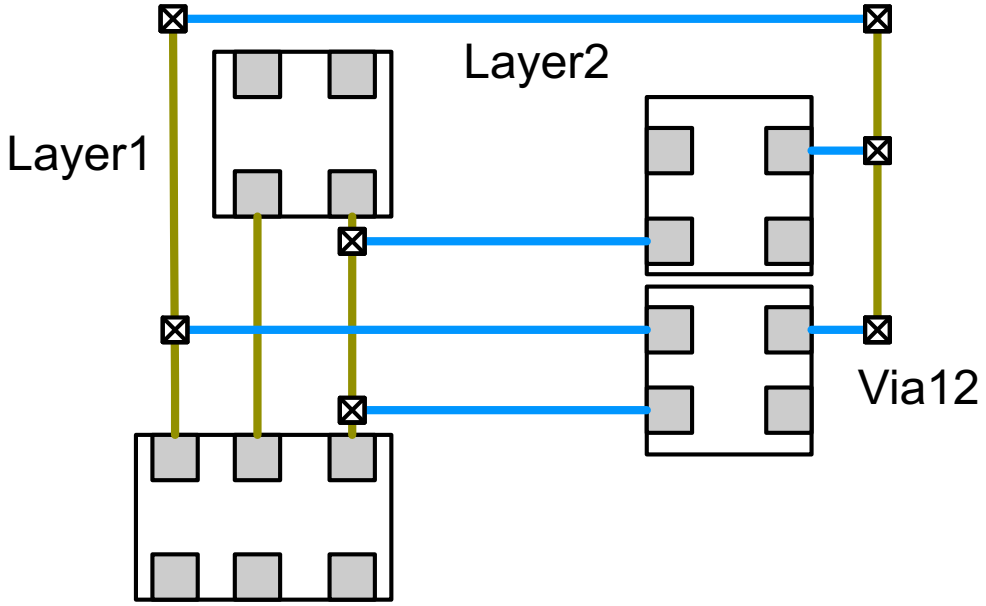
- Apply it after placement
- Input
  - Netlist
  - Timing budget for, typically, critical nets
  - Locations of blocks and locations of pins

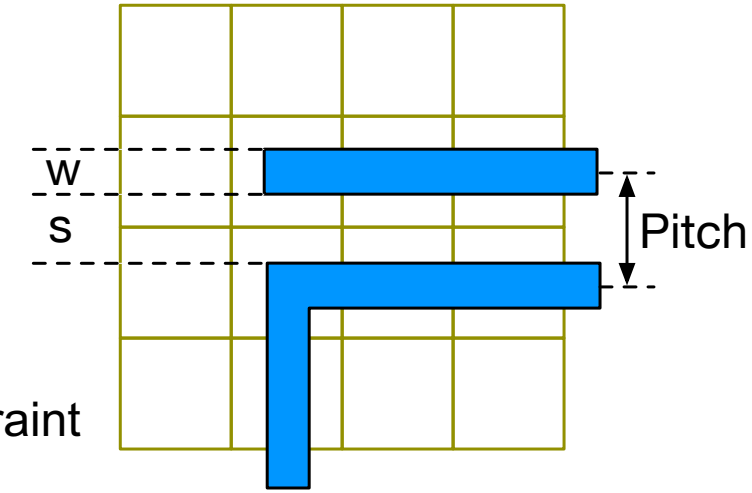


- Output
  - Geometric layouts of all nets
- Objective
  - Minimize the total wire length, the number of vias, or just completing all connections without increasing the chip area.
  - Each net meets its timing budget

The Routing Constraints

- Placement constraint
- Number of routing layers
- Delay constraint
- Meet all geometrical constraints (design rules)
- Physical/Electrical/Manufacturing constraints:
  - Crosstalk
  - Process variations, yield, or lithography issues?



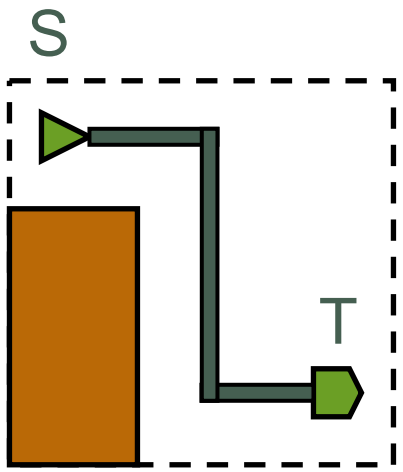


Geometrical constraint

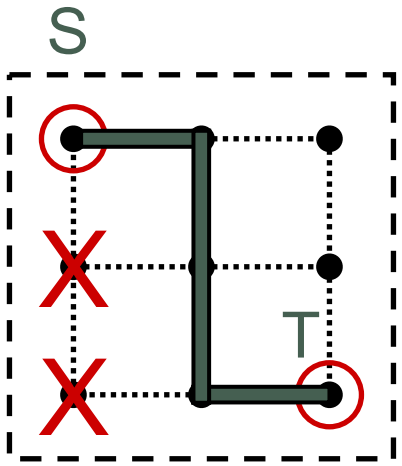
Maze Routing Problem

- Input
  - A planar rectangular grid graph.
  - Two points S and T on the graph.
  - Obstacles modeled as blocked vertices.
- Objective
  - Find the shortest path connecting S and T.
- This technique can be used in global or detailed routing problems.

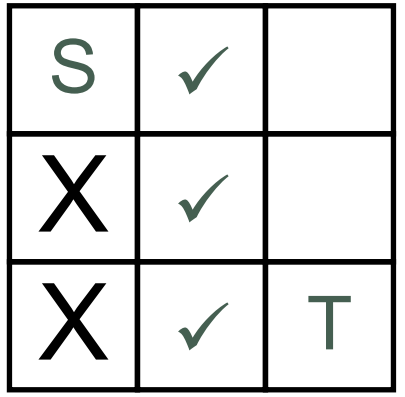
Grid Graph



Area Routing

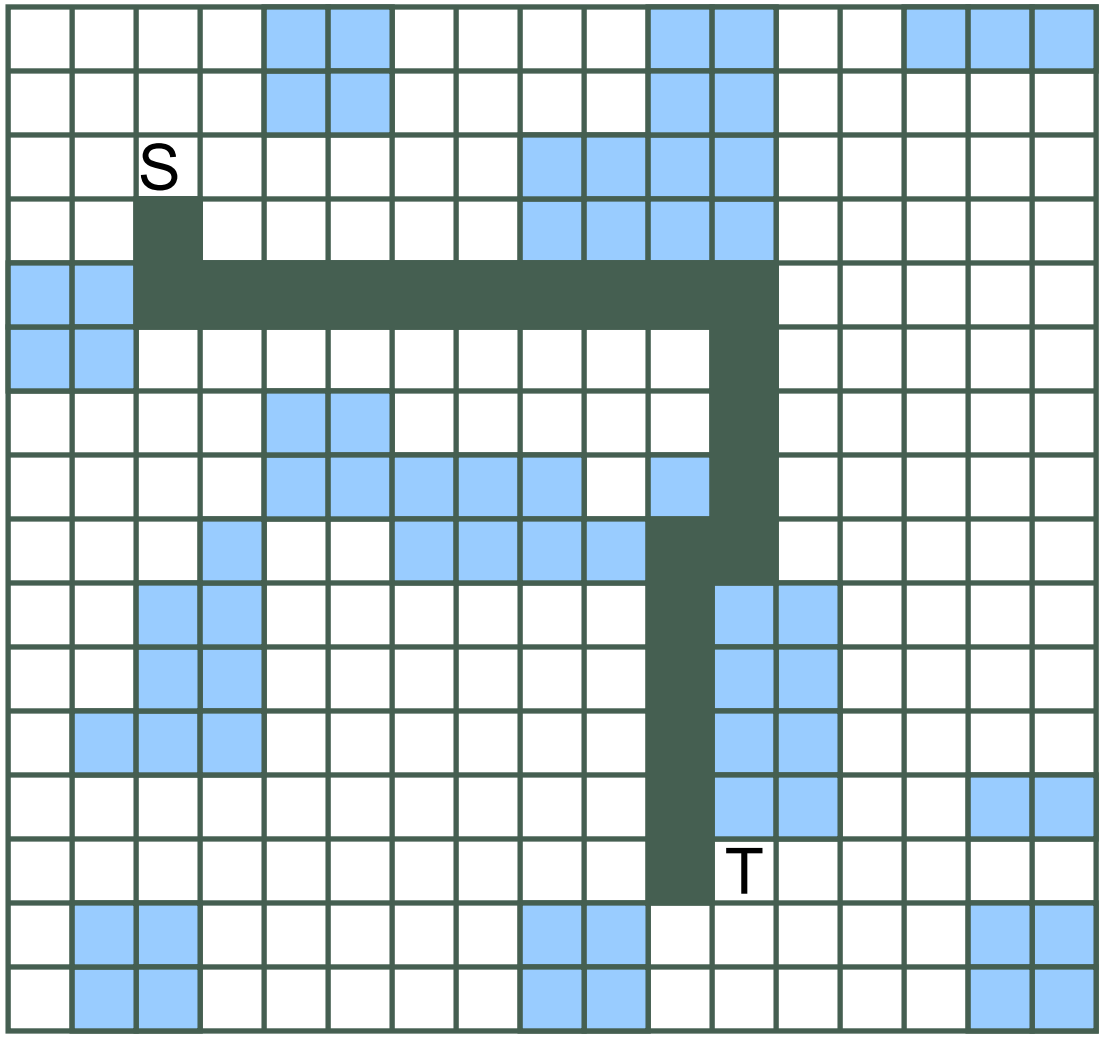


Grid Graph
(Maze)



Simplified
Representation

Maze Routing



Lee's Algorithm

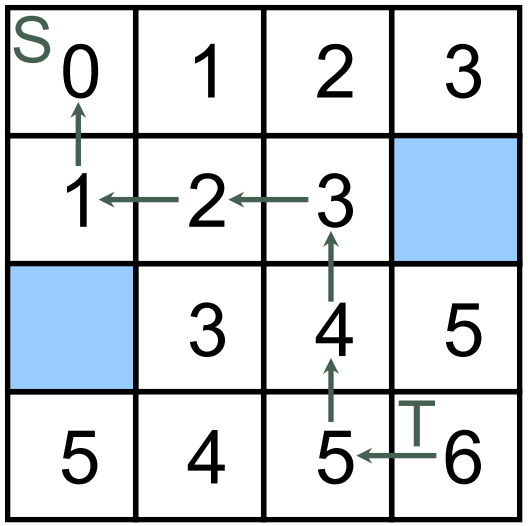
- Basic idea
- A Breadth-First Search (BFS) of the grid graph.
- Always find the shortest path possible.
- Consists of two phases:
  - Wave Propagation
  - Retrace

S 0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

Lee, Chin Yang. "An algorithm for path connections and its applications." IRE transactions on electronic computers 3 (1961): 346-365.

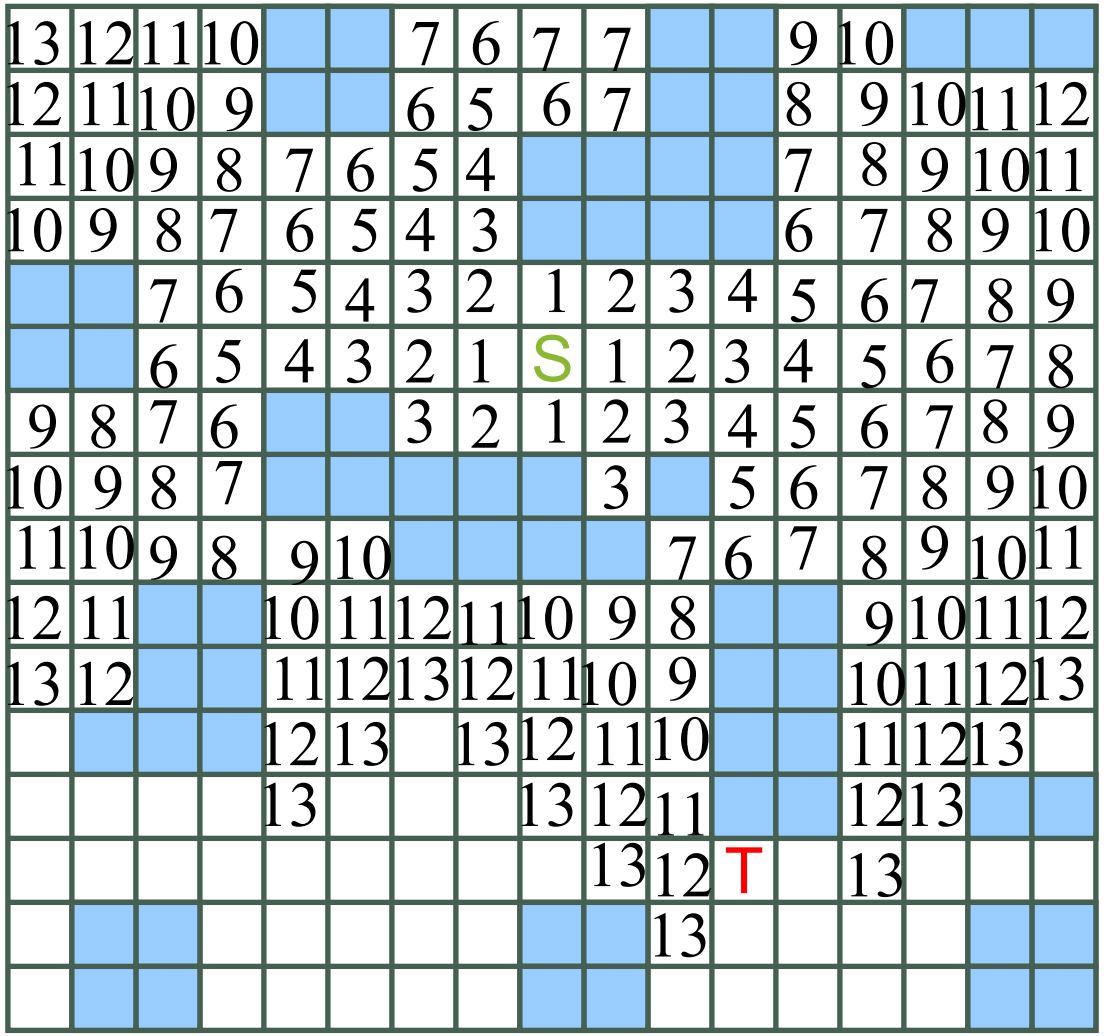
Retrace

- Trace back the actual route.
- Starting from *T*.
- At *vertex with k, go to any vertex with label k*-1.



Final labeling

How many grids visited using Lee's algorithm?



Time and Space Complexity

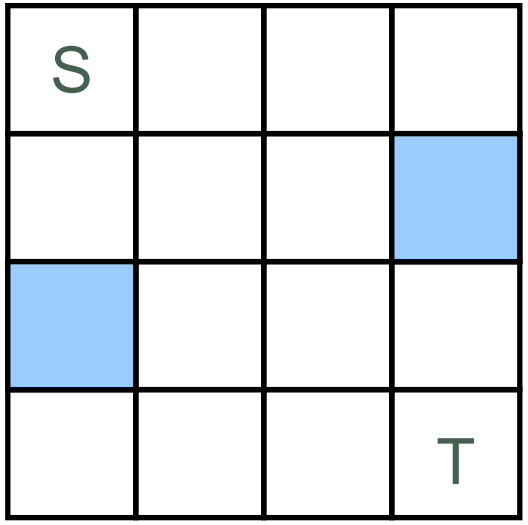
- For a grid structure of size $w \times h$:
  - Time per net = O(wh)
  - Space = O(wh log wh) (O(log wh) bits are needed to store each label.)
- For a 4000 $\times$ 4000 grid structure:
  - 24 bits per label
  - Total 48 Mbytes of memory!

Improvement to Lee's Algorithm

- Improvement on memory:
  - Aker's Coding Scheme
- Improvement on run time:
  - Starting point selection
  - Double fan-out
  - Framing
  - Hadlock's Algorithm
  - Soukup's Algorithm

Aker's Coding Scheme to Reduce Memory Usage

- For the Lee's algorithm, labels are needed during the retrace phase.
- But there are only two possible labels for neighbors of each vertex labeled i, which are, $i-1$ and $i+1$.
- So, is there any method to reduce the memory usage?
- **One bit** (independent of grid size) is enough to distinguish between the two labels.



Sequence:
..... (what sequence?)

(Note: In the sequence, the labels before and after each label must be different in order to tell the forward or the backward directions.)

Akers, Sheldon B. "A modification of Lee's path connection algorithm." *IEEE Transactions on Electronic Computers* 1 (1967): 97-98.

Aker's Coding Scheme to Reduce Memory Usage

- For the Lee's algorithm, labels are needed during the retrace phase.
- But there are only two possible labels for neighbors of each vertex labeled i, which are, $i-1$ and $i+1$.
- So, is there any method to reduce the memory usage?
- **One bit** (independent of grid size) is enough to distinguish between the two labels.

S	0	1	0
0	1	0	
	0	1	0
0	1	0	1T

Correct?

Akers, Sheldon B. "A modification of Lee's path connection algorithm." *IEEE Transactions on Electronic Computers* 1 (1967): 97-98.

Aker's Coding Scheme to Reduce Memory Usage

- For the Lee's algorithm, labels are needed during the retrace phase.
- But there are only two possible labels for neighbors of each vertex labeled i, which are, $i-1$ and $i+1$.
- So, is there any method to reduce the memory usage?
- **One bit** (independent of grid size) is enough to distinguish between the two labels.

S	1	1	0
1	1	0	
	0	0	1
1	0	1	1T

Akers, Sheldon B. "A modification of Lee's path connection algorithm." *IEEE Transactions on Electronic Computers* 1 (1967): 97-98.

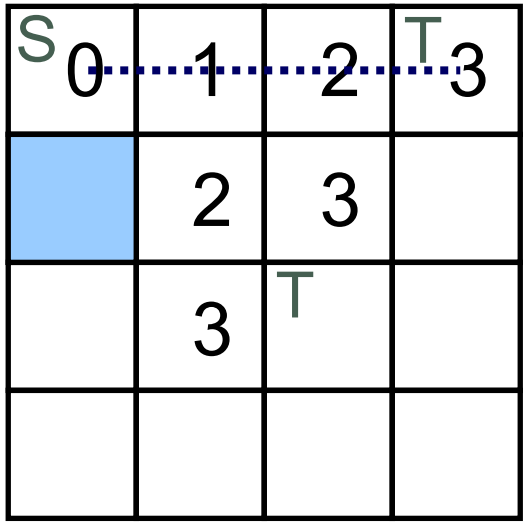
Multi-Pin Nets

- For a k-pin net, connect the k pins using a rectilinear Steiner tree with the shortest wire length on the maze.
- This problem is NP-Complete.
- Just want to find some good heuristics.

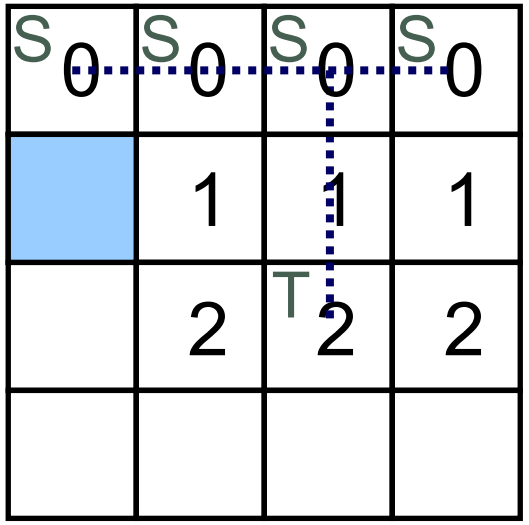
- This problem can be solved by extending the Lee's algorithm:
  - Connect one pin at a time, or
  - Search for several targets simultaneously, or
  - Propagate wave fronts from several different sources simultaneously.

Extension to Multi-Pin Nets

1st Iteration

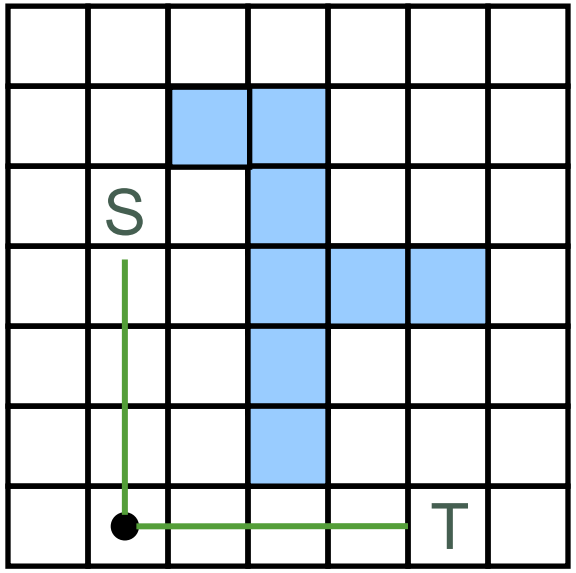


2nd Iteration

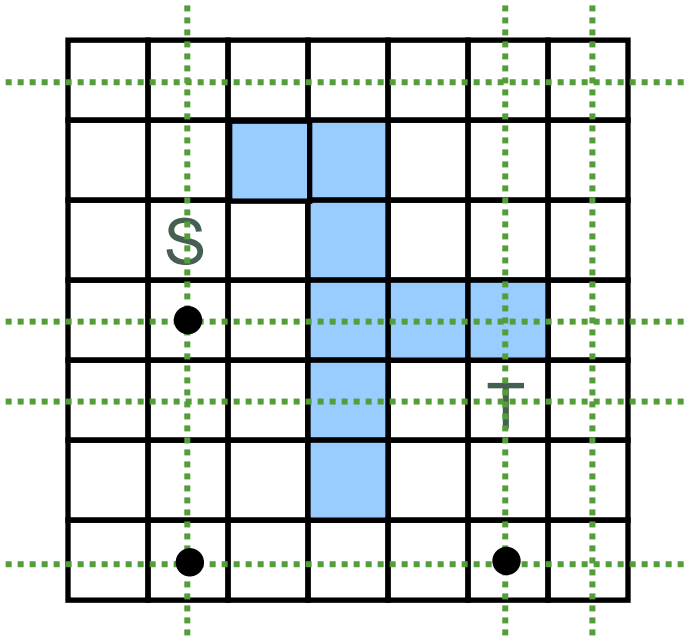


Speedup Maze Routing

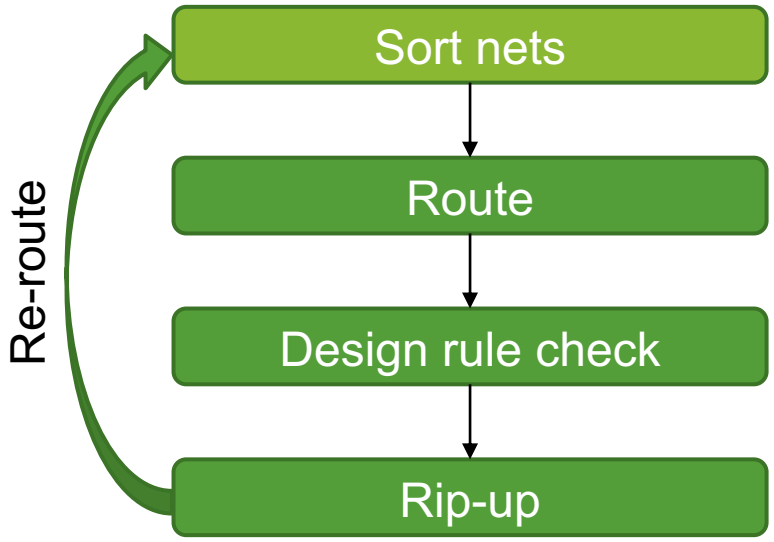
- Pattern routing
  - Most nets are simple, e.g., L-shape
  - Can connect >80% nets
  - Dynamic programming
  - [NCTUgr, TCAD2013]



- Coarsening search steps
  - Only check grids that can make a turn
  - [Dr.CU, ICCAD2019]

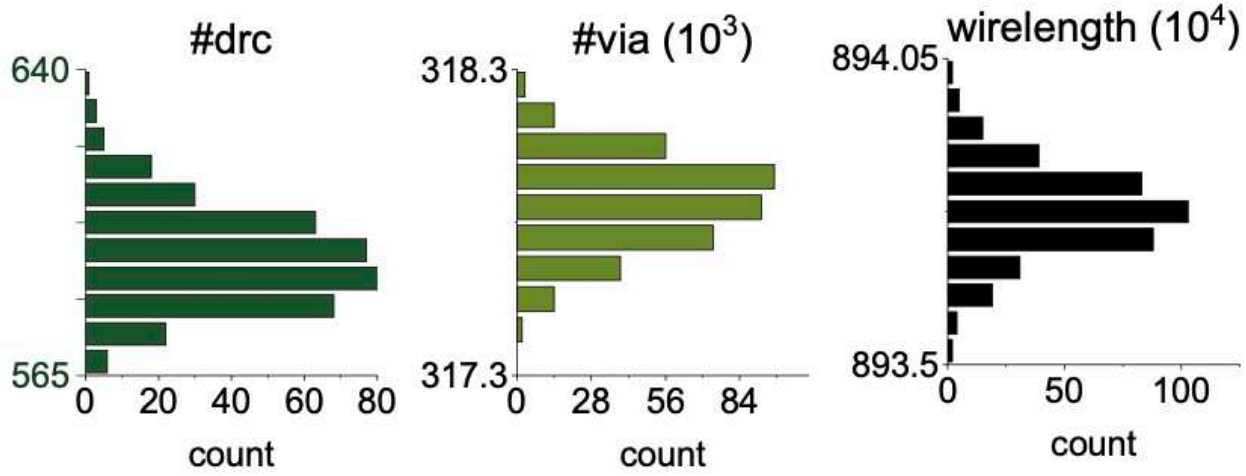


Typical Routing Flow



Dr.CU series

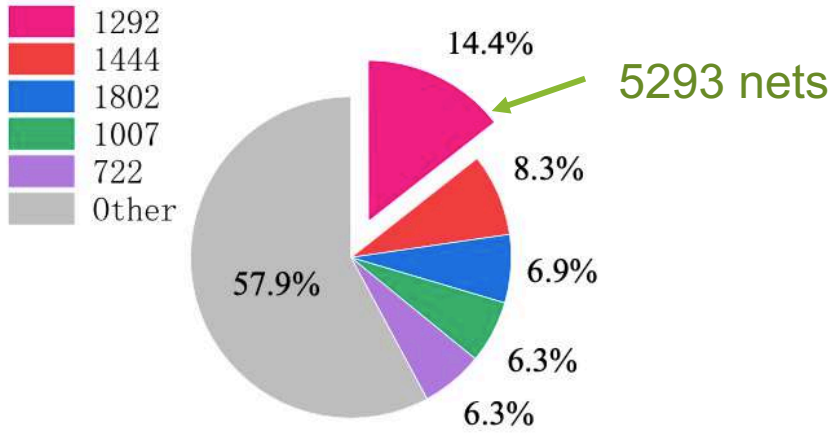
Net Order is critical for sequential routing



Distribution of solution quality with random net ordering (300 iterations).
Their ranges are 70.00, 845.00, and 4996.31

Net Ordering Strategy

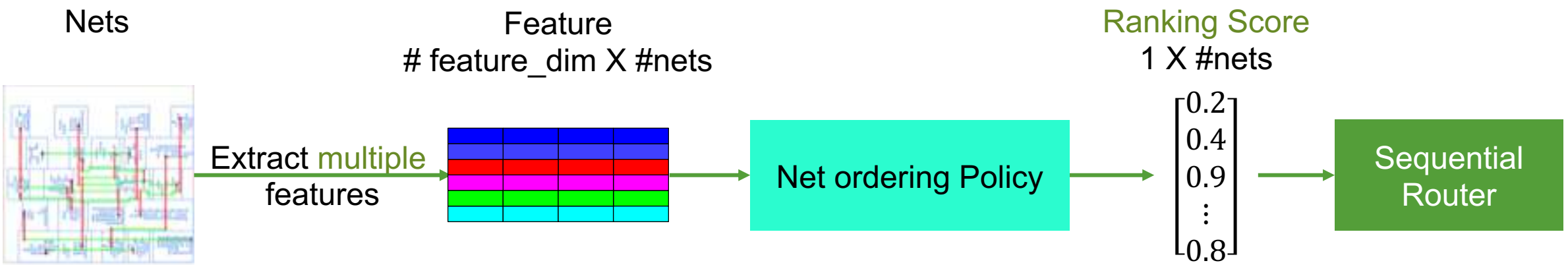
- The ascending order of pin number
- The ascending order of routing region size [Li+, 2019]
- The descending order of DRC number
- …



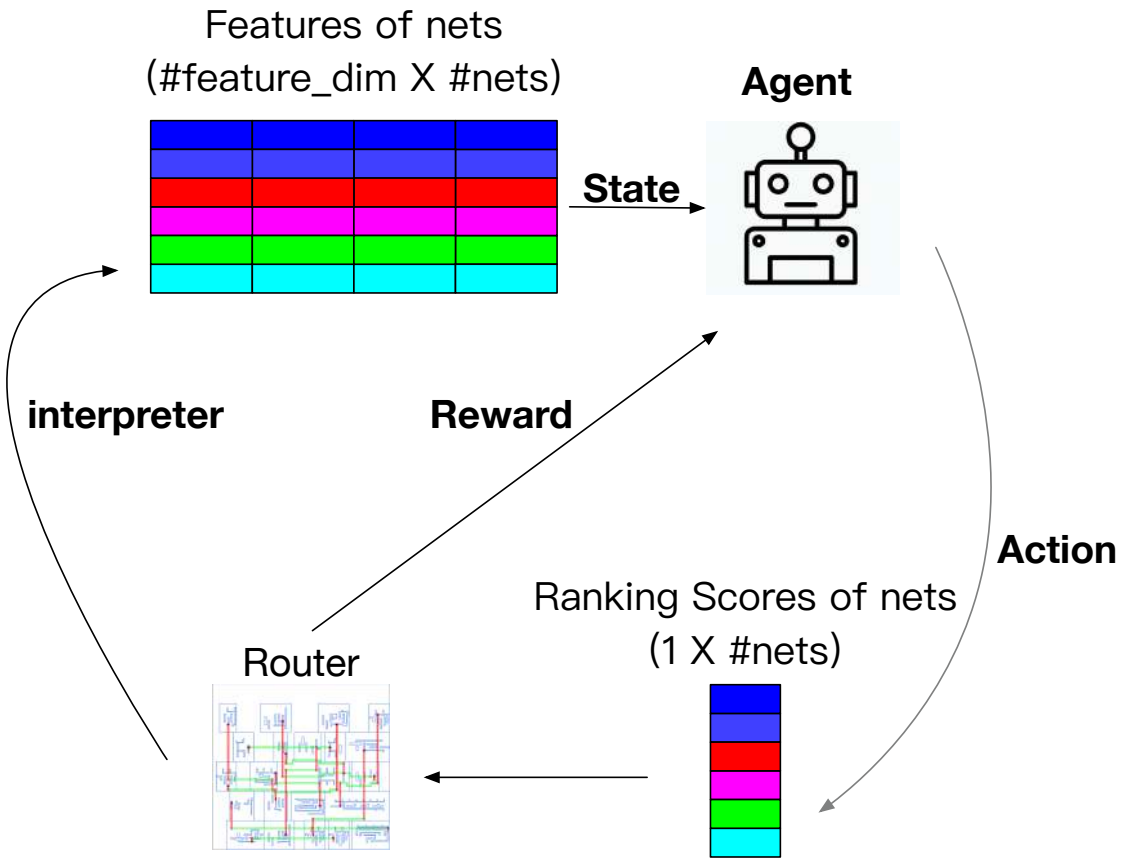
Distribution of the net routing region sizes

Our Contribution – RL to Determine Net Ordering

- Input: a set of nets
- Output: net ordering for a sequential router
- Task: Find an optimal net ordering for minimum cost of routing
  - the total wirelength of all nets
  - the number of the total used vias
  - the number of DRC violations.

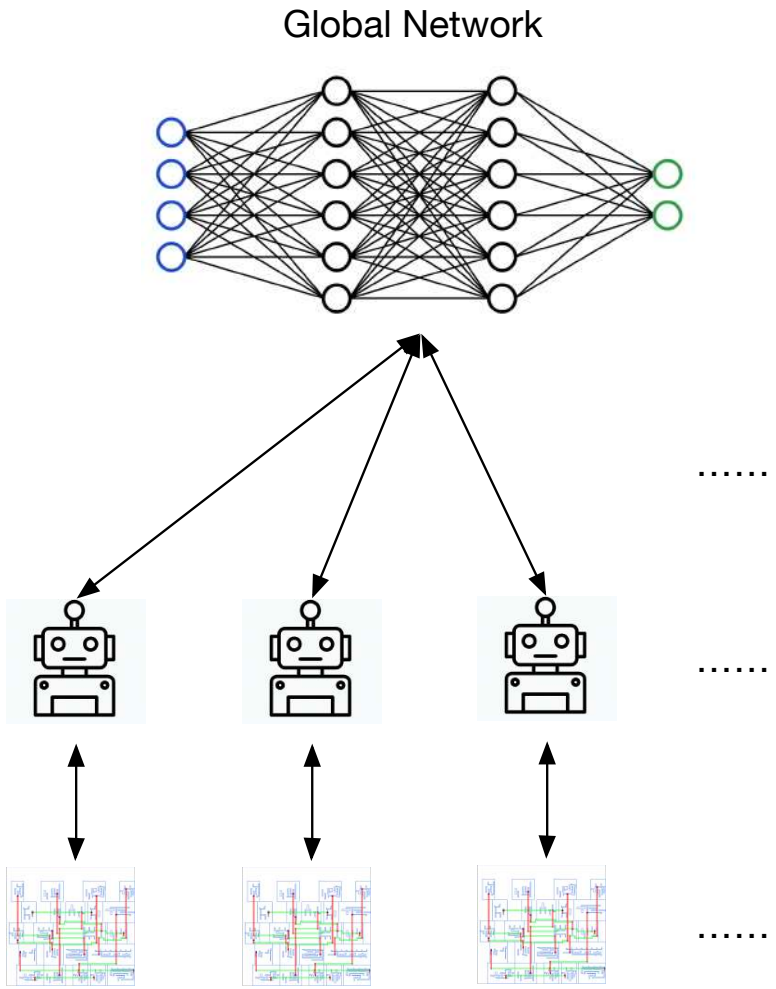


Asynchronous RL Routing Framework



- RL Setup
  - State: Features
  - Action: Ranking score of nets
  - Reward
    - $R = \frac{-(w_1 \cdot \#\text{via} + w_2 \cdot \text{wl} + w_3 \cdot \#\text{drc})}{\text{cost}}$
  - Router: Dr. CU [Li+, 2019]
  - Algorithm: A3C [Mnih+, 2016]

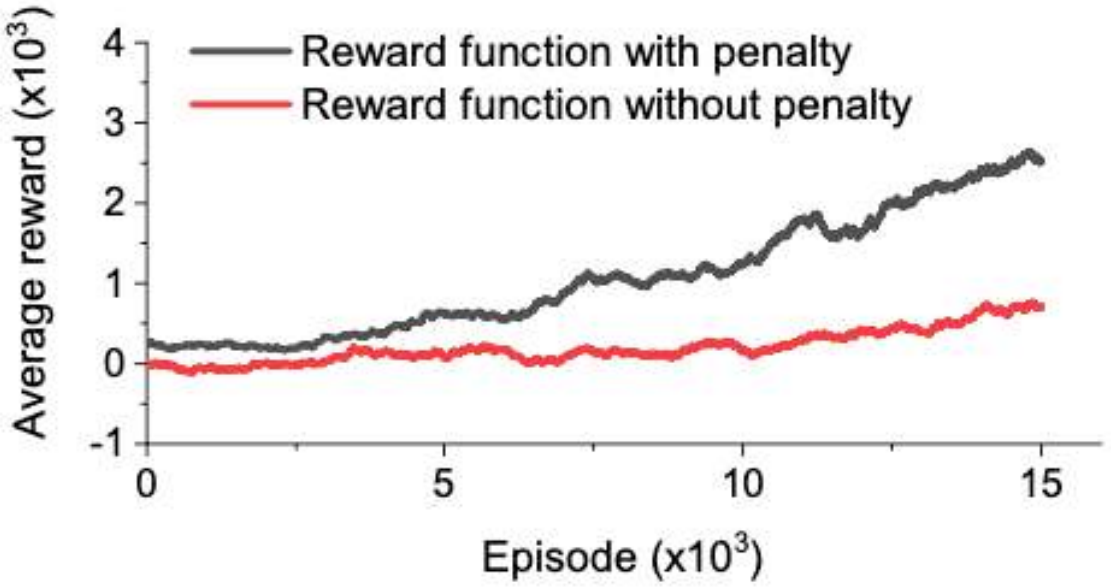
Asynchronous RL Routing Framework



- RL Setup
  - State: Features
  - Action: Ranking score of nets
  - Reward
    - $R = \frac{-(w_1 \cdot \#\text{via} + w_2 \cdot \text{wl} + w_3 \cdot \#\text{drc})}{\text{cost}}$
  - Router: Dr. CU [Li+, 2019]
  - Algorithm: A3C [Mnih+, 2016]

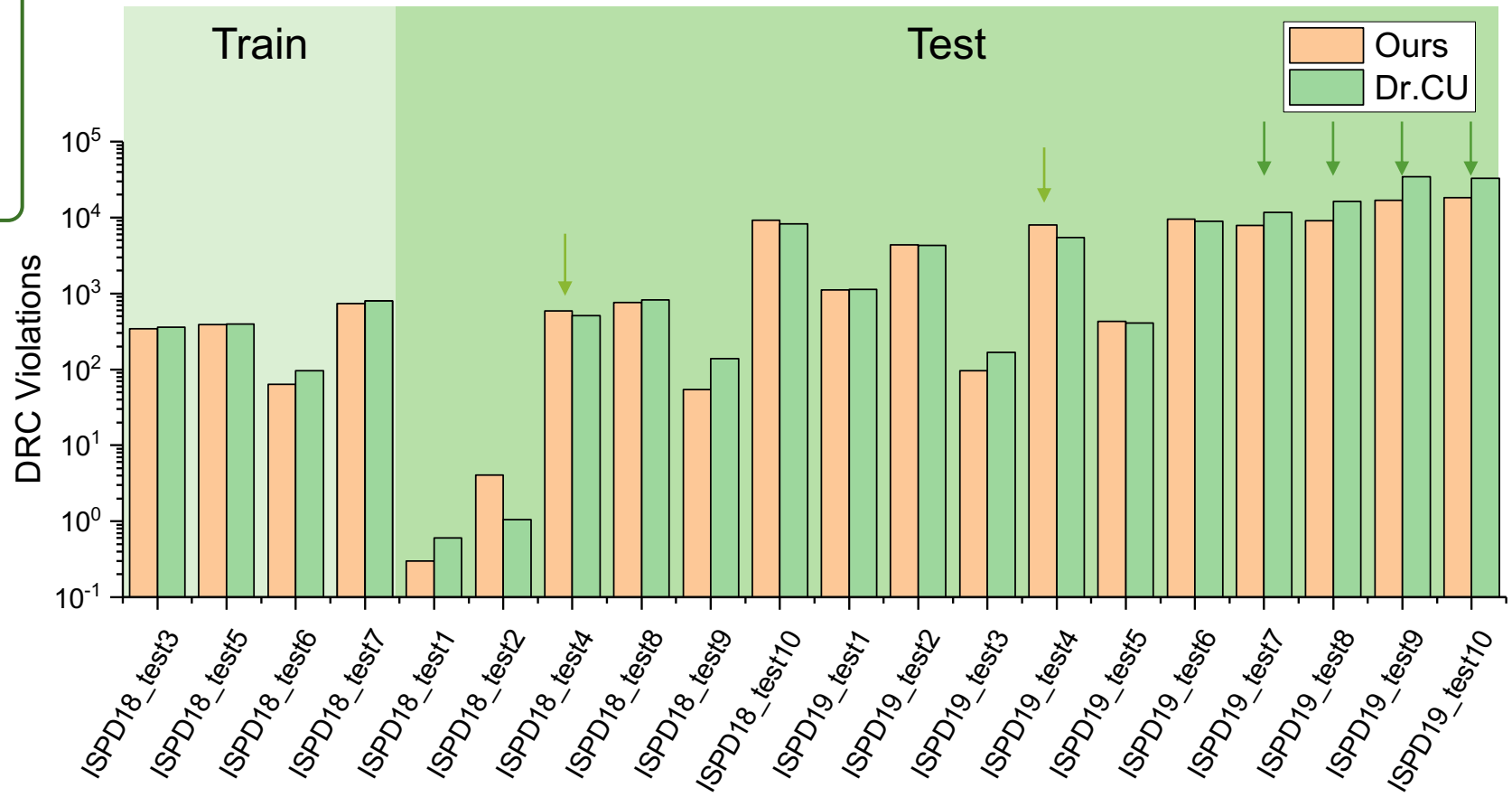
Mismatch Penalty to Dr.CU Strategy

- Default net ordering strategy
  - Routing region sizes
- Modified reward
  - $R = -C_{agent} + C_{dr.cu} - \frac{\alpha}{k}\sum_{i=1}^{k}\Delta a_i^2$, k is the number of nets to be routed



Experimental Results

- ISPD 2018&2019 benchmarks
- 14% fewer DRC violations
- 0.7% fewer total cost
- 6% runtime overhead



Tong Qu, et al. "Asynchronous Reinforcement Learning Framework for Net Order Exploration in Detailed Routing, " DATE 2021 & TCAD 2021

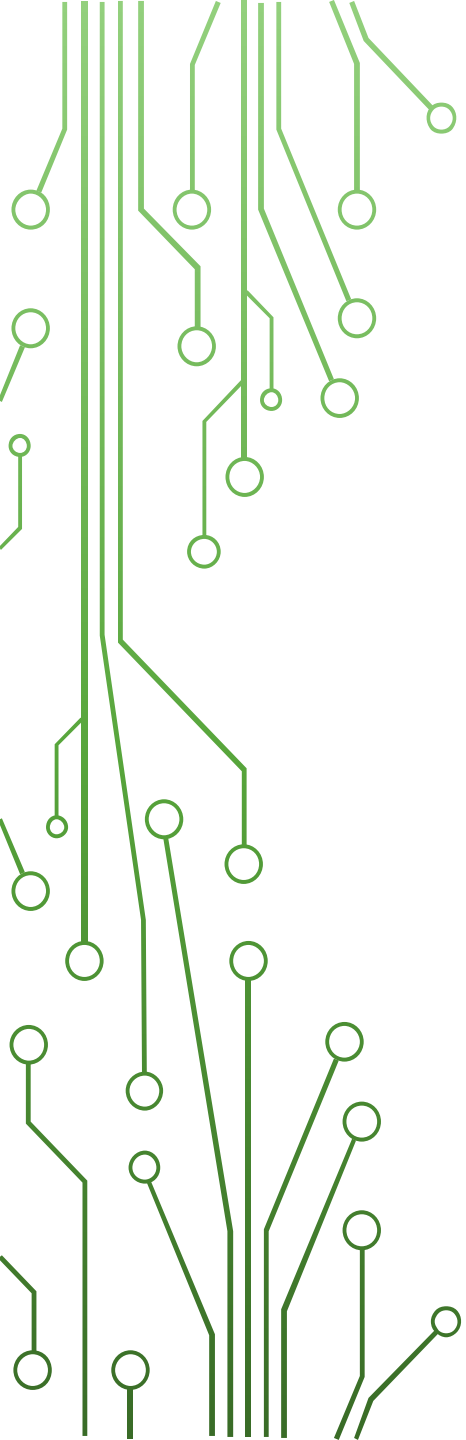
Summary of Routing

- Maze routing
  - Lee's algorithm
  - Aker's coding scheme
- Reinforcement learning to determine routing orders

Summary

- EDA is fundamental to the semiconductor industry
- Require extensive algorithm design and programming skills
  - Dynamic programming
  - Graph theory
  - Convex optimization
  - …
- New opportunities in EDA
  - New algorithms
  - New problems
  - AI for EDA
  - Hardware acceleration for EDA

You are welcome to work in this exciting area!



北京大学高能效计算与应用中心
Center for Energy-efficient Computing and Applications

Thanks!

Questions are welcome

Website: https://yibolin.com
Email: yibolin@pku.edu.cn