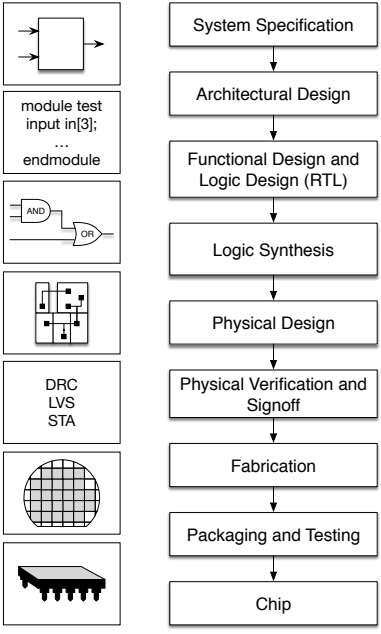
香港中文大學
The Chinese University of Hong Kong

Placement in Advanced Technology Nodes

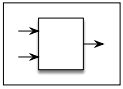
Bei Yu
Department of Computer Science & Engineering
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

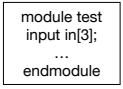
September 30, 2021

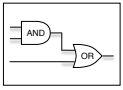
Placement in Design Flow

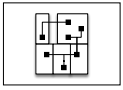


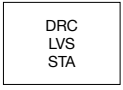
Placement in Design Flow

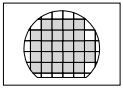


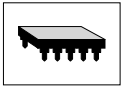


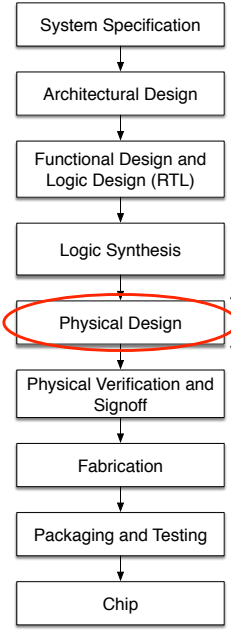


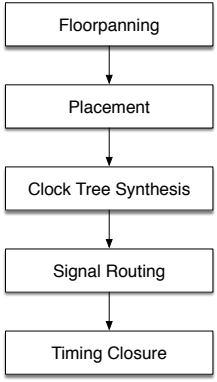




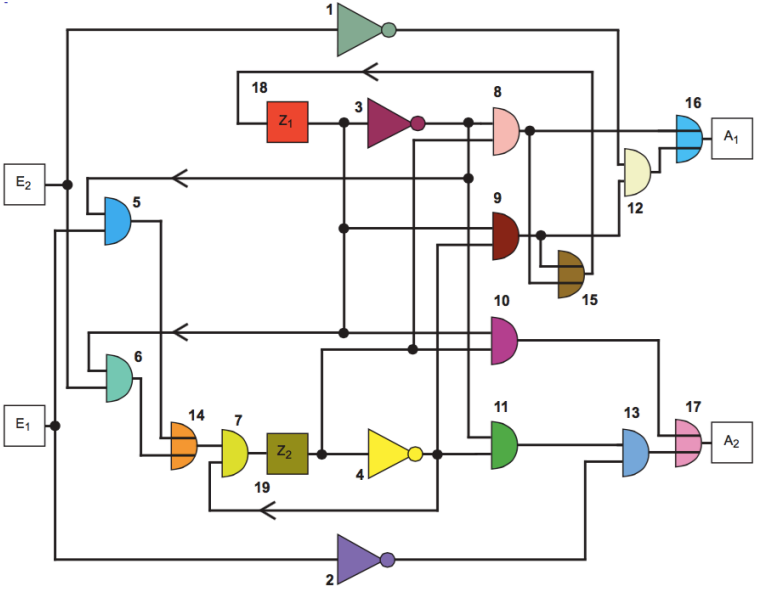




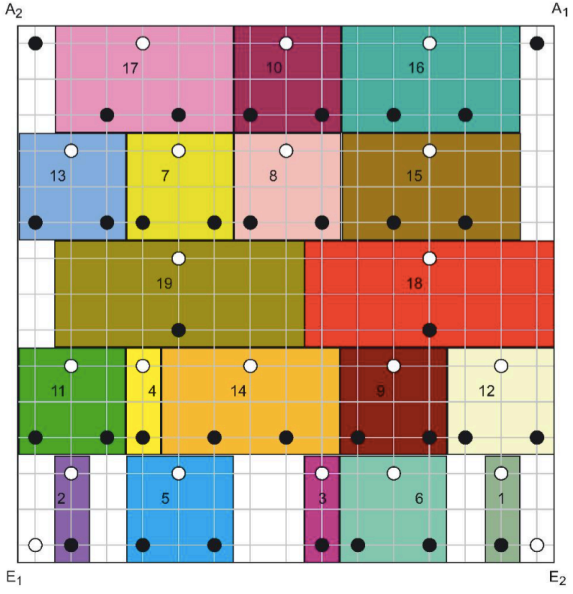




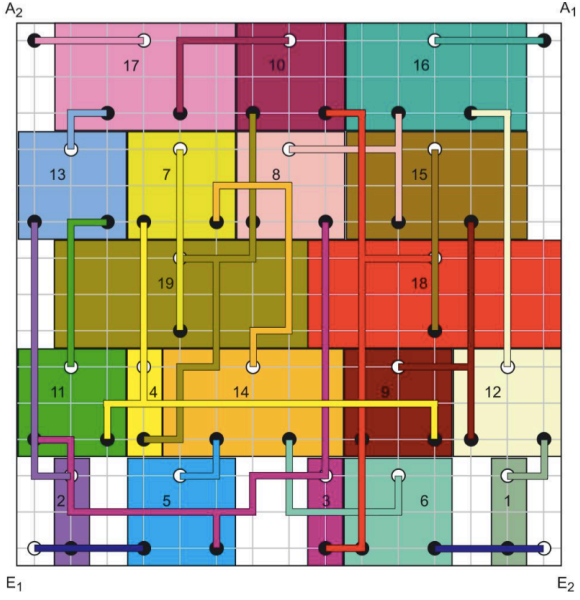
EDA Toy Example [Jens Vygen,2006]



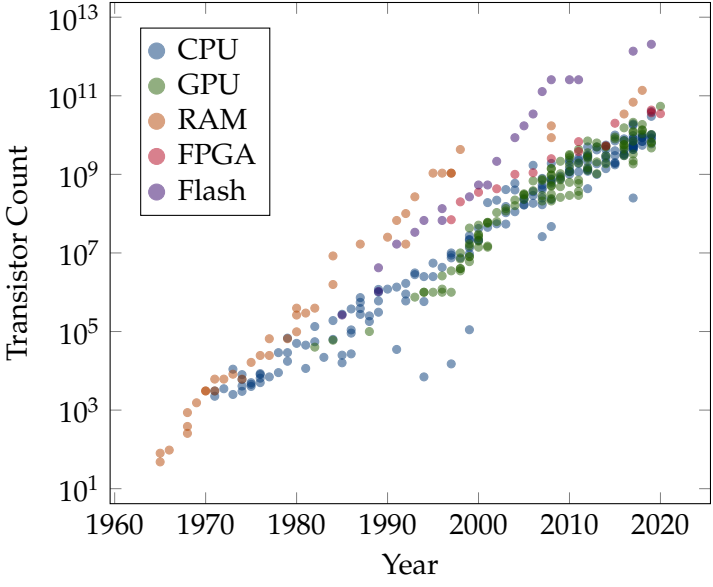
EDA Toy Example [Jens Vygen,2006]



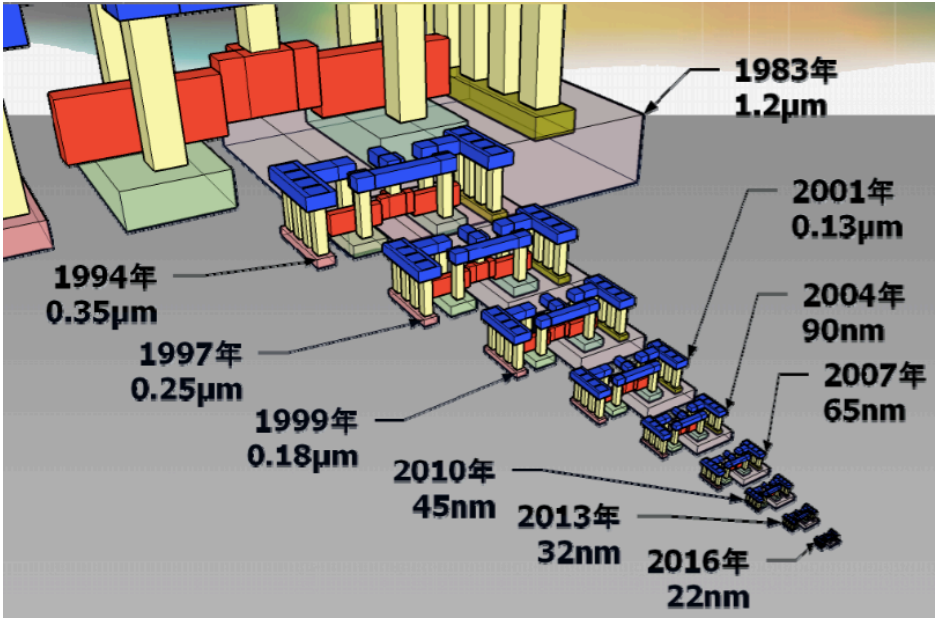
EDA Toy Example [Jens Vygen,2006]



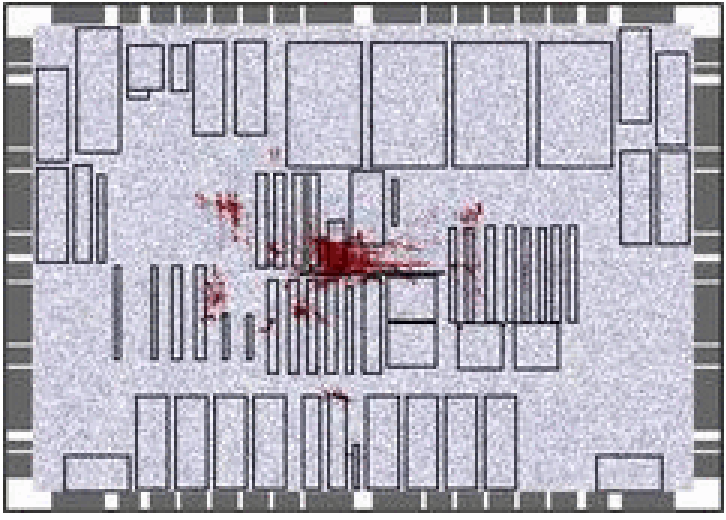
Moore's Law to Extreme Scaling



An Inverter Example

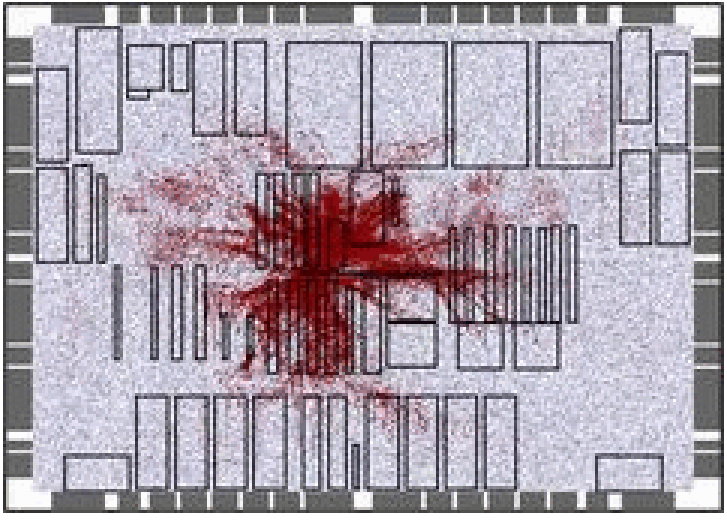


Challenge: Larger and Larger Scale



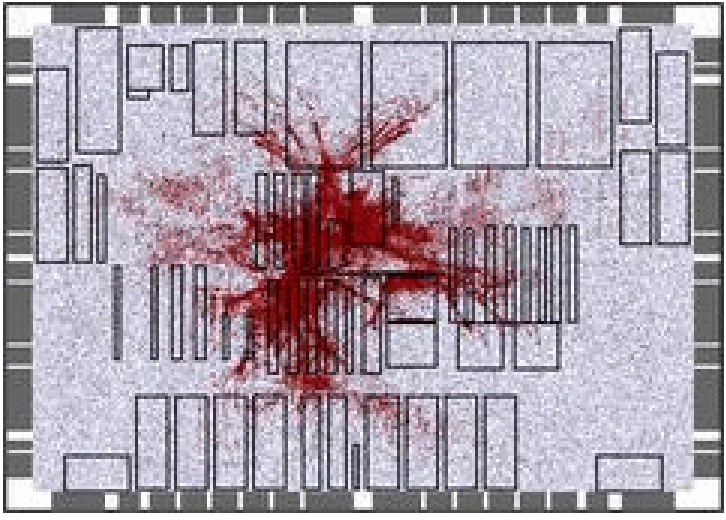
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



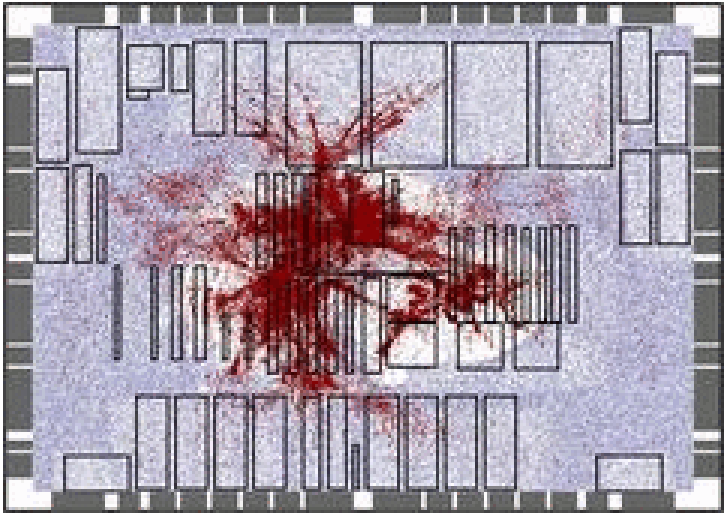
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



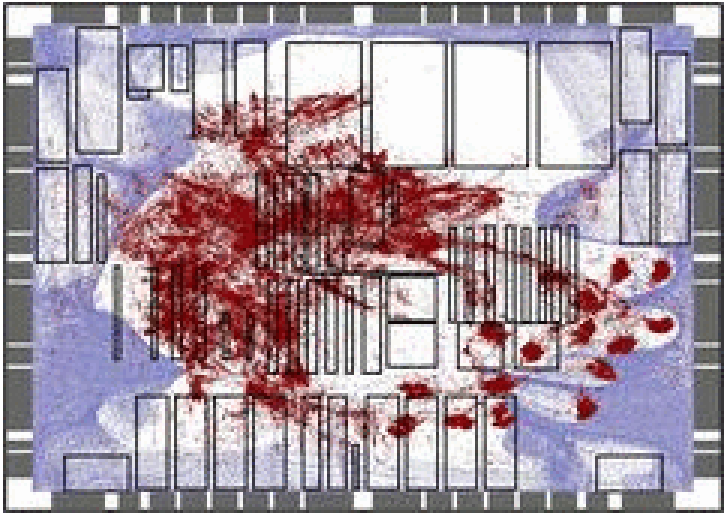
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



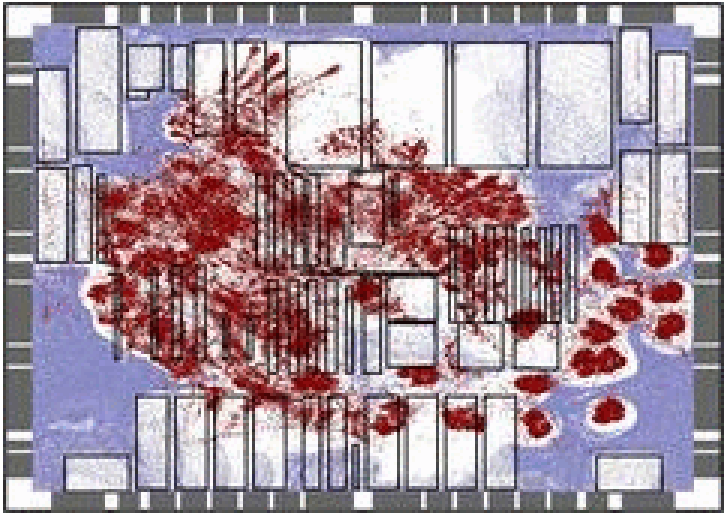
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



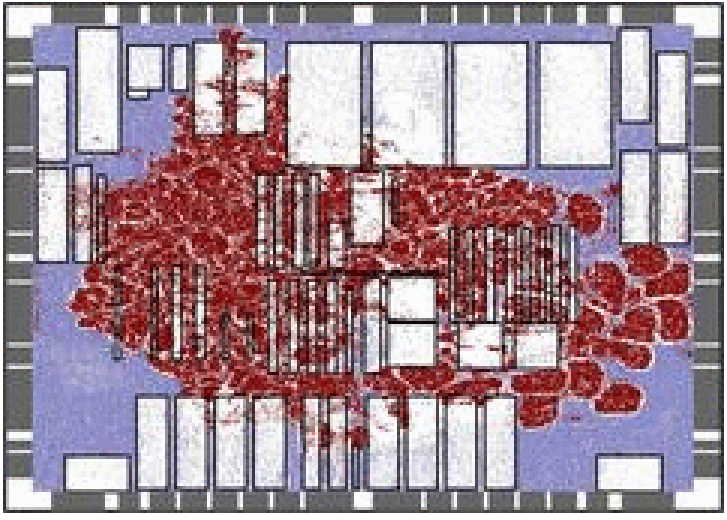
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



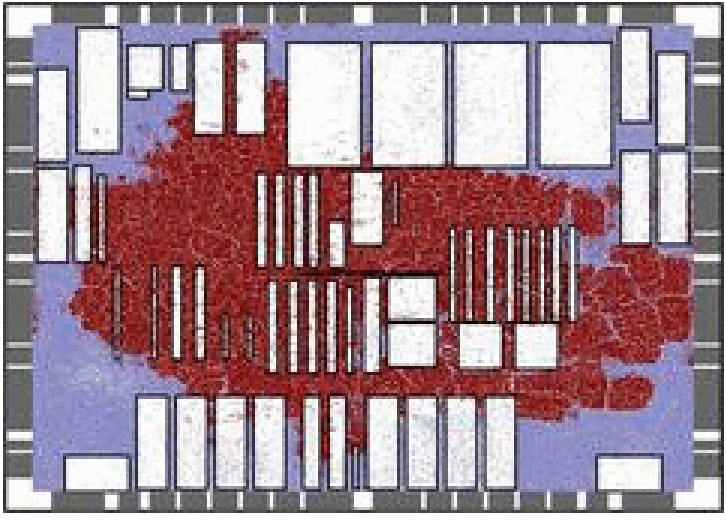
Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

Challenge: Larger and Larger Scale



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

Challenge: Larger and Larger Scale

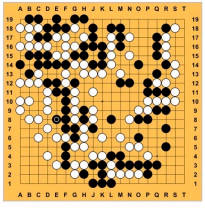


Placement [Lu+,DAC'14]: 221K nets, 63 fixed macros and 210K movable cells.

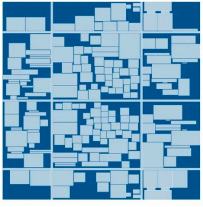
How Difficult Placement is

- 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}

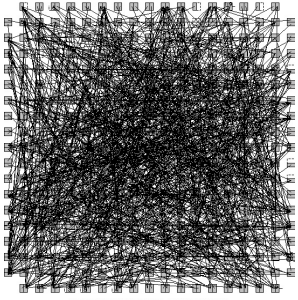


#states: >$10^{100,000}$

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

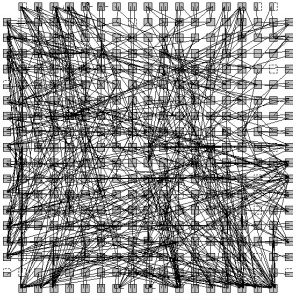
Good Placement vs. Bad Placement

Random Initial



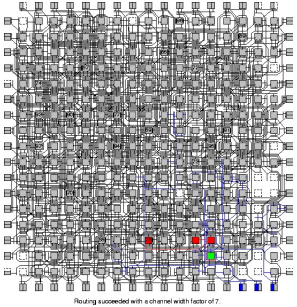
WL = 5.47e+4

Final Solution



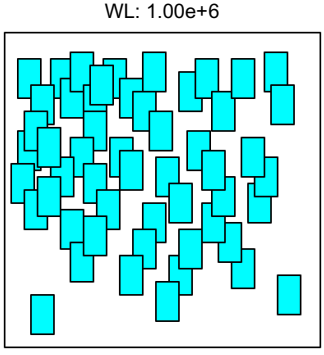
WL = 6.73e+3

Routing Solution



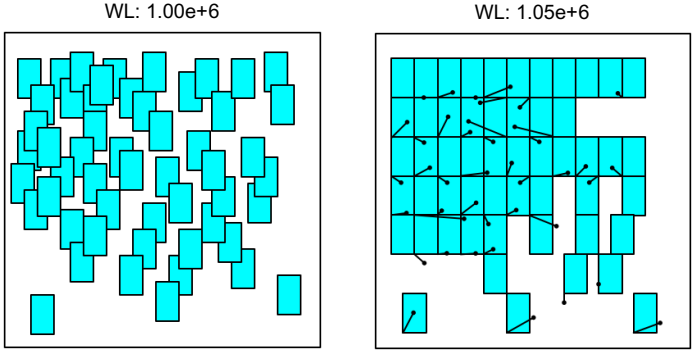
230 cells in FPGA (design `e64` in the MCNC benchmark suite)

Typical Placement Flow



1. Global placement

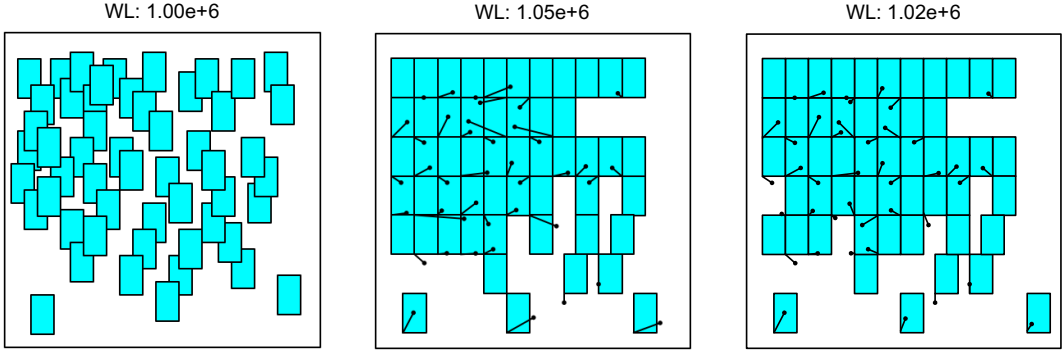
Typical Placement Flow



1. Global placement

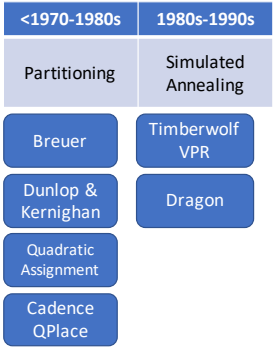
2. Legalization

Typical Placement Flow

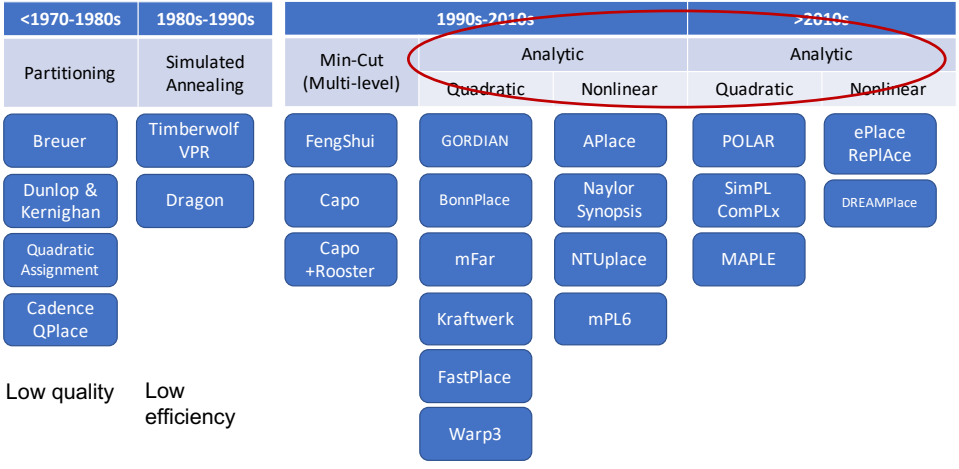


Global Placement Algorithms

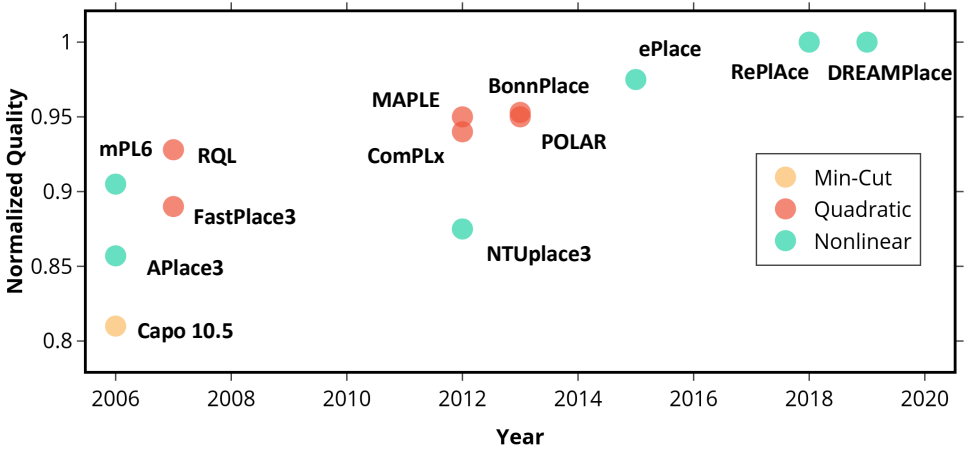
The History of Placement Algorithms



The History of Placement Algorithms

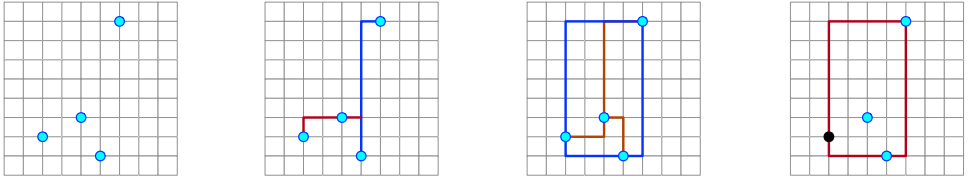


Recent Development of Placement



Quadratic Placement – Optimizing Wirelength Objective

$\sum|x_i - x_j| + \sum|y_i - y_j|$ $max|x_i - x_j| + max|y_i - y_j|$

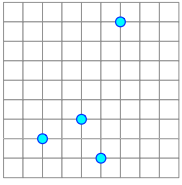


4-pin net | Optimal: min. Steiner tree | Clique model | HPWL

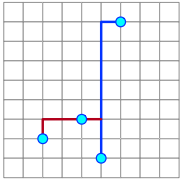
Manhattan-distance model

Quadratic Placement – Optimizing Wirelength Objective

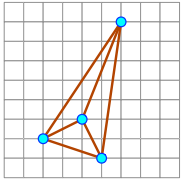
$$\sum (x_i - x_j)^2 + \sum (y_i - y_j)^2$$



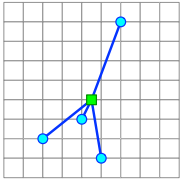
4-pin net



Optimal: min. Steiner tree



Clique model

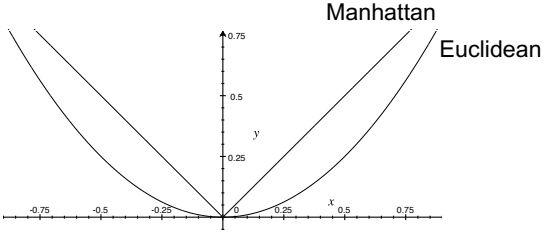


Star model

Euclidean-distance model

Quadratic Placement – Optimizing Wirelength Objective

		Pessimistic		Optimistic	
Model	**Min. Steiner Tree**	**Clique Model**		**HPWL**	**Star Model**
Distance	Manhattan	Manhattan	Euclidean	Manhattan	Euclidean
Accuracy	★★★★★	★	★	★★★★	★★★
Smoothness	★	★★	★★★★★	★★	★★★★★



Quadratic Placement – Optimizing Wirelength Objective

Compute wirelength for **net1**

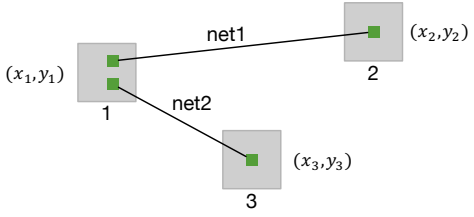
$WL_1 = (x_1 - x_2)^2 + (y_1 - y_2)^2$

Compute wirelength for **net2**

$WL_2 = (x_1 - x_3)^2 + (y_1 - y_3)^2$

Minimize total wirelength

Physical intuition?



How to solve?

Quadratic Programming (QP)

$$\min. \frac{1}{2} x^T A x - b^T x$$

Gradient of x

$$Ax - b = 0$$

Quadratic Placement – Optimizing Wirelength Objective

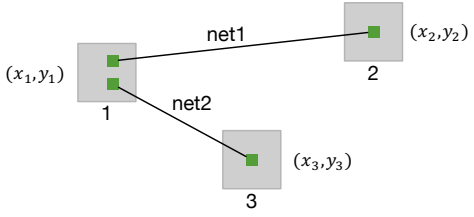
Compute wirelength for **net1**

$WL_1 = (x_1 - x_2)^2 + (y_1 - y_2)^2$

Compute wirelength for **net2**

$WL_2 = (x_1 - x_3)^2 + (y_1 - y_3)^2$

Minimize total wirelength



Physical intuition?

How to solve?

Quadratic Programming (QP)

$$\min. \frac{1}{2} x^T A x - b^T x$$

Gradient of x

$$Ax - b = 0$$

Any problem?

Optimal solution

$x_1 = x_2 = x_3$

$y_1 = y_2 = y_3$

All blocks overlap!

Quadratic Placement – Satisfying Constraints

- Rough Legalization
  - Leverage **anchors**

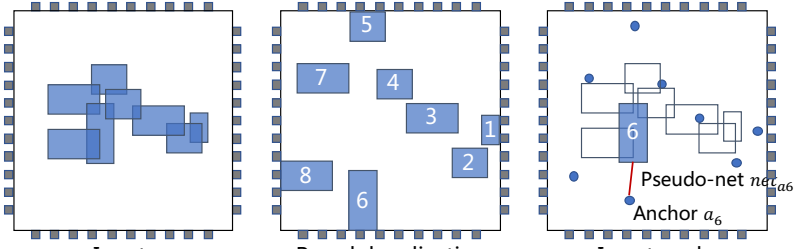
j connected to 6

Original obj. $min. \cdots + \sum (x_6 - x_j)^2 + \cdots$

Anchor loc.

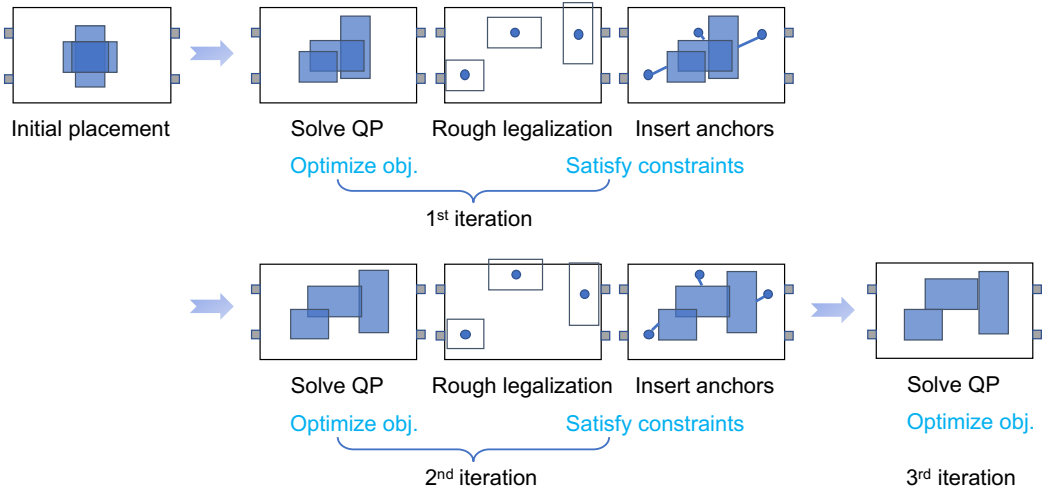
Obj. with anchors $min. \cdots + \sum (x_6 - x_j)^2 + w_6 (x_6 - x_{a_6})^2 + \cdots$

$WL_{net_{a6}}$

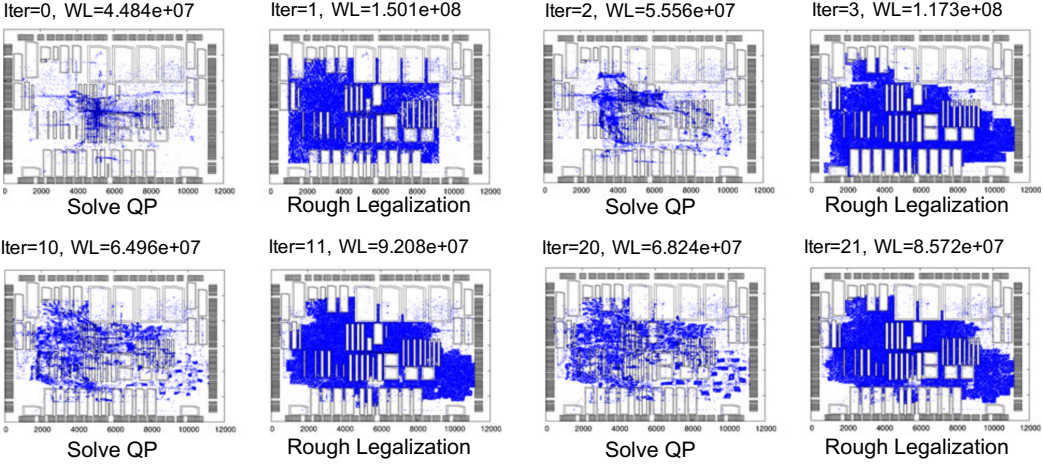


Input | Rough legalization | Insert anchors

Quadratic Placement – Overall Optimization Flow

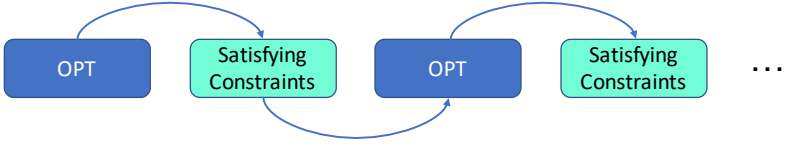


Quadratic Placement – Overall Optimization Flow



Quadratic Placement – Summary

- Iterative optimization
- Wirelength models: HPWL, clique model, star model
- QP solver is fast!
- Rough legalization



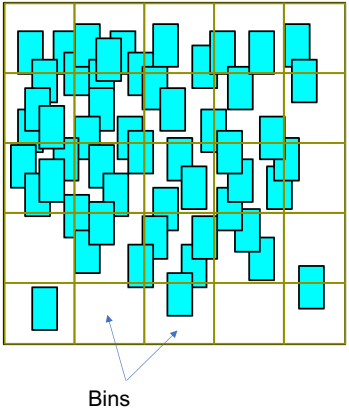
Nonlinear Placement

- Mathematical formulation
  - d_i denotes the density of bin i

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

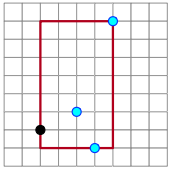
- Nonlinear placement objective
  - Lagrangian relaxation

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



Nonlinear Placement – Wirelength Smoothing[1]

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

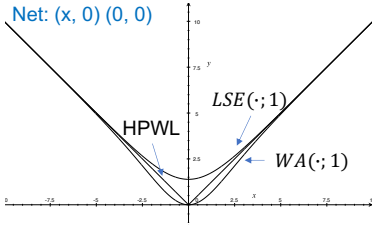


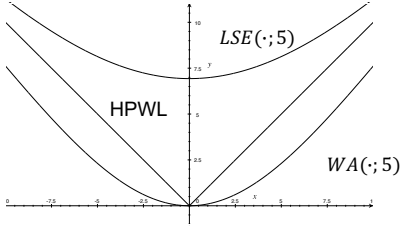
[1] William C. Naylor, Ross Donelly, and Lu Sha (2001). *Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer*. US Patent 216,632.

Nonlinear Placement – Wirelength Smoothing[2]

- 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





[2]Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang (2013). "TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model". In: *IEEE TCAD* 32.4, pp. 497–509.

2.3.1 Numerical Stability and Approximation Error. Although multivariate wirelength models (*LSE*, *WA*) are able to approximate maximum (minimum) as close as possible in theory, numerical instability may happen when γ is too small. On a modern computer, a double-precision floating-point cannot handle e^z when $z > 710$ (overflow issue), or $z < -746$ (precision limit). Thus, an appropriate value of γ needs to be determined carefully for *LSE* and *WA*. On the other hand, there is no exponential function in a bivariate function, so the value of γ can be much smaller, as long as it is representable and appropriate. As a result, the bivariate wirelength model is able to achieve a lower approximation error of *HPWL*, when compared to the multivariate wirelength model [14]. However, a lower approximation error does not always imply better solution quality, since the optimization process of the placement problem is very complicated.

2.3.2 Total Runtime. In an optimization process, the total runtime is roughly equal to the number of iterations times the runtime per iteration. The runtime per iteration can be further divided into the intrinsic cost of f_γ and the cost of calling the function. The bivariate wirelength model has a smaller intrinsic computational cost, which is apparent since the multivariate wirelength model uses more complex functions such as logarithm and exponentiation, whereas the bivariate wirelength model only uses square and square root. However, the calculation process of a bivariate wirelength model requires recursive function calls, which induces considerable computational overheads. The number of iterations to convergence is difficult to analyze in a placement problem since there are overlapping and density constraints. However, it has been observed that the bivariate wirelength model needs more iterations to converge [14]. Combining all these factors, the multivariate wirelength model could be

Algorithm: *BiG* wirelength model
Input:
$X = [x_1, \ldots, x_n]$: variables (array)
f: smooth maximum function
Output:
$G = [g_1, \ldots, g_n]$: derivatives for each variables (array)
max': estimated maximum value (scalar)

$max_1 \leftarrow -\infty$ //first largest value
$max_2 \leftarrow -\infty$ //second largest value
for each x_i:
  if $x_i > max_1$:
    $max_2 \leftarrow max_1$
    $max_1 \leftarrow x_i$
  else if $x_i > max_2$:
    $max_2 \leftarrow x_i$
$sum_g \leftarrow 0$
for each x_i:
  if $x_i == max_1$:
    $g_i \leftarrow \frac{\partial}{\partial x_i} f(x_i, max_2)$
  else:
    $g_i \leftarrow \frac{\partial}{\partial x_i} f(x_i, max_1)$
  $sum_g \leftarrow sum_g + g_i$
for each g_i:
  $g_i \leftarrow \frac{g_i}{sum_g}$
$max' \leftarrow max_1$
return G, max'

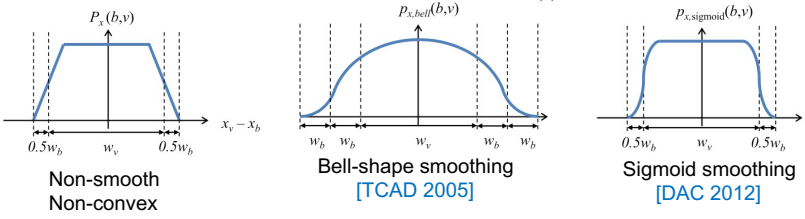
Figure 1: Algorithm of *BiG*.

[3]

[3]Fan-Keng Sun and Yao-Wen Chang (2019). "BiG: A bivariate gradient-based wirelength model for analytical circuit placement". In: *Proc. DAC*.

Nonlinear Placement – Density Penalty[4],[5]

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



[4] Andrew B Kahng and Qinke Wang (2005). "Implementation and extensibility of an analytic placer". In: *IEEE TCAD* 24.5, pp. 734–747.

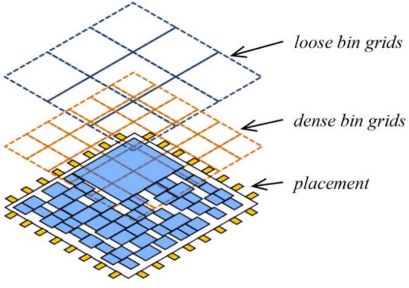
[5] Sheng Chou, Meng-Kai Hsu, and Yao-Wen Chang (2012). "Structure-aware placement for datapath-intensive circuit designs". In: *Proc. DAC*, pp. 762–767.

Nonlinear Placement – Density Penalty

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

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

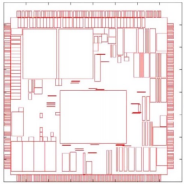
- Challenges
  - Gradient only has local view
  - Need multi-level bins



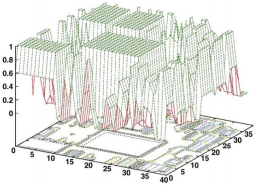
Multi-level bins

Nonlinear Placement – Density Penalty

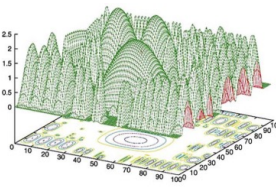
- Potential function for fixed macros
  - Bell-shape smoothing works well for standard cells
  - For fixed macros, $P'(x, y) = G(x, y) * P(x, y)$



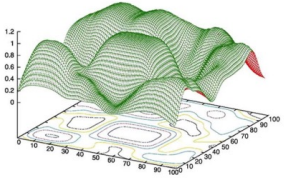
ISPD2005
adaptec2



Exact potential
$P(x, y)$



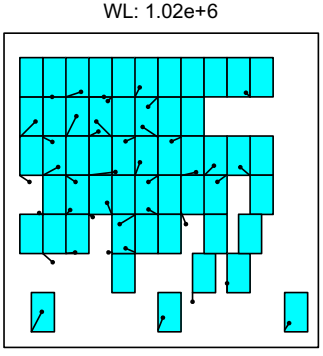
Bell-shape smoothing



Gaussian smoothing
$P'(x, y)$

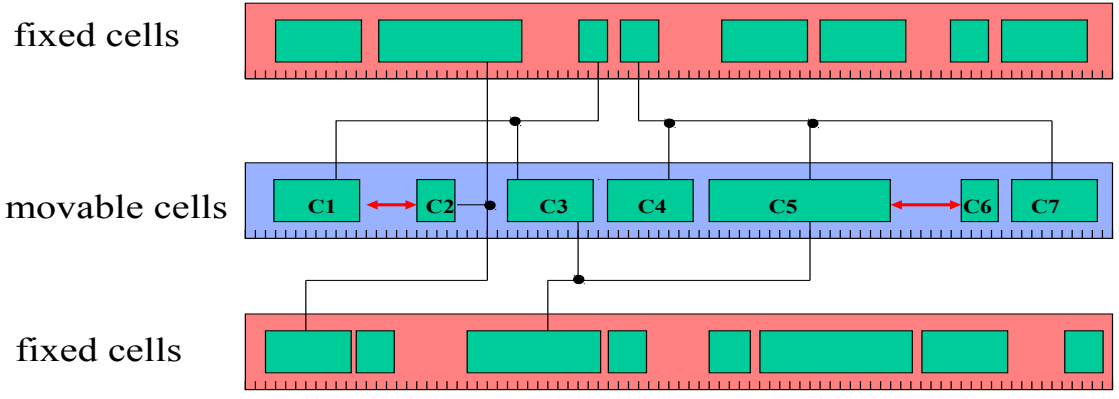
Detailed Placement Algorithms

NP-Hardness



- Multi-bin Knapsack?
- TSP?

Single-Row Problem



Single-Row Placement[6]

$$
\begin{aligned}
\min \quad & \sum_{n \in N} w(n)\,(\overline{x}(n) - \underline{x}(n)) \\
\text{s.t.} \quad & \underline{x}(n) \le x(c(p)) + xoffs(p) \le \overline{x}(n) && \text{(for all } p \in n\text{)} \\
& x^i_{\min} \le x(c^i_1) && \text{(for all } i\text{)} \\
& x(c^i_j) + \frac{w(\{c^i_j, c^i_{j+1}\})}{2} \le x(c^i_{j+1}) && \text{(for all } i, j\text{)} \\
& x(c^i_{m_i}) \le x^i_{\max} && \text{(for all } i\text{)}
\end{aligned}
$$

This LP is the dual of a minimum–cost flow problem. To see this, we introduce an auxiliary variable v_0 with the value zero. Then every constraint has the form $v_i - v_j \ge a_{ij}$, where a_{ij} is a constant and v_i and v_j are variables. Each of these constraints corresponds to a directed edge in a graph and to a dual variable f_{ij}. As the dual LP we then obtain

$$
\begin{aligned}
\max \quad & \sum_{i,j} a_{ij} f_{ij} \\
\text{s.t.} \quad & \sum_i f_{ij} - \sum_k f_{jk} = \begin{cases} -w(n) & \text{if } v_j \text{ is } \underline{x}(n) \\ w(n) & \text{if } v_j \text{ is } \overline{x}(n) \\ 0 & \text{otherwise} \end{cases} && \text{(for all } j\text{)} \\
& f_{ij} \ge 0 && \text{(for all } i, j\text{)}
\end{aligned}
$$

- Dual of Min-Cost Flow
- Unimodular matrix

[6] Jens Vygen (1998). "Algorithms for detailed placement of standard cells". In: *Proc. DATE*, pp. 321–324.

Single-Row Placement[7],[8]

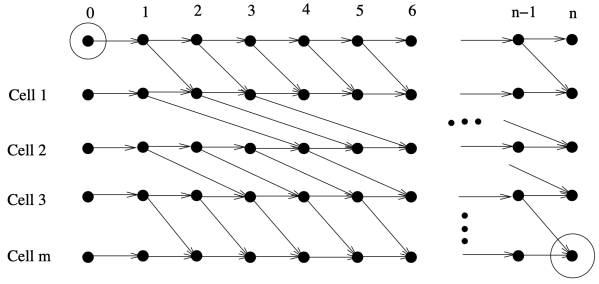


Figure 4: Shortest path computation for legalizing a row placement.

[7] Andrew B Kahng, Paul Tucker, and Alex Zelikovsky (1999). "Optimization of linear placements for wirelength minimization with free sites". In: *Proc. ASPDAC*, pp. 241–244.

[8] Andrew B. Kahng, Igor L. Markov, and Sherief Reda (2004). "On legalization of row-based placements". In: *Proc. GLSVLSI*, pp. 214–219.

Single-Row Placement[9]

Single-Segment Clustering Algorithm

```
num_old_cluster      n
Initialize old_cluster[i]  as standard cell Ci , i =1, 2, ..., num_old_cluster .
do
  Find the bounds list and the Optimal Region Center Xic  for Ki ,
        and set X(old_cluster[i])  = Xic
  newcount      1  // the count for the number of new clusters
  new_cluster[1]      old_cluster[1]  // initialize the first new cluster
  j      1
  while(i < num_old_cluster )
     do
        if new_cluster[newcount]  and old_cluster[j+1]  has overlap
           Cluster new_cluster[newcount]  and old_cluster[j+1]  to form the
                 new new_cluster[newcount]
           Merge the bounds list for new_cluster[newcount]  and old_cluster[j+1]
                 to get the new bounds list for new_cluster[newcount]
           Find the Optimal Region Center Xc  for new_cluster[newcount]
                 based on the new bounds list
           X(new_cluster[newcount])      Xc
        else
           newcount      newcount  + 1   //begin a new cluster new_cluster[newcount+1]
           j     j+1

  num_old_cluster      newcount
  old_cluster[i]      new_cluster[i]  (i =1, ..., newcount )
until no overlap among old_cluster[i] , ( i =1, ..., num_old_cluster  )
Assign the Ci  ( i =1, 2, ..., n ) to the positions according to the positions of the
old_cluster[j]  ( j =1, 2, ..., num_old_cluster ) they belong to
```

[9] Min Pan, Natarajan Viswanathan, and Chris Chu (2005). "An efficient and effective detailed placement algorithm". In: *Proc. ICCAD*, pp. 48–55.

Can we consider Cell Ordering?

- Sung Woo Hur and John Lillis (2000). "Mongrel: hybrid techniques for standard cell placement". In: *Proc. ICCAD*, pp. 165–170
- Yuelin Du and Martin D. F. Wong (2014). "Optimization of standard cell based detailed placement for 16 nm FinFET process". In: *Proc. DATE*, 357:1–357:6

DREAMPlace

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

Objective of nonlinear placement

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

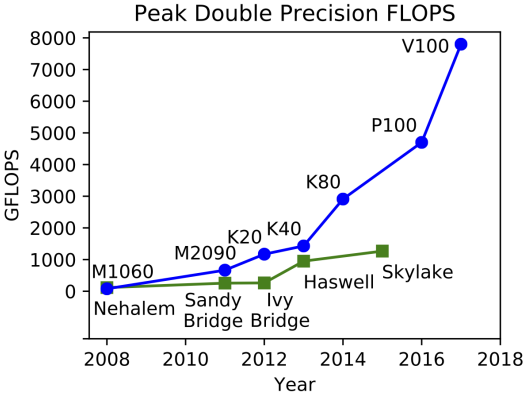
Challenges of Nonlinear Placement

Low efficiency
- >3h for 10M-cell design

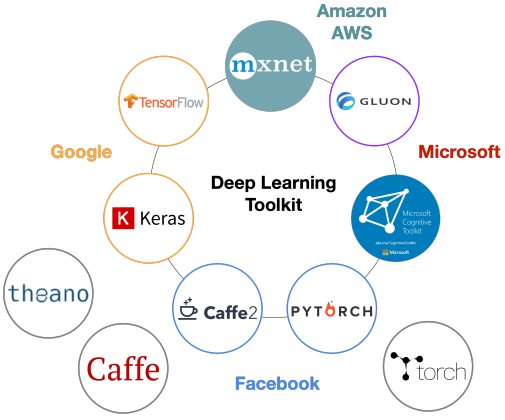
Limited acceleration
- Limited speedup, e.g. mPL, due to clustering

Huge development effort
- >1year for ePlace/RePlAce

Advances in Deep Learning Hardware/Software



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



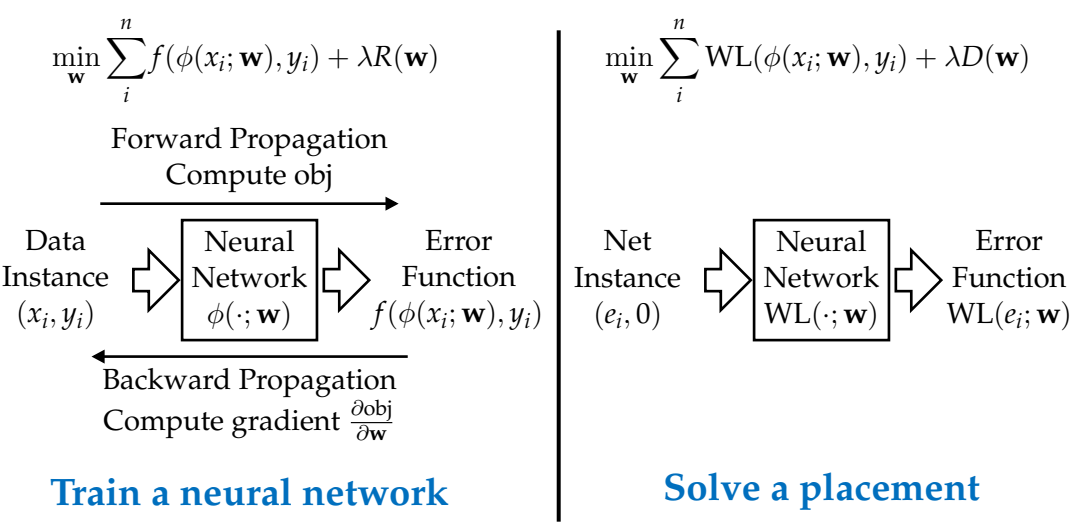
Deep learning toolkits

DREAMPlace Strategies

- 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.

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

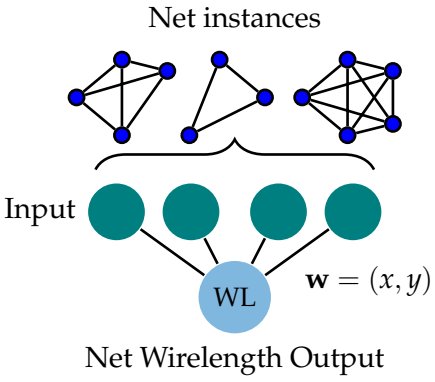
Analogy between NN Training and Placement [DAC'19][11]



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

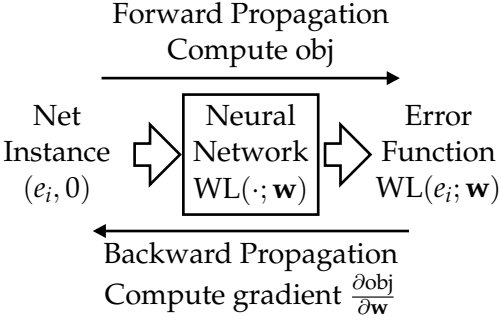
Analogy between NN Training and Placement [DAC'19][11]

Casting the placement problem into neural network training



Train a neural network

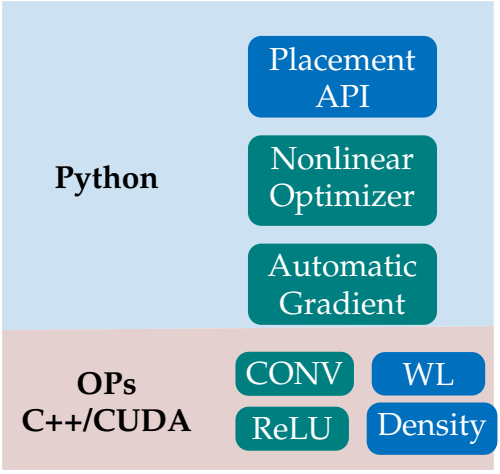
$$\min_{\mathbf{w}} \sum_{i}^{n} \mathrm{WL}(e_i; \mathbf{w}) + \lambda D(\mathbf{w})$$



Solve a placement

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

Leverage highly optimized deep learning toolkit PyTorch



Match RePlAce

Nesterov's Method

[TCAD'19,Cheng+]

12

[12]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

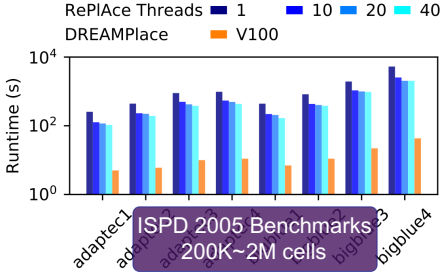
DREAMPlace
- CPU: Intel E5-2698 v4 @2.20GHz
- GPU: 1 NVIDIA Tesla V100
- Single CPU thread was used

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

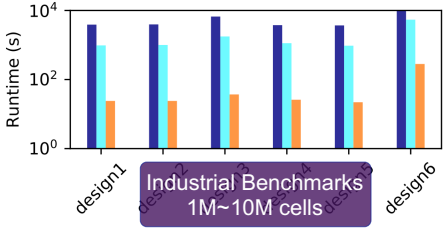
Same quality of results!

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

34x speedup

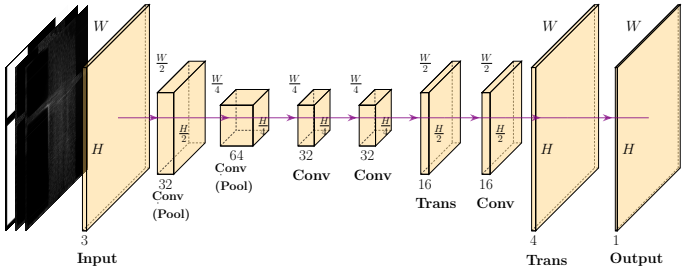


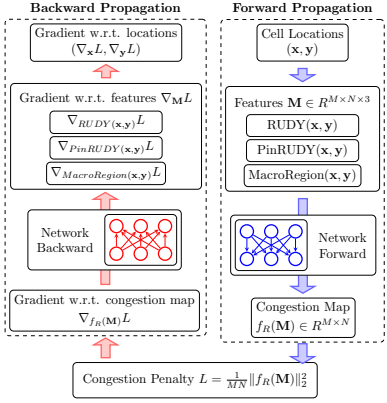
43x speedup



Future Directions

Ex: Routability Estimation × DREAMPlace [DATE'21][13]

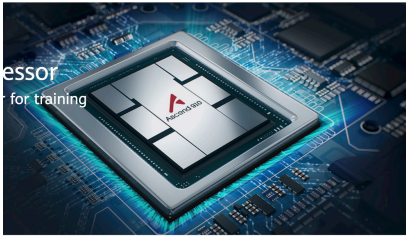


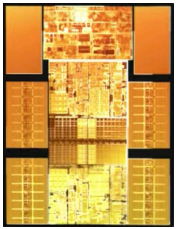


[13] Siting Liu et al. (2021). "Global Placement with Deep Learning-Enabled Explicit Routability Optimization". In: *Proc. DATE*.

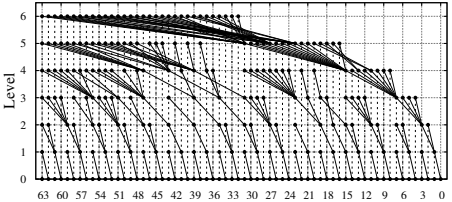
Datapath

Huawei Ascend 910

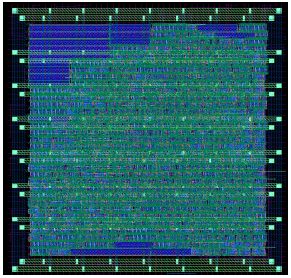




Adder is one of the most important component!

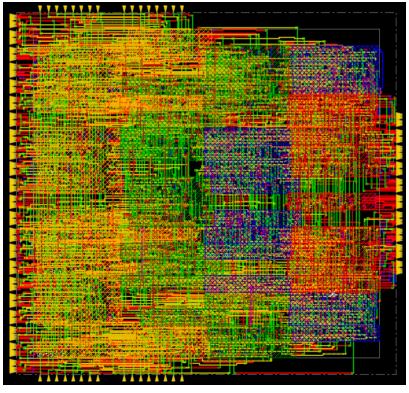


(a) Logic perspective

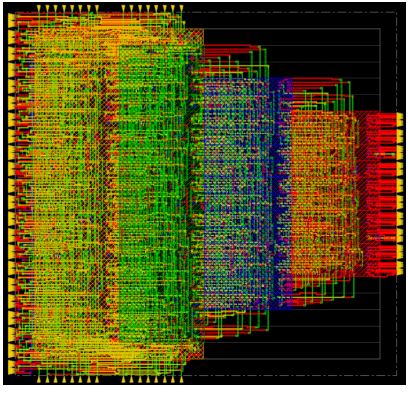


(b) Physical synthesis perspective

EDA Challenges: How to Design an AI Chip Component?



(c) Current EDA tool output



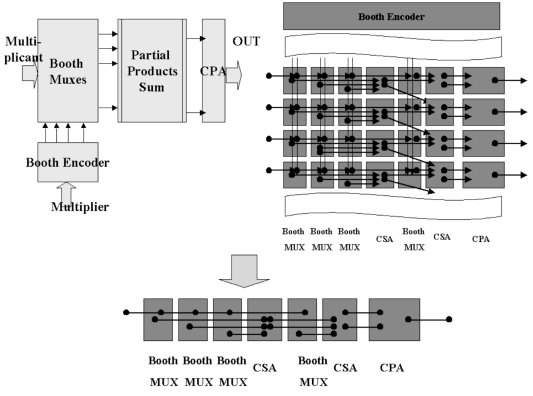
(d) Manual design

Datapath Driven Standard Cell Placement

- Classical idea: bit-sliced DSP datapaths [Cai, DAC'90]
  - Decide ordering of linearly placed blocks
  - Solved by A$*$ in the search space
- Standard cell placement [Tsay, TCAD'95]
  - Strongly connected subcircuits (cones) are extracted
  - BFS + heuristics
  - Placed as macro cells

Datapath Driven Placement: Abstract Physical Model

- Placer has little control of exact locations if datapath is generated separately
- Abstract physical model [Ye, ICCAD'00]
  - Bit-sliced abstraction
  - Compiled from HDL
  - Blocks are placed abutted
- Two-step heuristic for linear placement
  - quadratic assignment
  - sliding window optimization



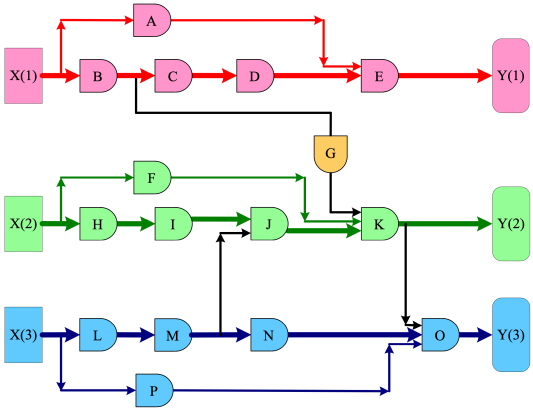
APM of a booth multiplier [Ye, ICCAD'00]

Regularity Extraction

- Consider cells with the same bit-slice are lined up horizontally [Nijssen IWLS'96]
  - geometric regularity: circuit is fitted onto a matrix of rectangular buckets
  - interconnect regularity: most nets are within one slice/one stage
- Typical methods for regularity extraction
  - Search-wave expansion [Nijssen IWLS'96]
  - Signature-based [Arikati, ICCAD'97]
  - Template covering [Chowdhary, TCAD'99]
  - Network flow [Xiang, ISPD'13]
  - Bipartite graph vertex covering [Huang, DAC'17]

Regularity Extraction: Network Flow Approach

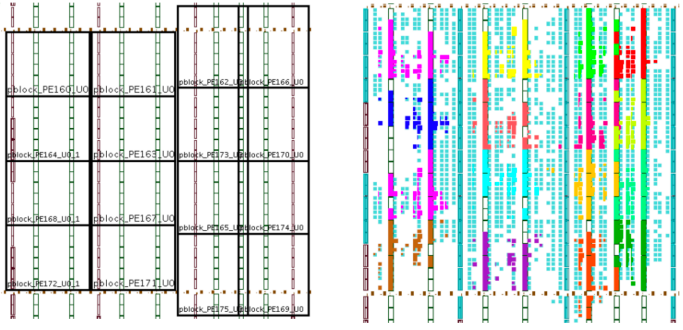
- *Datapath main frame* (DMF) [Xiang, ISPD'13]
  - a set of n disjoint paths from input to output
  - maximize the number of datapath gates on these paths
- Can be optimally identified by the min-cost max-flow algorithm



DMF identification can be transformed to a network flow problem [Xiang, ISPD'13]

Datapath Driven Systolic Array Placement

- Systolic arrays are a popular choice to support neural network computations
- Current FPGA CAD tools cannot synthesize them in high quality
- One solution: restrict fixed locations for PEs [Zhang, ISCAS'19]
  - Sufficient DSPs, close to used I/O banks



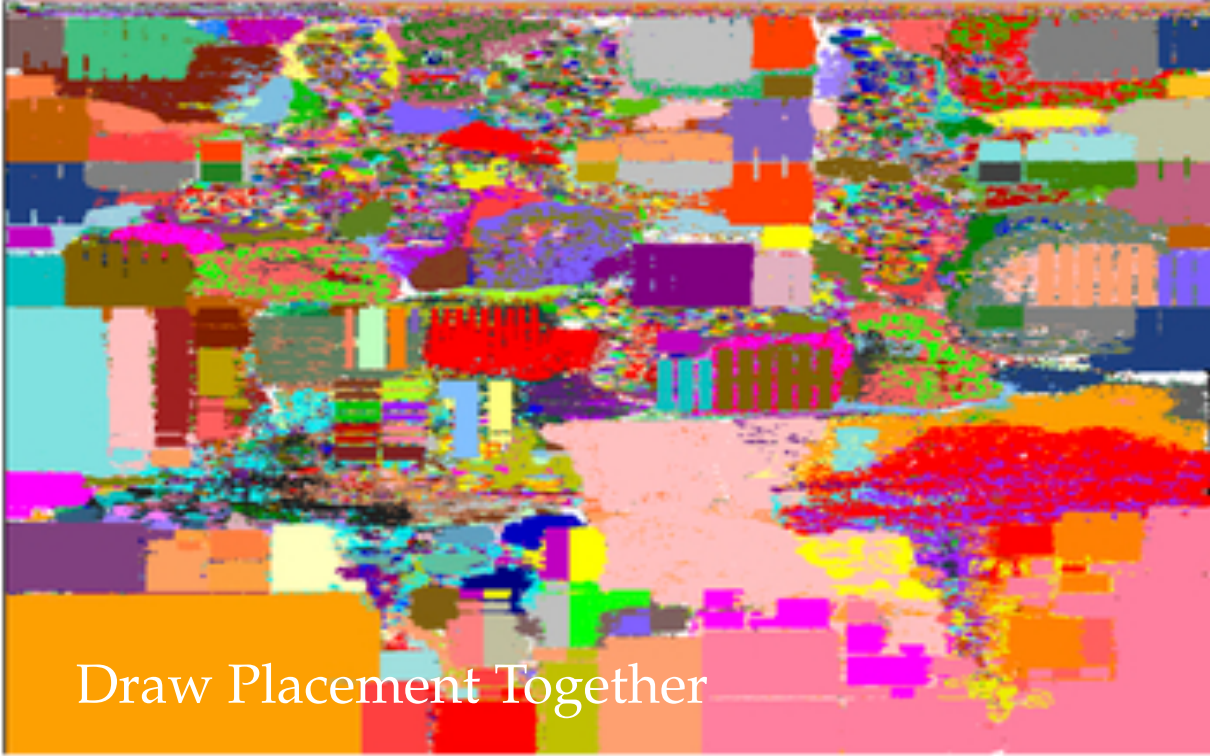
PE placement with floorplan constraints [Zhang, ISCAS'19]

Datapath Driven Placement: Many More

- Detailed placement [Serdar, DATE'01]
- SOC placement [Tong, JOS'02] [Jing, ICCCAS'02]
- Parallel multiplier design [Bae, ISPD'15]
- Genaral ASIC design [Ye, ISCAS'02] [Chou, DAC'12] [Wang, IETCDS'17]
- ...

These slides contain/adapt materials developed by

- Course “Optimization and Machine Learning in VLSI Design Automation”, Peking Univ, 2021.
- Yibo Lin et al. (2019). “DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement”. In: *Proc. DAC*
- Zhuolun He et al. (2021). “Physical synthesis for advanced neural network processors”. In: *Proc. ASPDAC*, pp. 833–840


Draw Placement Together