Placement in Advanced Technology Nodes

Bei Yu

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Placement in Design Flow

module test
input in[3];
...
endmodule

AND
OR

DRC
LVS
STA

System Specification
Architectural Design
Functional Design and Logic Design (RTL)
Logic Synthesis
Physical Design
Physical Verification and Signoff
Fabrication
Packaging and Testing
Chip
Placement in 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
- Floorpanning
- Placement
- Clock Tree Synthesis
- Signal Routing
- Timing Closure

```
module test_input in[3];
...
endmodule
```

AND
OR

DRC
LVS
STA

Floorpanning
Placement
Clock Tree Synthesis
Signal Routing
Timing Closure
EDA Toy Example [Jens Vygen, 2006]
EDA Toy Example [Jens Vygen, 2006]
EDA Toy Example [Jens Vygen, 2006]
Moore’s Law to Extreme Scaling

[Graph showing transistor count over years for different components: CPU, GPU, RAM, FPGA, Flash. The x-axis represents the year from 1960 to 2020, and the y-axis represents the transistor count on a logarithmic scale from $10^1$ to $10^{13}$. Each component shows a doubling trend per year.]
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: $\sim 10^{123}$

#states: $\sim 10^{360}$

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

Google AlphaGo
Train **40 days** using **176 GPUs**
Good Placement vs. Bad Placement

Random Initial

Final Solution

Routing Solution

WL = 5.47e+4

WL = 6.73e+3

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

Background: Placement Algorithms

New Direction: Integrating GPU Power into Placement

New Direction: Datapath aware Placement
Outline

Background: Placement Algorithms

New Direction: Integrating GPU Power into Placement

New Direction: Datapath aware Placement
Typical Placement Flow

1. Global placement
WL: 1.00e+6

2. Legalization
WL: 1.05e+6

3. Detailed Placement
WL: 1.02e+6
Typical Placement Flow

1. Global placement
   WL: $1.00e+6$

2. Legalization
   WL: $1.05e+6$
Typical Placement Flow

1. Global placement
   WL: 1.00e+6

2. Legalization
   WL: 1.05e+6

3. Detailed Placement
   WL: 1.02e+6
The History of Placement Algorithms

<table>
<thead>
<tr>
<th>&lt;1970-1980s</th>
<th>1980s-1990s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td>Breuer</td>
<td>Timberwolf VPR</td>
</tr>
<tr>
<td>Dunlop &amp; Kernighan</td>
<td>Dragon</td>
</tr>
<tr>
<td>Quadratic Assignment</td>
<td></td>
</tr>
<tr>
<td>Cadence QPlace</td>
<td></td>
</tr>
</tbody>
</table>

Low quality  Low efficiency
# The History of Placement Algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partitioning</strong></td>
<td>Breuer</td>
<td>Timberwolf</td>
<td>FengShui</td>
<td>POLAR</td>
</tr>
<tr>
<td></td>
<td>Dunlop &amp; Kernighan</td>
<td>VPR</td>
<td>GORDIAN</td>
<td>ePlace</td>
</tr>
<tr>
<td></td>
<td>Quadratic Assignment</td>
<td>Dragon</td>
<td>Capo</td>
<td>RePLAce</td>
</tr>
<tr>
<td></td>
<td>Cadence QPlace</td>
<td></td>
<td>Capo +Rooster</td>
<td></td>
</tr>
</tbody>
</table>

- Low quality
- Low efficiency

<table>
<thead>
<tr>
<th><strong>Min-Cut (Multi-level)</strong></th>
<th>Analytic</th>
<th>Analytic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonlinear</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- <1970-1980s
- 1980s-1990s
- 1990s-2010s
- >2010s
Quadratic Placement – Optimizing Wirelength Objective

Optimal: min. Steiner tree

Clique model

4-pin net

Manhattan-distance model

\[ \sum |x_i - x_j| + \sum |y_i - y_j| \]

\[ \max |x_i - x_j| + \max |y_i - y_j| \]

HPWL

4-pin net

Optimal: min. Steiner tree

Clique model

HPWL

Manhattan-distance model
Quadratic Placement – Optimizing Wirelength Objective

Optimal: min. Steiner tree

Clique model

Star model

4-pin net

Optimal: min. Steiner tree

Clique model

Star model

Euclidean-distance model

\[ \sum (x_i - x_j)^2 + \sum (y_i - y_j)^2 \]
Quadratic Placement – Optimizing Wirelength Objective

<table>
<thead>
<tr>
<th>Model</th>
<th>Min. Steiner Tree</th>
<th>Clique Model</th>
<th>HPWL</th>
<th>Star Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>Manhattan</td>
<td>Manhattan</td>
<td>Euclidean</td>
<td>Manhattan</td>
</tr>
<tr>
<td>Accuracy</td>
<td>★★★★★</td>
<td>★</td>
<td>★</td>
<td>★★★★</td>
</tr>
<tr>
<td>Smoothness</td>
<td>★</td>
<td>★</td>
<td>★★★★★</td>
<td>★★★★★★</td>
</tr>
</tbody>
</table>

-1.25 -1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1 1.25

Manhattan
Euclidean

Pessimistic
Optimistic
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 \( x \)
\[ A x - b = 0 \]

Any problem?

Optimal solution
\[ x_1 = x_2 = x_3 \]
\[ y_1 = y_2 = y_3 \]

All blocks overlap!
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

\[ \text{Optimal solution} \]
\[ x_1 = x_2 = x_3 \]
\[ y_1 = y_2 = y_3 \]

All blocks overlap!

Physical intuition?

How to solve?

Quadratic Programming (QP)
\[ \min \frac{1}{2} x^T Ax - b^T x \]

Gradient of \( x \)
\[ Ax - b = 0 \]
Quadratic Placement – Satisfying Constraints

• Rough Legalization
• Leverage anchors

\[
\begin{align*}
\text{Original obj.} & \quad \min \cdots + \sum (x_j - x_6)^2 + \cdots \\
\text{Obj. with anchors} & \quad \min \cdots + \sum (x_j - x_6)^2 + w_6(x_j - x_{a_6})^2 + \cdots \\
\end{align*}
\]

Input  
Rough legalization  
Insert anchors

Anchor \(a_6\) connected to 6
Pseudo-net \(n_{a_{6}}\)

14/40
Quadratic Placement – Overall Optimization Flow

1. Initial placement
2. Solve QP
3. Insert anchors
4. Rough legalization
5. Optimize obj.
6. Satisfy constraints
7. 1st iteration
8. Solve QP
9. Rough legalization
10. Insert anchors
11. Optimize obj.
12. Satisfy constraints
13. 2nd iteration
14. Solve QP
15. Insert anchors
17. Satisfy constraints
18. 3rd iteration
Quadratic Placement – Overall Optimization Flow

Iter=0, WL=4.484e+07

Iter=1, WL=1.501e+08

Iter=2, WL=5.556e+07

Iter=3, WL=1.173e+08

Iter=10, WL=6.496e+07

Iter=11, WL=9.208e+07

Iter=20, WL=6.824e+07

Iter=21, WL=8.572e+07

Solve QP

Rough Legalization

Solve QP

Rough Legalization

Solve QP

Rough Legalization
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 $i$

\[
\min_{x,y} \quad WL(x, y), \\
\text{s.t.} \quad d_b(x, y) \leq t_d, \forall b \in \text{Bins}
\]

• Nonlinear placement objective
  • Lagrangian relaxation

\[
\min_{x,y} \quad WL(x, y) + \lambda D(x, y)
\]

Wirelength  Density

Bins
Nonlinear Placement – Wirelength Smoothing

- \( WL(x, y) = \sum_{e \in E} WL_e(x, y) \)
- \( HPWL = \max |x_i - x_j| + \max |y_i - y_j| \)
  - Equivalently \( (\max x_i - \min x_i) + (\max y_i - \min y_i) \)
- Log-sum-exp (LSE)
  - \( LSE(x; \gamma) = \gamma \ln \sum_i e^{\frac{x_i}{\gamma}} \)
  - \( \max\{x_1, \ldots, x_n\} < LSE(x; \gamma) \leq \max\{x_1, \ldots, x_n\} + \gamma \ln(n) \)
  - \( LSE(x; \gamma) \approx \max\{x_1, \ldots, x_n\} \)
  - \( -LSE(x; -\gamma) \approx \min\{x_1, \ldots, x_n\} \)
  - \( WL_e(x, y; \gamma) = \gamma (\ln \sum_{v_i \in e} e^{\frac{x_i}{\gamma}} + \ln \sum_{v_i \in e} e^{\frac{y_i}{\gamma}} + \ln \sum_{v_i \in e} e^{-\frac{y_i}{\gamma}}) \)
Nonlinear Placement – Wirelength Smoothing

- Weighted average (WA)

\[ W L_e(x, y; \gamma) = \left( \frac{\sum_{v_i \in e} x_i e^{x_i / \gamma}}{\sum_{v_i \in e} e^{x_i / \gamma}} - \frac{\sum_{v_i \in e} x_i e^{-x_i / \gamma}}{\sum_{v_i \in e} e^{-x_i / \gamma}} \right) + \left( \frac{\sum_{v_i \in e} y_i e^{y_i / \gamma}}{\sum_{v_i \in e} e^{y_i / \gamma}} - \frac{\sum_{v_i \in e} y_i e^{-y_i / \gamma}}{\sum_{v_i \in e} e^{-y_i / \gamma}} \right) \]

- Larger \( \gamma \rightarrow \) smoother, but less accurate

More recent work
DAC2019
BiG: Bivariant smoothing

Nonlinear Placement – 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 \( v \)
  • \( D_b(x, y) = \sum_{v \in V} P_x(b, v)P_y(b, v) \)

---


Nonlinear Placement – Density Penalty

• Potential function for standard cells
  • Smoothed potential function
    • $\overline{D}_b(x, y) = \sum_{v \in V} \overline{P}_x(b, v) \overline{P}_y(b, v)$
  • $\min_{x,y} WL(x, y) + \lambda D(x, y)$

  $\lambda \sum_b (\overline{D}_b(x, y) - t_d)^2$

• 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, $P'(x, y) = G(x, y) * P(x, y)$
Recent Development of Placement
Outline

Background: Placement Algorithms

New Direction: Integrating GPU Power into Placement

New Direction: Datapath aware Placement
Typical Nonlinear Placement Algorithm

Objective of nonlinear placement

\[
\min_{x,y} \text{WL}(x, y), \quad \text{s.t.} \quad D(x, y) \leq t_d
\]

\[
\min \left( \sum_{e \in E} \text{WL}(e; x, y) \right) + \lambda D(x, y)
\]

Wirelength \quad 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/RePIAce
Advances in Deep Learning Hardware/Software

Over 60x speedup in neural network training since 2013
DREAMPlace Strategies [DAC’19]

- 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.

Analogy between NN Training and Placement [DAC’19]

\[
\min_w \sum_i^n f(\phi(x_i; w), y_i) + \lambda R(w)
\]

Forward Propagation
Compute obj

Data Instance \((x_i, y_i)\)

\[
\begin{align*}
\text{Neural Network} & : \phi(\cdot; w) \\
\text{Error Function} & : f(\phi(x_i; w), y_i)
\end{align*}
\]

Backward Propagation
Compute gradient \(\frac{\partial \text{obj}}{\partial w}\)

Train a neural network

\[
\min_w \sum_i^n WL(\phi(x_i; w), y_i) + \lambda D(w)
\]

Solve a placement

Net Instance \((e_i, 0)\)

\[
\begin{align*}
\text{Neural Network} & : WL(\cdot; w) \\
\text{Error Function} & : WL(e_i; w)
\end{align*}
\]

Casting the placement problem into neural network training

\[
\min_{\mathbf{w}} \sum_{i} \text{WL}(e_i; \mathbf{w}) + \lambda D(\mathbf{w})
\]

**Forward Propagation**
- Compute obj
- Neural Network \( \text{WL}(\cdot; \mathbf{w}) \)
- Error Function \( \text{WL}(e_i; \mathbf{w}) \)

**Backward Propagation**
- Compute gradient \( \frac{\partial \text{obj}}{\partial \mathbf{w}} \)

Train a neural network

Solve a placement

---

Leverage highly optimized deep learning toolkit PyTorch

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

New Solvers
- SGD, ADAM, etc.

New Objectives
- Routability, timing, etc.
- Gate sizing, floorplanning, ...
- SGD, ADAM, etc.
- Multi-GPU, distributed computing, mixed precision, ...

Applicable to Other CAD Problems

New Accelerations
**Siting Liu et al. (2021).** “Global Placement with Deep Learning-Enabled Explicit Routability Optimization”. In: *Proc. DATE*. 

**Ex: Routability Estimation × DREAMPlace [DATE’21]**

---

**Backward Propagation**
- Gradient w.r.t. locations \( \nabla_x L, \nabla_y L \)
- Gradient w.r.t. features \( \nabla_M L \)
- Gradient w.r.t. congestion map \( \nabla_{f(R(M))} L \)

**Forward Propagation**
- Cell Locations \((x, y)\)
- Features \(M \in R^{M \times N \times 3}\)
- RUDY \((x, y)\)
- PinRUDY \((x, y)\)
- MacroRegion \((x, y)\)

**Network**
- Forward
- Backward

**Congestion Penalty**
\[ L = \frac{1}{MN} \| f_R(M) \|_2^2 \]
Outline

Background: Placement Algorithms

New Direction: Integrating GPU Power into Placement

New Direction: Datapath aware Placement
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]
- General ASIC design [Ye, ISCAS’02] [Chou, DAC’12] [Wang, IETCDS’17]
- ...
These slides contain/adapt materials developed by

Draw the Placement Together!!