Xplace: An Extremely Fast and Extensible Global Placement Framework

Lixin Liu, Bangqi Fu, Martin D.F. Wong, Evangeline F.Y. Young

CSE Department
The Chinese University of Hong Kong
Global Placement Problem

A fundamental step in VLSI physical design
• Highly affect the circuit’s PPA

Modern circuits contain millions of standard cells
• Highly increase the computational complexity of GP
• Bring huge challenges to the leading-edge global placers
Global Placement Problem

Objective:
• Minimize the total HPWL of all the nets
• Satisfy the cell density constraint

Analytical Global Placement:
• A smooth approximation of HPWL
• A density penalty

\[
\begin{align*}
\min_p \text{HPWL}(p) &= \min_p \sum_{e \in E} \text{HPWL}_e(p) \\
\text{s.t. } D_b &\leq D_t, \forall b \in B
\end{align*}
\]

\[
\begin{align*}
\min_p \sum_{e \in E} \text{WLe}(p) + \lambda D(p)
\end{align*}
\]

GPU-accelerated Global Placers

- Rapid development of GPU's computational power
- GPU acceleration becomes an important direction

Recently, DREAMPlace[1]
- Implemented the approach of ePlace[2] on GPU
- Produced the SOTA solution quality and performance

It is a big challenge to further improve on DREAMPlace’s performance.

Proposed Framework: Xplace

Placement Core Engine

- Cell Pos $p_t \in \mathbb{R}^{N \times 2}$
- Params $\theta_t \in \mathbb{R}^P$

Grad Engine

- Calc WL / Density Grad
- $\nabla_{W/L} p_t \in \mathbb{R}^{N \times 2}$
- $\nabla_D p_t \in \mathbb{R}^{N \times 2}$

Calc Total Grad

- Cell Grad $\nabla p_t \in \mathbb{R}^{N \times 2}$

Optimizer

- Next Step Cell Pos $p_{t+1} \in \mathbb{R}^{N \times 2}$

Next Step Params $\theta_{t+1} \in \mathbb{R}^P$

- Params Scheduler
  - Metrics $\in \mathbb{R}^{M \times (t+1)}$
- Recorder
  - HPWL $\in \mathbb{R}$
  - OVFL $\in \mathbb{R}$
- Evaluator

PyTorch

Deep Learning Library
Operator-Level Optimization

1. Wirelength Operator Combination (OC)

**Observation:** Both the HPWL function and the stable WA wirelength function need the min and max cell positions in a net.

**Method:** combining the three operators with heavy wirelength-related workload, **WA wirelength**, **WA gradient** and **HPWL**, into one operator

**Result:** avoid redundant computation of the min and max function
Operator-Level Optimization

2. Density Operator Extraction (OE)

Overflow ratio and density computation:

\[ OVF_{FL} = \frac{\sum_{b \in B} \max(D_b - D_t, 0)A_b}{\sum_{i \in V_{mov}} A_i} \]

\[ D_b = \frac{\sum_{i \in V} A_i \cap A_b}{A_b}, \quad \forall b \in B \]

\( D_t \): target density, \( D_b \): bin \( b \)'s cell density, \( A_b \) and \( A_t \) denote the area for bin \( b \) and cell \( i \),
Operator-Level Optimization

2. Density Operator Extraction (OE)

Overflow ratio and density computation:

\[
OV_{FL} = \frac{\sum_{b \in B} \max(D_b - D_t, 0) A_b}{\sum_{i \in V_{mov}} A_i} \quad D_b = \frac{\sum_{i \in V} A_i \cap A_b}{A_b}, \quad \forall b \in B
\]

\(D_t\): target density, \(D_b\): bin b’s cell density, \(A_b\) and \(A_t\) denote the area for bin b and cell i,

Need to insert filler cells inside the electrostatic system [1]

\[
\tilde{D}_b = \frac{\sum_{i \in V \cup V_{fl}} A_i \cap A_b}{A_b} = D_b + \frac{\sum_{i \in V_{fl}} A_i \cap A_b}{A_b}, \quad \forall b \in B
\]

\(V\): the set of cells, \(V_{fl}\): the set of fillers, \(\tilde{D}_b\): bin b’s total density (incl. filler density)

Operator-Level Optimization

2. Density Operator Extraction (OE)

Overflow ratio and density computation:

\[
OVFL = \frac{\sum_{b \in B} \max(D_b - D_t, 0)A_b}{\sum_{i \in V_{mov}} A_i}
\]

\[
D_b = \frac{\sum_{i \in V} A_i \cap A_b}{A_b}, \forall b \in B
\]

\(D_t\): target density, \(D_b\): bin \(b\)'s cell density, \(A_b\) and \(A_t\) denote the area for bin \(b\) and cell \(i\),

Need to insert filler cells inside the electrostatic system [1]

\[
\tilde{D} = D + D_{fl}
\]

Matrix form of the total density map. \(\tilde{D}, D, D_{fl} \in \mathbb{R}^{M \times M}, M\) is the grid size

Operator-Level Optimization

2. Density Operator Extraction (OE)

\[ OVFL = \frac{\sum_{b \in B} \max(D_b - D_i, 0) A_b}{\sum_{i \in V_{mov}} A_i} \]

\[ \tilde{D} = D + D_{fl} \]

**Observation:** Both the calculation of \( OVFL \) and total density map \( \tilde{D} \) need the cell density map \( D \).

**Method:** common sub-operator \( D \) extraction, compute the cell density map \( D \) and the filler density map \( D_{fl} \) separately.
Operator-Level Optimization

2. Density Operator Extraction (OE)

\[ OVL = \frac{\sum_{b \in B} \max(D_b - D_t, 0)A_b}{\sum_{i \in \mathcal{V}_{\text{mov}}} A_i} \]

\[ \tilde{D} = D + D_{fl} \]

**Observation**: Both the calculation of \( OVL \) and total density map \( \tilde{D} \) need the cell density map \( D \).

**Result**: reduce the total computation time of the cell density map \( D \).
3. Operator Reduction (OR)

**Observation:**
- The number of forward operators are almost the same as that in the backward
- Invoking the heavy autograd engine will almost **double the number of operators**
  and bring **large kernel launching overhead on CPU**

**Method:**
- Avoid invoking the heavy **autograd engine**
- Directly derive the numerical solutions of the WL / density grad
- Assign a weighted accumulated gradient to each cell

**Result:** Reduce the total kernel launching time
Operator-Level Optimization

3. Operator Reduction (OR)

**Other Methods:**
- Use in-place operators as much as possible
  - Avoid redundant copying
- Reorder the operators that need sync to the end of the execution queue
  - Reduce the frequency of interrupting the GPU pipeline
Operator-Level Optimization

4. Operator Skipping (OS)

**Observation:**

• The ratio $r = \frac{\lambda |\nabla D_{x,y}|}{|\nabla W_{Lx,y}|}$ is ultra-small in the early placement stage

**Method:**

• When $(r < 0.01) \land (\text{iter} < 100)$, the density grad operator will only be executed once per 20 iterations

**Result:**

• Skip some density grad calculation in early placement stage
Placement-Stage-Aware Parameters Scheduling

Precondition matrix of $\tilde{H}^{-1} = (H_W + \lambda H_D)^{-1}$ is applied to accelerate convergence [1]

$$H_W = \text{diag}(|S_1|, |S_2|, ..., |S_N|) \quad H_D = \text{diag}(A_1, A_2, ..., A_N)$$

$|S_i|$: the number of nets connecting cell $i$, $A_i$ the area of cell $i$

We introduce the precondition weighted ratio $\omega = \frac{\lambda |H_D|}{|H_W| + \lambda |H_D|} \in [0,1]$ to measure the placement stage

Placement-Stage-Aware Parameters Scheduling

Precondition weighted ratio \( \omega = \frac{\lambda |H_D|}{|H_W| + \lambda |H_D|} \in [0,1] \)

- \( \omega > 0.95 \) cells are forced to a final position with minimum local penalty
- \( 0.05 < \omega < 0.95 \) cells are spreading over the whole map and the overlap ratio significantly decreases
- \( \omega < 0.05 \) wirelength-dominated and cells are driven to the position with minimum wirelength
Placement-Stage-Aware Parameters Scheduling

Precondition weighted ratio $\omega = \frac{\lambda |H_D|}{|H_W| + \lambda |H_D|} \in [0,1]$

To fully exploit the optimization space

Algorithm 1 Placement-Stage-Aware Parameters Scheduling

1: $\gamma \leftarrow \gamma_0$ \hspace{1em} \triangleright wirelength coefficient
2: $\lambda \leftarrow \lambda_0$ \hspace{1em} \triangleright density weight
3: while iteration < ITER and NOT Convergence do
4: \hspace{1em} if $0.5 < \omega < 0.95$ and iteration%3 ≠ 0 then
5: \hspace{2em} SKIP_UPDATE
6: \hspace{1em} else
7: \hspace{2em} $\gamma \leftarrow \gamma \times \text{coef}(overflow)$
8: \hspace{2em} $\lambda \leftarrow \lambda \times \mu(\Delta h p w l)$ \hspace{1em} \triangleright Both $\gamma$ and $\lambda$ are derived from [10]
9: \hspace{2em} $\omega \leftarrow \frac{\lambda |H_D|}{|H_W| + \lambda |H_D|}$

ISPD 2005 / adaptec1
Extending the Framework via Neural Enhancement

How to solve a 2D PDE problem by deep learning?

Image-to-image networks -> Solve PDE in spatial domain conv

2D Fourier-Neural-Operator (FNO) [1] -> Solve PDE in frequency domain conv

Extending the Framework via Neural Enhancement

How to solve a 2D PDE problem by deep learning?

× Image-to-image networks -> Solve PDE in spatial domain conv

✓ 2D Fourier-Neural-Operator (FNO) [1] -> Solve PDE in frequency domain conv

Poisson's Equation

\[
\begin{align*}
\nabla \cdot \nabla \psi(x, y) &= -\rho(x, y), \\
\hat{n} \cdot \nabla \psi(x, y) &= 0, (x, y) \in \partial R, \\
\iint_R \rho(x, y) &= \iint_R \psi(x, y) = 0,
\end{align*}
\]

Electron Distribution $\rho$ -> 2D Density map $D$ of placement

Electric Field $\nabla \psi_x, \nabla \psi_y$ -> moving force on x and y-axis

Many PDEs can be solved by Fourier transform.

Extending the Framework via Neural Enhancement

Input \( I = \{D; M_x; M_y\} \)

- Density map \( D \)
  \[ M_x(x, y) = \frac{x}{X} \]
  \[ M_y(x, y) = \frac{y}{Y} \]
  \( X, Y \) are the map sizes

Input transform: \( I_m = FC(I) \)

- \( Freq_{layer}(I_m) = \mathcal{F}^{-1}\left(\mathcal{W}^T \cdot L(\mathcal{F}(I_m))\right) \)

Output transform: \( FC^{-1}\left(O(I_m)\right) \)

- \( \mathcal{F} \): FFT, \( \mathcal{F}^{-1} \): IFFT
- \( FC \): fully-connected layer, \( L \): low-pass-filter

Relative L2 Loss:

\[ L_2(x_i, f(x_i; \theta)) = \frac{||f(x_i, \theta) - y_i||_2}{||y_i||_2} \]
Extending the Framework via Neural Enhancement

Model Training Data Collection

1. ISPD 2005 contest benchmarks with their respective macros
2. Standard cells are randomly generated at a starting position
3. Pushed cells all over the map with only the density objective $D(p)$
4. The density map and electric fields are used as training data and labels

Why train the model in low-resolution data

1. The resolution of the input maps will not affect the convolution results
2. Low frequency components describe the global information
3. Improve the adaptability of the model and speedup inference
Extending the Framework via Neural Enhancement

How to apply the nn-predicted density gradient $\nabla_{nn} D_{x,y}$

Smooth function: $\sigma(\omega) = 1 - 1/(1 - 5e^{\omega/0.05-0.5})$

Total gradient: $\nabla' D_{x,y} = (1 - \sigma)\nabla D_{x,y} + \sigma \nabla_{nn} D_{x,y}$
## Experimental Results

### Validation on Contest Benchmarks

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>DREAMPlace [14]</th>
<th>Xplace</th>
<th>ISPD 2005</th>
<th>ISPD 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HPWL</td>
<td>GP/s</td>
<td>DP/s</td>
<td>HPWL</td>
</tr>
<tr>
<td>adaptec1</td>
<td>72.89</td>
<td>4.15</td>
<td>34.9</td>
<td>72.93</td>
</tr>
<tr>
<td>adaptec2</td>
<td>81.84</td>
<td>3.73</td>
<td>46.2</td>
<td>81.04</td>
</tr>
<tr>
<td>adaptec3</td>
<td>191.68</td>
<td>4.54</td>
<td>88.1</td>
<td>190.94</td>
</tr>
<tr>
<td>adaptec4</td>
<td>173.45</td>
<td>4.90</td>
<td>95.4</td>
<td>172.41</td>
</tr>
<tr>
<td>bigblue1</td>
<td>89.39</td>
<td>4.03</td>
<td>42.3</td>
<td>89.12</td>
</tr>
<tr>
<td>bigblue2</td>
<td>136.57</td>
<td>4.68</td>
<td>129.3</td>
<td>136.56</td>
</tr>
<tr>
<td>bigblue3</td>
<td>302.58</td>
<td>8.05</td>
<td>207.9</td>
<td>301.36</td>
</tr>
<tr>
<td>bigblue4</td>
<td>742.95</td>
<td>13.38</td>
<td>459.7</td>
<td>741.18</td>
</tr>
<tr>
<td>superblue12</td>
<td>25803.0</td>
<td>92.45</td>
<td>89.1</td>
<td>19633.1</td>
</tr>
<tr>
<td>superblue14</td>
<td>23015.5</td>
<td>63.56</td>
<td>4.63</td>
<td>21205.0</td>
</tr>
<tr>
<td>superblue19</td>
<td>15833.1</td>
<td>61.82</td>
<td>4.56</td>
<td>27867.6</td>
</tr>
<tr>
<td>des_perf_a</td>
<td>4217.9</td>
<td>80.30</td>
<td>3.97</td>
<td>9017.9</td>
</tr>
<tr>
<td>des_perf_b</td>
<td>27867.6</td>
<td>44.86</td>
<td>3.82</td>
<td>3267.9</td>
</tr>
<tr>
<td>edit_dist_a</td>
<td>361.8</td>
<td>30.55</td>
<td>3.54</td>
<td>361.8</td>
</tr>
<tr>
<td>matrix_mult_b</td>
<td>741.1</td>
<td>22.89</td>
<td>6.77</td>
<td>741.1</td>
</tr>
<tr>
<td>matrix_mult_c</td>
<td>33411.2</td>
<td>54.51</td>
<td>5.59</td>
<td>33411.2</td>
</tr>
<tr>
<td>pci_bridge32_a</td>
<td>25600.9</td>
<td>65.85</td>
<td>4.38</td>
<td>25600.9</td>
</tr>
</tbody>
</table>

| Sum          | 148571 | 1129.70 | 88.03 | 82.06 | 148364 | 1129.77 | 31.03 | 82.84 |
| Ratio        | 1.001  | 1.000   | 2.837 | 0.991 | 1.000  | 1.000   | 1.000 | 1.000 |
## Experimental Results

### Ablation Studies of the Operator-Level Optimization Techniques

<table>
<thead>
<tr>
<th>Methods</th>
<th>OR</th>
<th>OC</th>
<th>OE</th>
<th>OS</th>
<th>adaptec1</th>
<th>adaptec2</th>
<th>adaptec3</th>
<th>adaptec4</th>
<th>bigblue1</th>
<th>bigblue2</th>
<th>bigblue3</th>
<th>bigblue4</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>234%</td>
<td>194%</td>
<td>136%</td>
<td>124%</td>
<td>198%</td>
<td>140%</td>
<td>123%</td>
<td>121%</td>
<td>159%</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>110%</td>
<td>109%</td>
<td>113%</td>
<td>115%</td>
<td>105%</td>
<td>115%</td>
<td>119%</td>
<td>118%</td>
<td>113%</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>107%</td>
<td>107%</td>
<td>107%</td>
<td>108%</td>
<td>104%</td>
<td>108%</td>
<td>113%</td>
<td>112%</td>
<td>108%</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>104%</td>
<td>102%</td>
<td>104%</td>
<td>104%</td>
<td>102%</td>
<td>104%</td>
<td>106%</td>
<td>105%</td>
<td>104%</td>
</tr>
<tr>
<td>Xplace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ratio</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>GP / Iter Time (ms)</td>
<td>1.478</td>
<td>1.671</td>
<td>2.325</td>
<td>2.688</td>
<td>1.572</td>
<td>2.441</td>
<td>4.974</td>
<td>10.018</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DREAMPlace</td>
<td>Ratio</td>
<td>462%</td>
<td>345%</td>
<td>288%</td>
<td>254%</td>
<td>376%</td>
<td>288%</td>
<td>199%</td>
<td>158%</td>
<td>296%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GP / Iter Time (ms)</td>
<td>6.832</td>
<td>5.769</td>
<td>6.699</td>
<td>6.840</td>
<td>5.915</td>
<td>7.023</td>
<td>9.904</td>
<td>15.831</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Experimental Results

#### Neural-Enhanced Performance

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>DREAMPlace</th>
<th>Xplace</th>
<th>Xplace-NN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HPWL</td>
<td>GP/s</td>
<td>DP/s</td>
</tr>
<tr>
<td>adaptec1</td>
<td>72.89</td>
<td>4.15</td>
<td>34.9</td>
</tr>
<tr>
<td>adaptec2</td>
<td>81.84</td>
<td>3.73</td>
<td>46.2</td>
</tr>
<tr>
<td>adaptec3</td>
<td>191.68</td>
<td>4.54</td>
<td>88.1</td>
</tr>
<tr>
<td>adaptec4</td>
<td>173.45</td>
<td>4.90</td>
<td>95.4</td>
</tr>
<tr>
<td>bigblue1</td>
<td>89.39</td>
<td>4.03</td>
<td>42.3</td>
</tr>
<tr>
<td>bigblue2</td>
<td>136.57</td>
<td>4.68</td>
<td>129.3</td>
</tr>
<tr>
<td>bigblue3</td>
<td>302.58</td>
<td>8.05</td>
<td>207.9</td>
</tr>
<tr>
<td>bigblue4</td>
<td>742.95</td>
<td>13.38</td>
<td>459.7</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td>1791.36</td>
<td>47.46</td>
<td>1103.6</td>
</tr>
<tr>
<td><strong>Ratio</strong></td>
<td>1.003</td>
<td>1.626</td>
<td>0.995</td>
</tr>
</tbody>
</table>
Conclusions and Future Works

Conclusions

We develop Xplace, a new, fast and extensible GPU accelerated GP framework built on top of PyTorch, to consider factors at operator-level optimization.

• **Efficiency**: Xplace achieves around 3x speedup per GP iter with better quality compared to DREAMPlace

• **Extensibility**: we plug into Xplace a novel Fourier neural network and illustrate a possibility of adopting neural guidance in analytical global placement

Future Works

• Handling additional constraints in placement like routability and fence regions