IncreMacro: Incremental Macro Placement Refinement

Yuan Pu$^{1,2}$, Tinghuan Chen$^1$, Zhuolun He$^{1,2}$, Chen Bai$^1$, Haisheng Zheng$^2$, Yibo Lin$^3$, Bei Yu$^1$

$^1$The Chinese University of Hong Kong, Hong Kong SAR
$^2$Shanghai AI Laboratory, Shanghai, China
$^3$Peking University, Beijing, China
Introduction
Macro placement plays an important role for the QoR of the following physical design flows (e.g. std placement, routing).

Analytical placers (e.g. DREAMPlace\(^1\)) and AutoDMP\(^2\) may cause macro blockage, leading to discontinuous space for standard cell placement, thus bad wirelength, routability and timing.

---


Data Structure-based Metaheuristics

Methodology:

- Three-stage macro placement. (placement prototype → macro placement → standard cell placement).
- Macro positions → specific data structure (e.g., MP tree$^3$).
- Simulated Annealing: iteratively perturbation.

Objective: push macros to chip boundary, optimize macro displacement, wirelength, routability, etc.

Disadvantage:

- Disturb relative macro positions by placement prototype → disturb the timing optimization by analytical placers.

---

RL-based Approach\(^4\).  
Macro placement $\rightarrow$ sequential Markov Decision Process (MDP).  
- State: intermediate macro placement solution.  
- Action: assign one macro to a legalized position on the chip.  
Generate macro placement solution **from scratch**.  
Disadvantages:  
- From scratch $\rightarrow$ computationally prohibitive for parameters exploration.  
- No utilization of timing-optimization by analytical placement prototype.


Incremental Macro Placement Refinement

Given the placement prototype by an analytical placer, find the legalized coordinates of macros to optimize the objective of wirelength.

Requirements satisfied by IncreMacro:

- Pushing macros to chip periphery.
- Macro relative position (timing-opt by analytical placers) preserved.
Algorithm
Overall flow of IncreMacro:
• Regularly-placed macro satisfies one of the following conditions:
  • clings to chip boundary
  • nearest four 4 macros surround it

• Motivation: only poorly-placed macros need to be adjusted.

• Methodology for macro diagnosis: KD-tree based nearest neighbor search

• Example:
  • \(m_1\): poorly-placed
  • \(m_3\): regularly-placed
The Macro shifting gradient descent process.

- Make analogy between macro placement and deep learning:
  - macro coordinate offset $\rightarrow$ model weight
  - wirelength and periphery cost $\rightarrow$ cost function
  - netlist $\rightarrow$ data instance
- Implemented by pytorch
- Only shift poorly-placed macros

$$\min \mathcal{L} = \sum_{e \in \text{Net}_{macro}} WL_e + \alpha \sum_{m_i \in \mathcal{M}_{poor}} P_i,$$

where $WL_e$ is wirelength cost for net $e$, $P_i$ is periphery cost for macro $m_i$. 
• Half-perimeter wirelength (HPWL) for net $e$:

$$\text{HPWL}_e = \max_{v_i \in e} \{x_i\} - \min_{v_i \in e} \{x_i\} + \max_{v_i \in e} \{y_i\} - \min_{v_i \in e} \{y_i\}. \quad (2)$$

• HPWL is not differentiable.

• Weighted-average wirelength (WA)\textsuperscript{6} for a net $e$:

$$\text{WA}_e = \sum_{v_i \in e} x_i e^{\frac{x_i}{\gamma}} - \sum_{v_i \in e} x_i e^{-\frac{x_i}{\gamma}} + \sum_{v_i \in e} y_i e^{\frac{y_i}{\gamma}} - \sum_{v_i \in e} y_i e^{-\frac{y_i}{\gamma}}. \quad (3)$$

• Use WA as wirelength cost.

Gradient-based Macro Shift: Periphery cost

- Motivation: push macros to chip boundary.

\[
P_i^H = \left| \frac{w_{\text{core}}}{2} - x_i \right| + \frac{\left( \frac{w_{\text{core}}}{2} \right)^2}{\left| \frac{w_{\text{core}}}{2} - x_i \right|},
\]

\[
P_i^V = \left| \frac{h_{\text{core}}}{2} - y_i \right| + \frac{\left( \frac{h_{\text{core}}}{2} \right)^2}{\left| \frac{h_{\text{core}}}{2} - y_i \right|},
\]

\[
P_i = P_i^H + P_i^V.
\]
• **Purpose:**
  - Eliminate overlaps generated by last step (Macro shift)
  - Further push macros to nearest chip periphery

• **Constraint graphs:**
  - Relative positional relationship among macros, horizontally and vertically.
  - Constructed on the macro placement prototype by analytical placers.

• **Objective for LP (minimization):**
  - Macro displacement
  - Distance to the nearest periphery

---

Macro Legalization: Linear Programming Formulation

\[
\begin{align*}
\min & \quad \sum_{i=1}^{\left|M_{poor}\right|} |x'_i - x_i| + |y'_i - y_i| + \min(x_i, W - x_i) + \min(y_i, H - y_i) \\
\text{s.t.} \quad & \quad x'_j - x'_i \geq \frac{w_i + w_j}{2}, \quad \forall e_{ij} \in G_h \\
& \quad y'_j - y'_i \geq \frac{h_i + h_j}{2}, \quad \forall e_{ij} \in G_v \\
& \quad \frac{w_i}{2} \leq x'_i \leq W - \frac{w_i}{2}, \quad \frac{h_i}{2} \leq y'_i \leq H - \frac{h_i}{2}, \quad \forall m_i \in M_{poor}
\end{align*}
\]

(5)

- \(x_i/x'_i\): x-coordinate before/after macro legalization.
- Original objective for macro displacement: \(|x'_i - x_i| + |y'_i - y_i|\).
- \(\min(x_i, W - x_i) + \min(y_i, H - y_i)\): distance to the nearest boundary.
- First and second inequality: non-overlap constraint.
- Last inequality: out of boundary constraint.
Macro Legalization: Linear Programming Formulation (Con’t)

\[
\begin{align*}
\min & \quad \sum_{i=1}^{|M_{\text{poor}}|} x_i^p + x_i^q + y_i^p + y_i^q + \min(x_i, W - x_i) + \min(y_i, H - y_i) \\
\text{s.t.} & \quad x_i^p - x_i^q = x'_i - x_i, \quad y_i^p - y_i^q = y'_i - y_i, \\
& \quad x_i^p \geq 0, \quad x_i^q \geq 0, \quad y_i^p \geq 0, \quad y_i^q \geq 0, \\
& \quad x'_j - x'_i \geq \frac{w_i + w_j}{2}, \\
& \quad y'_j - y'_i \geq \frac{h_i + h_j}{2}, \\
& \quad \frac{w_i}{2} \leq x'_i \leq W - \frac{w_i}{2}, \quad \frac{h_i}{2} \leq y'_i \leq H - \frac{h_i}{2}.
\end{align*}
\]

\[\forall m_i \in M_{\text{poor}}\]
\[\forall m_i \in M_{\text{poor}}\]
\[\forall e_{ij} \in G_h\]
\[\forall e_{ij} \in G_v\]
\[\forall m_i \in M_{\text{poor}}\]

(6)

- First and second constraints: transformation on displacement objective (remove absolute value).
Experimental Results
Experiment Setting

- Benchmark: OpenSource RISC-V SOCs from chipyard\(^8\).
- PDK: ASAP7\(^9\).
- VLSI flow: standard cell placement $\rightarrow$ CTS $\rightarrow$ routing (commercial tool).
- Baseline: DREAMPlace, AutoDMP.

**Table:** Design information

<table>
<thead>
<tr>
<th>Benchmark</th>
<th># Macros</th>
<th># Std cells</th>
<th># Nets</th>
<th>freq. (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rocket</td>
<td>121</td>
<td>203633</td>
<td>208595</td>
<td>500.0</td>
</tr>
<tr>
<td>GemminiRocket</td>
<td>737</td>
<td>1176141</td>
<td>1189717</td>
<td>333.3</td>
</tr>
<tr>
<td>Sha3Rocket</td>
<td>121</td>
<td>235944</td>
<td>240907</td>
<td>666.7</td>
</tr>
<tr>
<td>LargeBoomAndRocket</td>
<td>636</td>
<td>856576</td>
<td>874017</td>
<td>333.3</td>
</tr>
<tr>
<td>FFTRocket</td>
<td>121</td>
<td>211543</td>
<td>216507</td>
<td>1000.0</td>
</tr>
<tr>
<td>SmallBoom</td>
<td>314</td>
<td>362479</td>
<td>372623</td>
<td>500.0</td>
</tr>
<tr>
<td>HwachaRocket</td>
<td>292</td>
<td>679667</td>
<td>687204</td>
<td>250.0</td>
</tr>
</tbody>
</table>


### Experiment: Post-Route (DREAMPlace)

#### Table: Post-Route PPA results for the incorporation of IncreMacro to DREAMPlace

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>DREAMPlace (two stage placement)</th>
<th>DREAMPlace + IncreMacro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL (um)</td>
<td>WNS (ns)</td>
</tr>
<tr>
<td>Rocket</td>
<td>14838552</td>
<td>-1.40</td>
</tr>
<tr>
<td>GemminiRocket</td>
<td>62420588</td>
<td>-3.06</td>
</tr>
<tr>
<td>Sha3Rocket</td>
<td>11584499</td>
<td>-0.22</td>
</tr>
<tr>
<td>LargeBoomAndRocket</td>
<td>36201457</td>
<td>0</td>
</tr>
<tr>
<td>FFTRocket</td>
<td>10749086</td>
<td>-0.01</td>
</tr>
<tr>
<td>SmallBoom</td>
<td>13967377</td>
<td>-0.13</td>
</tr>
<tr>
<td>HwachaRocket</td>
<td>48096700</td>
<td>-0.10</td>
</tr>
<tr>
<td>Normalize</td>
<td>1.065</td>
<td>1.599</td>
</tr>
</tbody>
</table>

#### Performance Improvement:

- **wirelength:** +6.5%
- **power:** +3.3%
- **WNS:** +59.9%
- **TNS:** +63.9%
### Table: Post-Route PPA results for the incorporation of IncreMacro to AutoDMP

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>AutoDMP (two-stage placement)</th>
<th>AutoDMP + IncreMacro</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL (um)</td>
<td>WNS (ns)</td>
</tr>
<tr>
<td>Rocket</td>
<td>14704880</td>
<td>-0.41</td>
</tr>
<tr>
<td>GemminiRocket</td>
<td>41031737</td>
<td>-0.24</td>
</tr>
<tr>
<td>FFTRocket</td>
<td><strong>8108137</strong></td>
<td>-0.001</td>
</tr>
<tr>
<td>HwachaRocket</td>
<td>31675727</td>
<td>-0.05</td>
</tr>
<tr>
<td>Normalize</td>
<td>1.168</td>
<td>1.996</td>
</tr>
</tbody>
</table>

**Performance Improvement:**

- **wirelength:** +16.8%
- **power:** +4.9%
- **WNS:** +99.6%
- **TNS:** +99.9%
Result Visualization

(f) GemminiRocket: DREAMPlace
(g) GemminiRocket: DREAMPlace + IncreMacro
(h) GemminiRocket: AutoDMP
(i) GemminiRocket: AutoDMP + IncreMacro
Runtime Analysis

Average runtime breakdown of IncreMacro + DREAMPlace.
Conclusion
• IncreMacro: incrementally enhances macro placement by SOTA analytical placers.

• Three main techniques:
  • KD- tree-based macro diagnosis
  • gradient-based macro shift
  • constraint-graph-based linear programming for macro legalization

• Two requirements for macro placement:
  • push macros to the chip boundary
  • Macro relative position preserved
THANK YOU!