FuILT: Full Chip ILT System With Boundary Healing

Shuo Yin\textsuperscript{1}, Wenqian Zhao\textsuperscript{1}, Li Xie\textsuperscript{2}, Hong Chen\textsuperscript{2}, Yuzhe Ma\textsuperscript{3}, Tsung-Yi Ho\textsuperscript{1}, Bei Yu\textsuperscript{1}

\textsuperscript{1}Chinese University of Hong Kong
\textsuperscript{2}Shenzhen GWX Technology Co., Ltd.
\textsuperscript{3}Hong Kong University of Science and Technology (Guangzhou)
Outline

1. Motivation
2. FuILT: Full Chip ILT System
3. Experimental Results
4. Conclusion
Motivation
Inverse Lithography Technology (ILT) treats the mask as a pixel-wise image and optimizes the mask shape to compensate for optical diffraction in order to get a better print image.

ILT Algorithms:

- MOSAIC\(^1\)
- Neural-ILT\(^2\)
- Levelset-GPU\(^3\)

---


Problem Of Full Chip OPC

- Memory Limitation: The mask/target pixel image exceeds the capacity of main memory for storage.
- Computation Limitation: Simulation or optimization processes are computationally intensive.

Lithography defects:

Visualization of lithography defects after mask stitching.
Previous Works

The meshing partition strategy of DAC’22\textsuperscript{4}, ASPDAC’23\textsuperscript{5}, TCAD’23\textsuperscript{6}.

![Diagram](image)

However, the boundary error remains undiscussed.


\textsuperscript{6}Guojin Chen et al. (2023). “DevelSet: Deep neural level set for instant mask optimization”. In: IEEE TCAD.
FuILT: Full Chip ILT System
Full Chip ILT System:

- **Multi-level Partition**: Macro-level meshing; Micro-level recursive partition
- **Distributed Optimization**: Server-Worker strategy; Multi-GPU pipeline
- **Multi-level Stitching/Healing**: Micro-level gradient fusion; Macro-level mask healing

Full-chip ILT system overall workflow
On top of parallelism, macro-level partition will handle memory bound first by cutting the whole design layer into grids of large patches.

\[
m^*, n^* = \arg\min_{m, n} \sum_{i} (H_{P_i} + W_{P_i}),
\]

s.t.

\[
\sum_{i} H_{P_i} < H_P + H'_{max}
\]

\[
\sum_{i} W_{P_i} < W_P + W'_{max},
\]

\[
\text{Size}(P_i) + \text{Size}(G_i) \leq \text{MemSize}.
\]

\(P_i\) represents the \(i\)-th patch and \(G_i\) denotes the gradient of \(i\)-th patch, \(H'_{max}/W'_{max}\) represent the max overlapping length, \(H_P/W_P\) and \(H_{P_i}/W_{P_i}\) denote height/width of the patch.
At the micro-level, we start handling each patch while considering the computational capability of GPUs.

We slice each large patch into four tiles and repeat recursively until each tile satisfies the mentioned GPU computation limits.

\[ T_1, \ldots, T_4 = \begin{bmatrix} T_1[s_h \times s_w] & T_3[s_h \times s_w] \\ T_2[s_h \times s_w] & T_4[s_h \times s_w] \end{bmatrix} \]  

At each level, a tile is further sliced into “□” grid of four smaller tiles with overlapping rate \( R \).
**Server:** distribute the workload to each worker and gather the gradients from the workers.

**Worker:** workers are responsible for computing the tile gradient by adapting the one-time ILT forward and backward process.

---

### Server-Worker strategy

- **Server:**
  - Load Data
  - Gather Gradient

- **Workers:**
  - Compute 
  - Loss 
  - Gradient Derivation 
  - Load 
  - Data 
  - Workload 
  - Pipelining 
  - Communication 
  - Gather 

### Multi-GPU Pipeline

- **Workload Pipelining:**
  - Loop $i$
  - Partition \times k
  - Load Data

- **Within Each GPU:**
  - Compute 
  - Loss 
  - Gradient Derivation

- **Loop $i + 1$:**
  - Gather 
  - Load Data

---

**Distributed Optimization**
Once we have the gradients $g_1, \ldots, 4^{k*}$ of all small tiles $T_1, \ldots, 4^{k*}$ partitioned from large patch $P_i$, we will fuse the gradient matrices back to a large gradient map $G_i^{(t)}$ in the iteration $t$.

Gradient fusion on the overlapping area.
After obtaining the masks of large patches with inner boundaries fixed, we also need to tackle the stitching boundary of these large patches.

Considering the huge patch size, we only apply a small healing box $M$ on the boundary area to heal stitching errors.

Macro-level boundary healing strategy. The healing box is the ILT lithography area. We add a mask on the gradient $\frac{\partial L}{\partial M}$ during backward propagation. The dark blue area is 1 where the gradient is kept, and the light blue area is 0 where the gradient is neglected.
Macro-level Stitching & Boundary Healing

Visualization of healing effect. The left figure is the overlapping area without macro-level healing. The right figure is the same area after macro-level healing. This crop is from our Metal-1 layer result.
Experimental Results
The benchmark used in our experiments is from an actual GCD design which generated by OpenROAD\(^7\) in FreePDK45\(^8\) process design kit.

### Table: Benchmark Details

<table>
<thead>
<tr>
<th>Bench</th>
<th>#Polygons</th>
<th>Bounding Box ((nm \times nm))</th>
<th>Total Area ((nm^2))</th>
<th>Average Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metal</td>
<td>5043</td>
<td>80528 × 80192</td>
<td>1675692925</td>
<td>5.8</td>
</tr>
<tr>
<td>Via</td>
<td>5411</td>
<td>73696 × 68992</td>
<td>22861474</td>
<td>4.0</td>
</tr>
<tr>
<td>Poly</td>
<td>1126</td>
<td>73696 × 68992</td>
<td>88904249</td>
<td>5.1</td>
</tr>
<tr>
<td>Pimplant</td>
<td>1830</td>
<td>80528 × 80192</td>
<td>3892569599</td>
<td>4.0</td>
</tr>
</tbody>
</table>

\(^7\)OpenROAD (n.d.). [https://github.com/The-OpenROAD-Project/OpenROAD.](https://github.com/The-OpenROAD-Project/OpenROAD)  
\(^8\)FreePDK45 (n.d.). [https://eda.ncsu.edu/freepdk/freepdk45/](https://eda.ncsu.edu/freepdk/freepdk45/)
## Table: Comparisons of EPE, PVBand, $L_2$ Loss and Runtime

<table>
<thead>
<tr>
<th>Bench</th>
<th>DAC’22 w/o. Macro Boundary Healing</th>
<th>FuILTw/o. Macro Boundary Healing</th>
<th>FuILT w. Macro Boundary Healing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#EPE PVB ($nm^2$) $L_2$ ($nm^2$) RT (s) $S(\times 10^7)$</td>
<td>#EPE PVB ($nm^2$) $L_2$ ($nm^2$) RT (s) $S(\times 10^7)$</td>
<td>#EPE PVB ($nm^2$) $L_2$ ($nm^2$) RT (s) $S(\times 10^7)$</td>
</tr>
<tr>
<td>Metal-1</td>
<td>24668 120961048 224397927 2956 83.16</td>
<td>1243 66938717 158733917 3179 43.27</td>
<td>1221 66810974 154685729 4563 42.80</td>
</tr>
<tr>
<td>Via</td>
<td>127 6520380 9851354 1827 3.66</td>
<td>10 6445143 7931745 1902 3.38</td>
<td>10 6432513 7815421 2741 3.36</td>
</tr>
<tr>
<td>Poly</td>
<td>164 40238769 43742084 1831 20.55</td>
<td>59 32249748 31708247 1936 16.10</td>
<td>53 32055284 30297295 2836 15.88</td>
</tr>
<tr>
<td>Pimplant</td>
<td>4885 76507491 58102567 2837 38.86</td>
<td>1674 76389450 56955213 3174 37.09</td>
<td>1668 76310462 56664328 4475 37.03</td>
</tr>
<tr>
<td>Total</td>
<td>29844 244227688 336093932 9451 146.23</td>
<td>2986 182023058 255329122 10191 99.84</td>
<td>2952 181609233 249462773 14615 99.07</td>
</tr>
<tr>
<td>Ratio</td>
<td>9.99 1.34 1.31 1.00 1.46</td>
<td>1.00 1.00 1.00 1.08 1.00</td>
<td>0.98 0.99 0.97 1.54 0.99</td>
</tr>
</tbody>
</table>

(a) Speed-up visualization with the multi-GPU mechanism.

Time consumption analysis of Full Chip ILT System.

(b) Time breakdown of complete flow.
Visualization of printed on-wafer image: large scale patterns and local boundary error areas.

(a) DAC’22 Yang, Li, et al. 2022

(b) Ours
Visualization of the via layer printed image. The figure compares via optimization result with the target image.
Conclusion
Main techniques:

- Recursively layout partitioning & Gradient stitching
- Distributed Optimization with pipeline
- Incremental re-optimization using healing boxes.

Main Contributions:

1. Solve the boundary stitching error in full-chip ILT.
2. Show the full-chip ILT result in a metal layer.
3. Present gradient stitching instead of mask stitching.
THANK YOU!