International Symposium on Physical Design

FuILT: Full Chip ILT System With Boundary Healing

Shuo Yin[1], Wenqian Zhao[1], Li Xie[2], Hong Chen[2], Yuzhe Ma[3], Tsung-Yi Ho[1], Bei Yu[1]

[1]Chinese University of Hong Kong
[2]Shenzhen GWX Technology Co., Ltd.
[3]Hong Kong University of Science and Technology (Guangzhou)

Outline

Motivation

Inverse Lithography Technology

ILT

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]

[1] Jhih-Rong Gao et al. (2014). "MOSAIC: Mask Optimizing Solution With Process Window Aware Inverse Correction". In: *Proc. DAC*, 52:1–52:6.

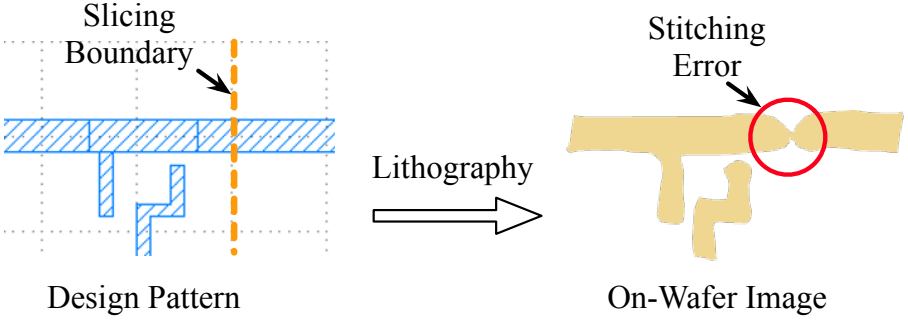
[2] Bentian Jiang et al. (2020). "Neural-ILT: Migrating ILT to neural networks for mask printability and complexity co-optimization". In: *Proc. ICCAD*, pp. 1–9.

[3] Ziyang Yu et al. (2022). "A GPU-enabled level-set method for mask optimization". In: *IEEE TCAD* 42.2, pp. 594–605.

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.

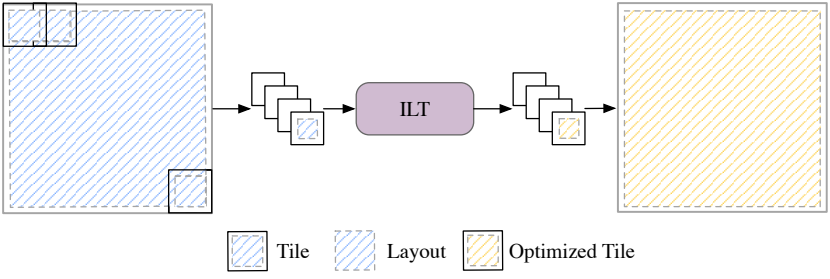
Lithography defects:



Visualization of lithography defects after mask stitching.

Previous Works

The meshing partition strategy of DAC'22[4], ASPDAC'23[5], TCAD'23[6].



However, the boundary error remains undiscussed.

[4]Haoyu Yang, Zongyi Li, et al. (2022). "Generic lithography modeling with dual-band optics-inspired neural networks". In: *Proc. DAC*, pp. 973–978.

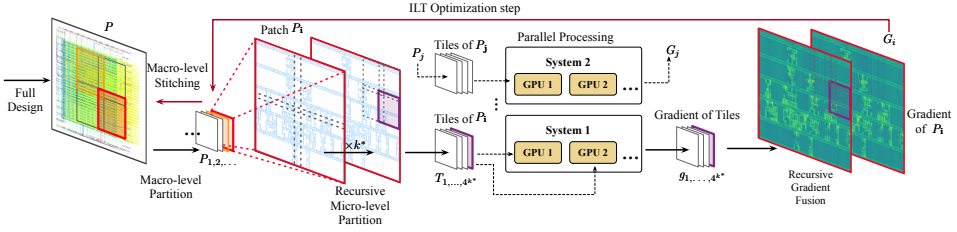
[5]Haoyu Yang and Haoxing Ren (2023). "Enabling scalable AI computational lithography with physics-inspired models". In: *Proc. ASPDAC*, pp. 715–720.

[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

Full Chip ILT System:



Full-chip ILT system overall workflow

- **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

Macro-level Partition

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

$$
\begin{aligned}
m^*, n^* = \underset{m,n}{\operatorname{argmin}} & \sum_{i}^{m \times n} (H_{\boldsymbol{P}_i} + W_{\boldsymbol{P}_i}), \\
\text{s.t.} \quad & \sum_{i}^{m} H_{\boldsymbol{P}_i} < H_{\boldsymbol{P}} + H'_{max} \\
& \sum_{i}^{n} W_{\boldsymbol{P}_i} < W_{\boldsymbol{P}} + W'_{max}, \\
& Size(\boldsymbol{P}_i) + Size(\boldsymbol{G}_i) \leq MemSize.
\end{aligned}
\tag{1}
$$

$\boldsymbol{P}_i$ represents the i-th patch and $\boldsymbol{G}_i$ denotes the gradient of i-th patch, H'_{max}/W'_{max} represent the max overlapping length, $H_{\boldsymbol{P}}/W_{\boldsymbol{P}}$ and $H_{\boldsymbol{P}_i}/W_{\boldsymbol{P}_i}$ denote height/width of the patch.

Micro-level Partition

At the micro-level, we start handling each patch while considering the computational capability of GPUs.

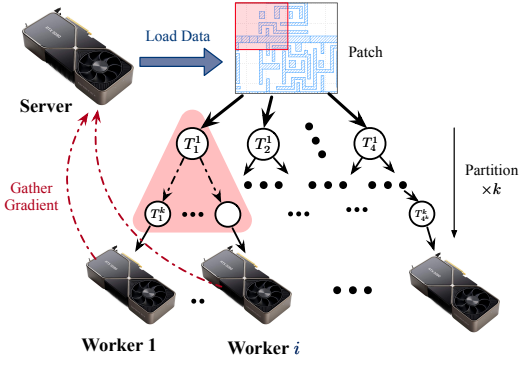
We slice each large patch into four tiles and repeat recursively until each tile satisfies the mentioned GPU computation limits.

$$\boldsymbol{T}_1, ..., \boldsymbol{T}_4 = \begin{bmatrix} \boldsymbol{T}_{1[s_h \times s_w]} & \boldsymbol{T}_{3[s_h \times s_w]} \\ \boldsymbol{T}_{2[s_h \times s_w]} & \boldsymbol{T}_{4[s_h \times s_w]} \end{bmatrix} \tag{2}$$

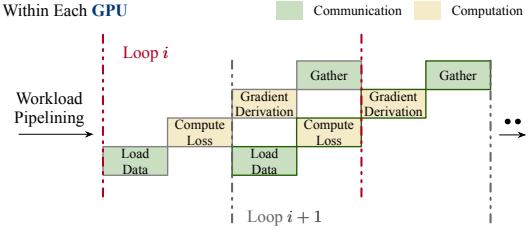
At each level, a tile is further sliced into "⊞" grid of four smaller tiles with overlapping rate R.

Distributed Optimization

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.



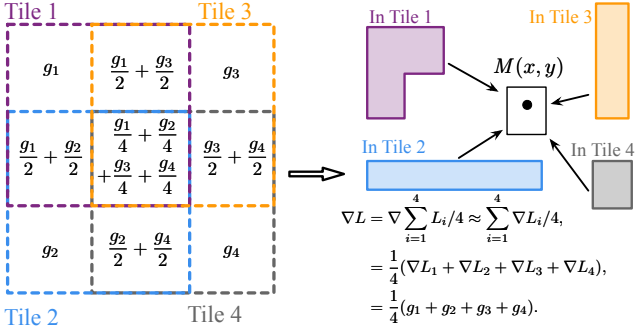
Server-Worker strategy



Multi-GPU Pipeline

Micro-level Stitching & Boundary Healing

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.

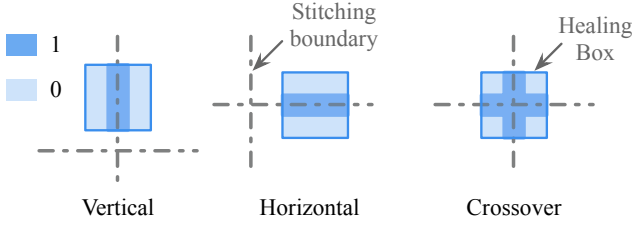


Gradient fusion on the overlapping area.

Macro-level Stitching & Boundary Healing

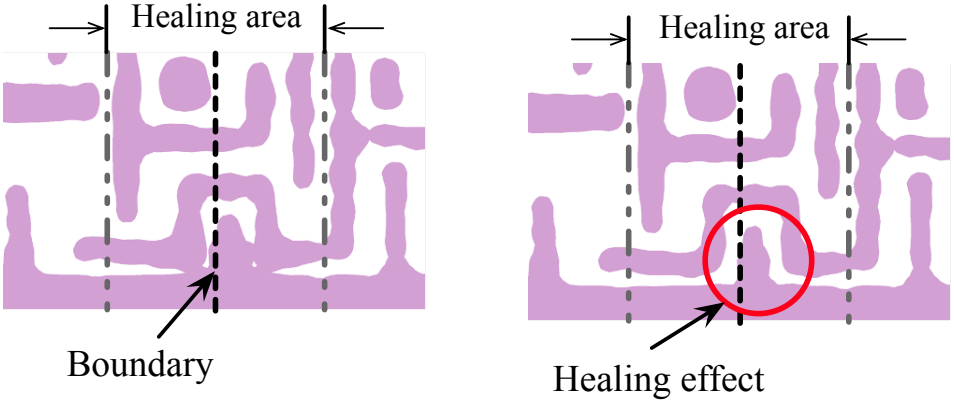
After obtaining the masks of large patches with inner boundaries fixed, we also need to tackle the stitching boundary of these large patches.

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



(a) (b)

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

Benchmark Details

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

Bench	#Polygons	Bounding Box ($nm \times nm$)	Total Area (nm^2)	Average Degree
Metal	5043	80528×80192	1675692925	5.8
Via	5411	73696×68992	22861474	4.0
Poly	1126	73696×68992	88904249	5.1
Pimplant	1830	80528×80192	3892569599	4.0

[7] *OpenROAD* (n.d.). https://github.com/The-OpenROAD-Project/OpenROAD.
[8] *FreePDK45* (n.d.). https://eda.ncsu.edu/freepdk/freepdk45/.

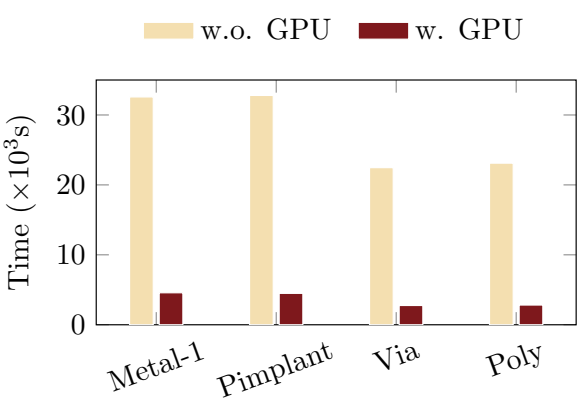
Result Comparison

Table: Comparisons of EPE, PVBand, L_2 Loss and Runtime

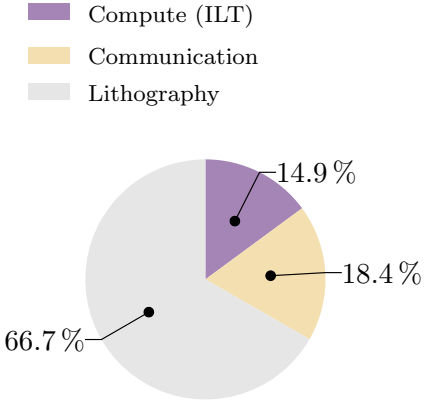
Bench	DAC'22[9] w/o. Macro Boundary Healing					FuILT w/o. Macro Boundary Healing					FuILT w. Macro Boundary Healing				
	#EPE	PVB (nm^2)	L_2 (nm^2)	RT (s)	S($\times 10^7$)	#EPE	PVB (nm^2)	L_2 (nm^2)	RT (s)	S($\times 10^7$)	#EPE	PVB (nm^2)	L_2 (nm^2)	RT (s)	S($\times 10^7$)
Metal-1	24668	120961048	224397927	2956	83.16	1243	66938717	158733917	3179	43.27	1221	66810974	154685729	4563	42.80
Via	127	6520380	9851354	1827	3.66	10	6445143	7931745	1902	3.38	10	6432513	7815421	2741	3.36
Poly	164	40238769	43742084	1831	20.55	59	32249748	31708247	1936	16.10	53	32055284	30297295	2836	15.88
Pimplant	4885	76507491	58102567	2837	38.86	1674	76389450	56955213	3174	37.09	1668	76310462	56664328	4475	37.03
Total	29844	244227688	336093932	9451	146.23	2986	182023058	255329122	10191	99.84	2952	181609233	249462773	14615	99.07
Ratio	9.99	1.34	1.31	**1.00**	1.46	**1.00**	**1.00**	**1.00**	1.08	**1.00**	0.98	0.99	0.97	1.54	0.99

[9] Haoyu Yang, Zongyi Li, et al. (2022). "Generic lithography modeling with dual-band optics-inspired neural networks". In: *Proc. DAC*, pp. 973–978.

Runtime Breakdown



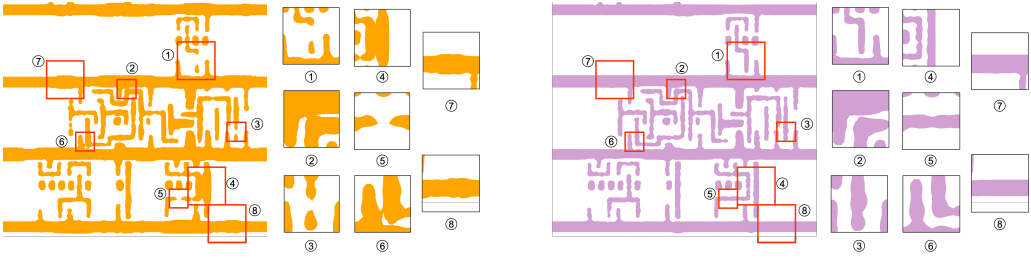
(a) Speed-up visualization with the multi-GPU mechanism.



(b) Time breakdown of complete flow.

Time consumption analysis of Full Chip ILT System.

Visualization Result

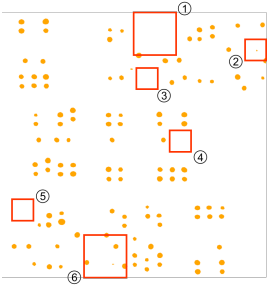


(a) DAC'22Yang, Li, et al. 2022

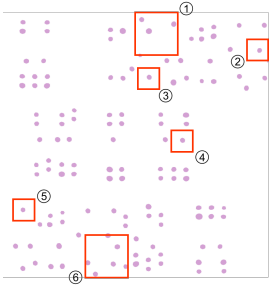
(b) Ours

Visualization of printed on-wafer image: large scale patterns and local boundary error areas.

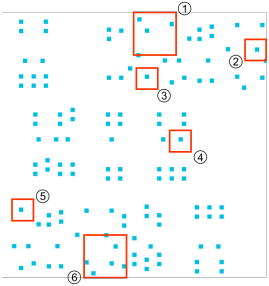
Visualization Result



(a) DAC'22 Yang, Li, et al. 2022



(b) Ours



(c) Target Image

Visualization of the `Via` layer printed image. The figure compares via optimization result with the target image.

Conclusion

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!