2023 IEEE 15th International Conference on ASIC

OpenILT: An Open Source Inverse Lithography Technique Framework

Su Zheng, Bei Yu, Martin Wong

Department of Computer Science & Engineering
The Chinese University of Hong Kong

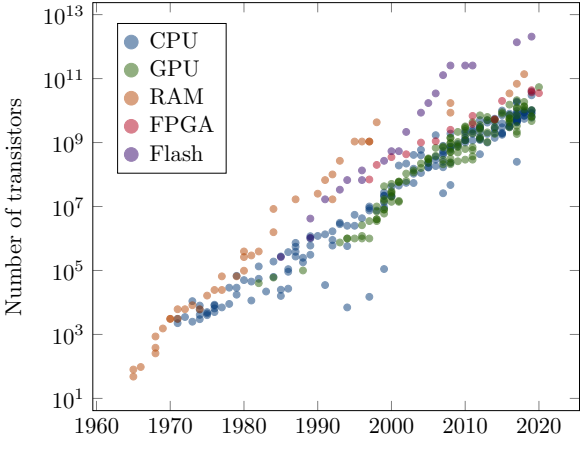
香港中文大學
The Chinese University of Hong Kong

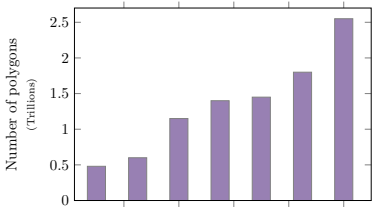
Outline

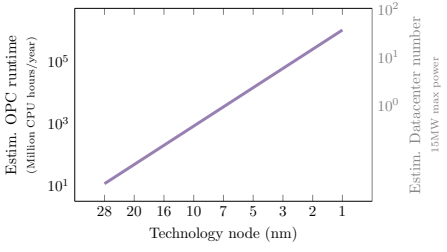
Introduction

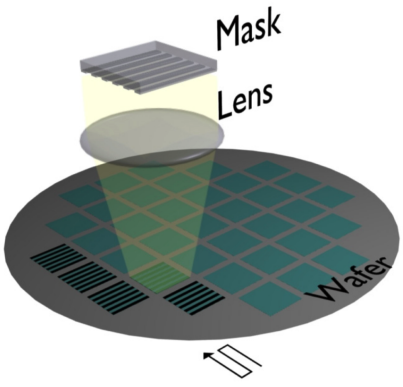
Moore's Law to Extreme Scaling

- Billions of transistors on a chip → ... Trillions of polygons

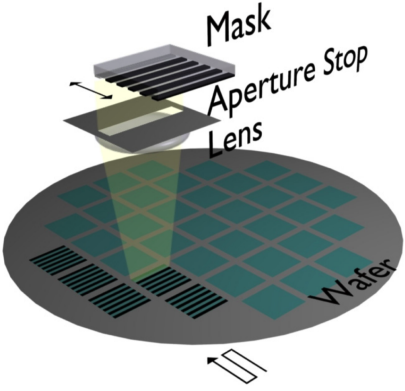






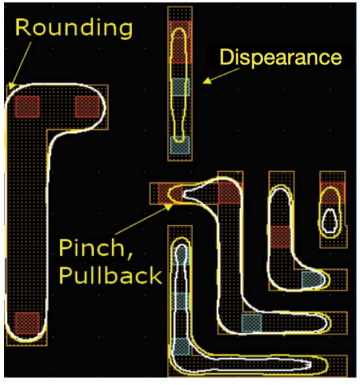


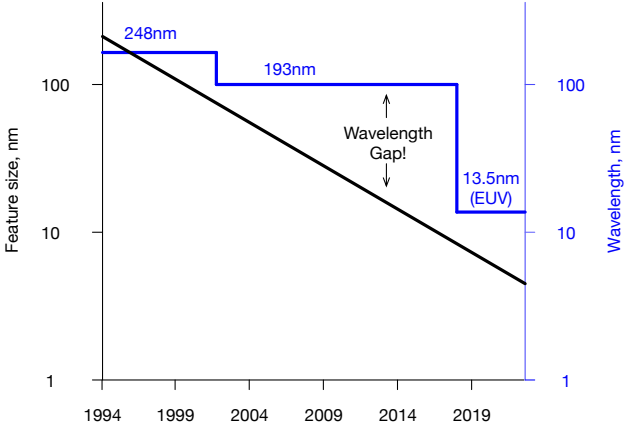
(a)



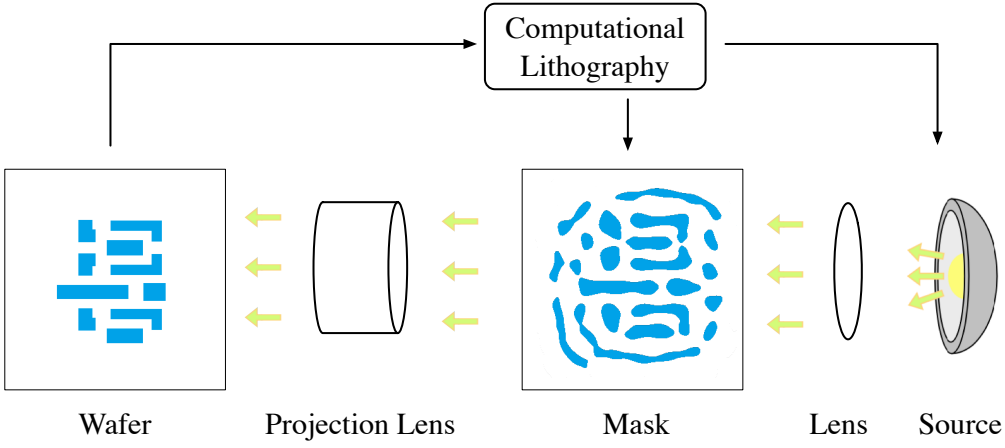
(b)

When feature is small: what you see ≠ what you want

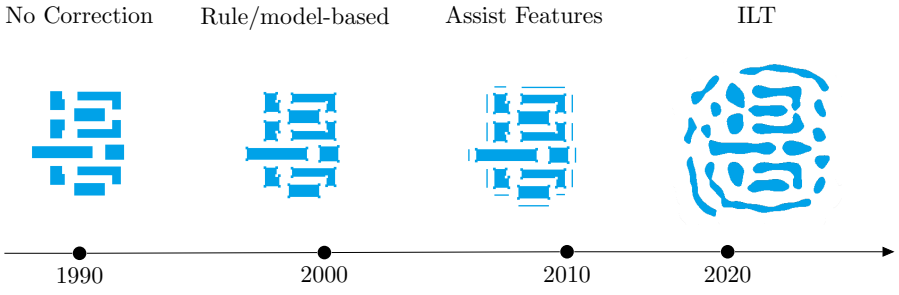




Computational Lithography



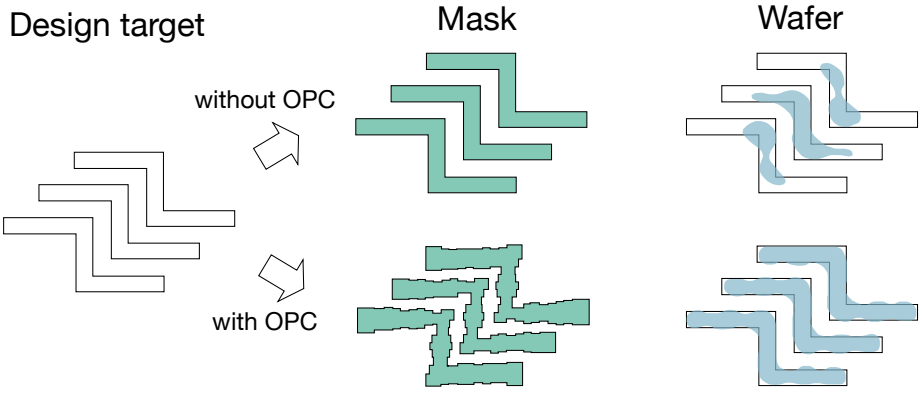
Timeline



- From simple distortion to complex modification
- Computation becomes more and more complicated

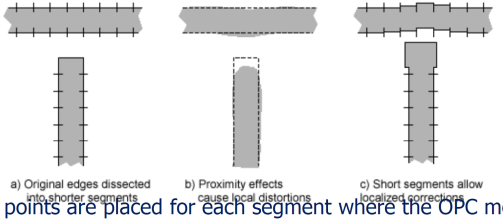
Related Work

Mask Optimization

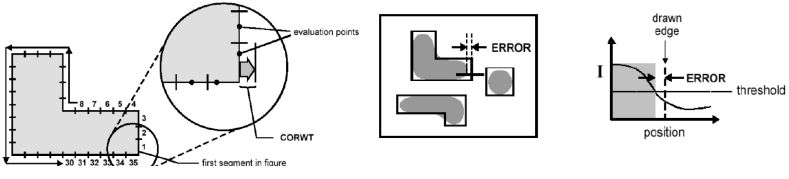


How does OPC work?

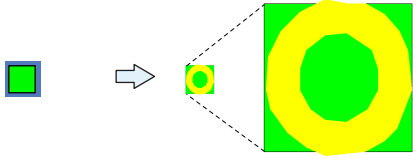
- The edges of the layout are dissected in shorter segments and features are classified into run segments, inner corners, outer corners, etc.



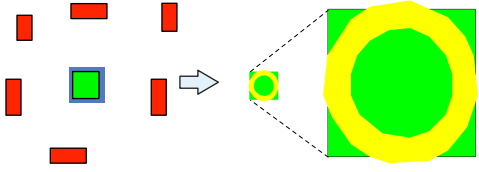
- Evaluation points are placed for each segment where the OPC model is evaluated and an "edge placement error" is calculated



- Sub-Resolution Assisted Feature
- Improve the robustness of the target patterns without printing themselves.



With OPC only



With SRAF and OPC

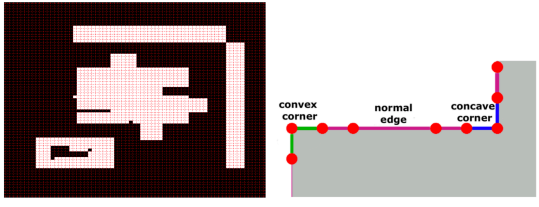
Classic OPC

- Model/Rule-based OPC [Cobb+,SPIE'02][Kuang+,DATE'15] [Awad+,DAC'16] [Su+,ICCAD'16]
  1. Fragmentation of shape edges;
  2. Move fragments for better printability.
- Inverse Lithography [Pang+,SPIE'05] [Gao+,DAC'14] [Poonawala+,TIP'07] [Ma+,ICCAD'17]
  1. Efficient model that maps mask to aerial image;
  2. Continuously update mask through descending the gradient of contour error.

Machine Learning OPC

[Matsunawa+,JM3'16] [Choi+,SPIE'16] [Xu+,ISPD'16] [Shim+,APCCAS'16]

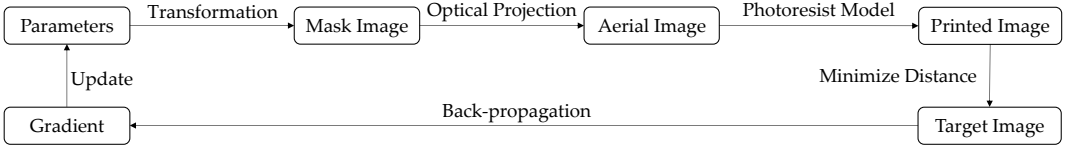
1. Edge fragmentation;
2. Feature extraction;
3. Model training.



Inverse Lithography Technique (ILT)

The main objective in ILT is minimizing the lithography error via gradient descent.

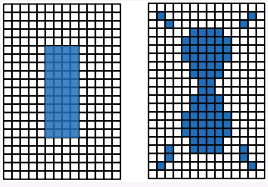
$$E = ||\mathbf{Z}_t - \mathbf{Z}||_2^2, \tag{1}$$



ILT Methodologies [DAC'14][1] [ICCAD'17][2] [DATE'21][3]

Typical ILT

- Mask → Image → Matrix
- Calculate gradient on each pixel.



Level-set method

- Boundary-based update
- Implicit representation; focus on boundaries

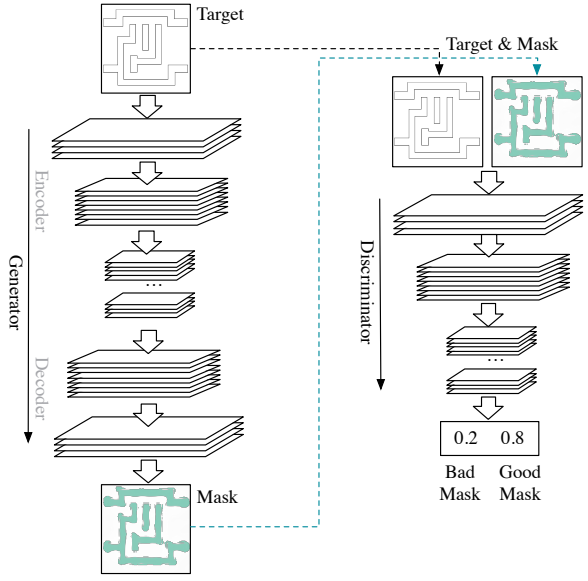
$$\begin{cases} \phi(t,\boldsymbol{x}) < 0 & \text{if } \boldsymbol{x} \in \Omega(t) \\ \phi(t,\boldsymbol{x}) = 0 & \text{if } \boldsymbol{x} \in \Gamma(t) \\ \phi(t,\boldsymbol{x}) > 0 & \text{if } \boldsymbol{x} \in \overline{\Omega(t)} \end{cases}$$

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

[2] Yuzhe Ma et al. (2017). "A Unified Framework for Simultaneous Layout Decomposition and Mask Optimization". In: *Proc. ICCAD*, pp. 81–88.

[3] Ziyang Yu et al. (2021). "A GPU-enabled Level Set Method for Mask Optimization". In: *Proc. DATE*.

GAN-OPC [DAC'18][4]



[4]Haoyu Yang et al. (2018). "GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets". In: *Proc. DAC*, 131:1–131:6.

github.com/OpenOPC/OpenILT/

README.md

```
   ____                   ____ __   _____
  / __ \ ____   ___   ____    /  _// /  /_  __/
 / / / // __ \ / _ \ / __ \   / / / /    / /
/ /_/ // /_/ //  __// / / / _/ / / /___ / /
\____// .___/ \___//_/ /_/ /___//_____//_/
     /_/
```

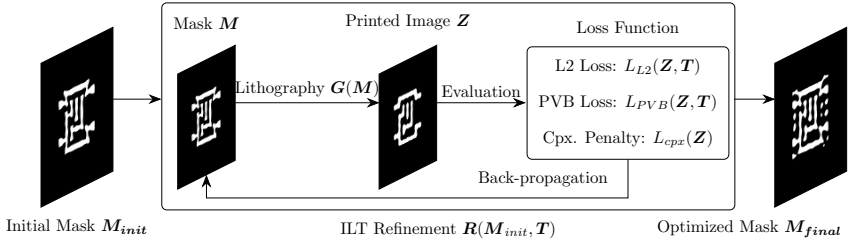
OpenILT: An Open-source Platform for Inverse Lithography Technology Research

OpenILT is a open-source platform for inverse lithography technology (ILT) research. It has a comprehensive and flexible ecosystem of libraries that enable the efficient development and evaluation of ILT algorithm. OpenILT decouples the ILT flow into different components, lithography simulation, initialization, optimization, and evaluation. ILT researchers can implement and evaluate their ideas quickly by replacing a component with the novel method. Moreover, the platform is implemented with *pytorch*, which enables easy GPU acceleration and deep-learning integration.

OpenILT Platform

OpenILT Flow

- Lithography simulation
- Initialization
- Solver
- Evaluation



Lithography simulation

- Lithography simulation = Optical Projection + Photoresist models
- Optical Projection: Hopkins' Model + FFT Acceleration

$$\boldsymbol{I}(x,y) = \sum_{k=1}^{N_h} w_k \left| \boldsymbol{M}(x,y) \otimes \boldsymbol{h}_k(x,y) \right|^2, \tag{2}$$

$$\boldsymbol{h}_k \otimes \boldsymbol{M} = \texttt{IFFT}(\texttt{FFT}(\boldsymbol{h}_k) \odot \texttt{FFT}(\boldsymbol{M})). \tag{3}$$

- Photoresist Model: Sigmoid Function

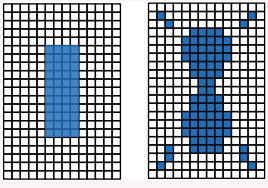
$$\boldsymbol{Z}(x,y) = \sigma_Z(\boldsymbol{I}(x,y)) = \frac{1}{1 + \exp\left(-\theta_Z(\boldsymbol{I}(x,y) - I_{th})\right)}. \tag{4}$$

- The lithography simulation models are implemented as PyTorch modules in OpenILT, enabling auto-gradient

Initialization

Pixel-based method

- Gray-scale initialization



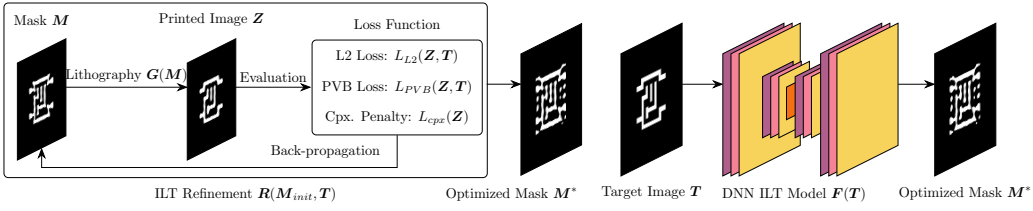
Level-set method

- Boundary-based initialization

$$\begin{cases} \phi(t,\boldsymbol{x}) < 0 & \text{if } \boldsymbol{x} \in \Omega(t) \\ \phi(t,\boldsymbol{x}) = 0 & \text{if } \boldsymbol{x} \in \Gamma(t) \\ \phi(t,\boldsymbol{x}) > 0 & \text{if } \boldsymbol{x} \in \overline{\Omega(t)} \end{cases}$$

Solver

- MOSAIC: Basic pixel-based method
- LevelSet: Basic levelset-based method
- MultiLevel: State-of-the-art ILT method
- GAN-OPC: Deep-learning-driven ILT method



Overview of (a) traditional ILT and (b) DNN-based ILT.

Evaluation

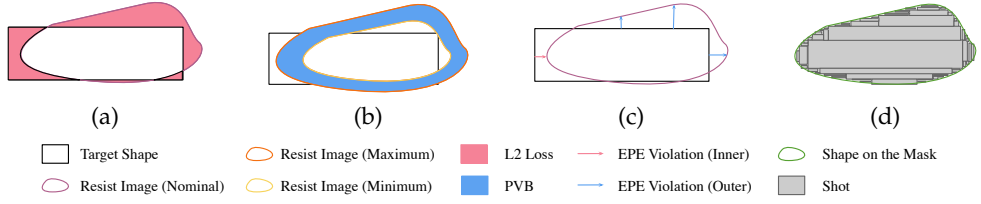


Illustration of the metrics. (a) *L2* measures the difference between the printed and target images. (b) *PVB* quantifies the maximum discrepancy between process corners. (c) *EPE* estimates the distortion of the printed image. (d) #*Shots* counts the rectangles needed to construct the mask.

Code Example

- Easy Implementation of ILT Methods

```
cfg, litho = SimpleCfg(), LithoSim()
solver = SimpleILT(cfg, litho)
design = Design("M1_test1.glp")
Zt,P = PixelInit().run(design)
l2,pvb,P,M = solver.solve(Zt,P)
l2,pvb,epe,shot = evaluate(M,Zt,litho)
```

(a) Target Image

(b) LevelSet

(c) MultiLevel

Experiments

Table: Comparison Between Reproduced and Original Methods

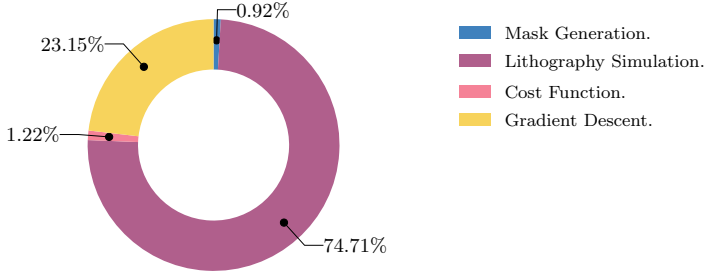
Benchmarks	MOSAIC				Our MOSAIC				LevelSet				Our LevelSet			
	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)
case1	6	49893	65534	318	8	48896	55028	0.95	4	46032	62693	123	6	45520	57468	2.49
case2	10	50369	48230	256	4	37327	46019	0.95	1	36177	50724	81	1	33571	49680	2.27
case3	59	81007	108608	321	47	81327	86685	0.94	29	71178	100945	214	39	78695	90748	2.26
case4	1	20044	28285	322	2	16409	26358	0.94	0	16345	29831	184	2	18040	27710	2.27
case5	6	44656	58835	315	0	37810	57472	0.94	1	47103	56510	76	2	38226	59035	2.26
case6	1	57375	48739	314	0	36706	52566	0.94	1	46205	51204	65	0	35962	54163	2.27
case7	0	37221	43490	239	2	29520	47598	0.94	0	28609	45056	64	2	30542	48173	2.27
case8	2	19782	22846	258	1	14291	24268	0.94	1	19477	22757	67	1	14252	25043	2.26
case9	6	55399	66331	322	2	47367	64932	0.94	0	52613	64597	63	1	43390	68229	2.26
case10	0	24381	18097	231	0	8950	19871	0.94	0	22415	18769	64	0	8919	20878	2.27
Average	9.1	44012	50899	289	**6.6**	**35860**	**48080**	**0.94**	**3.7**	38615	50309	100	5.4	**34712**	**50113**	**2.29**

Table: Comparison Between Reproduced and Original Methods

Benchmarks	GAN-OPC				Our GAN-OPC				MultiLevel				Our MultiLevel			
	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)	EPE	L_2 (nm^2)	PVB (nm^2)	Time (s)
case1	-	55425	58043	-	20	58712	52126	1.13	3	39303	46077	1.42	4	38577	47367	1.03
case2	-	40211	53020	-	1	36669	43861	1.13	0	28986	37626	1.24	1	32104	37572	1.03
case3	-	93090	75644	-	51	85677	68400	1.12	22	66151	68021	1.42	20	64245	72910	1.03
case4	-	22877	26401	-	1	15812	27559	1.13	0	15890	23511	0.72	0	10880	23270	1.03
case5	-	42650	59765	-	5	46249	55090	1.13	0	29138	49987	1.43	0	30454	51915	1.03
case6	-	39776	54878	-	0	37489	51545	1.13	0	30558	44503	1.42	0	30504	46394	1.03
case7	-	22761	49156	-	0	26882	45715	1.14	0	15765	37009	1.43	0	16056	39412	1.03
case8	-	16296	24441	-	0	14654	24076	1.13	0	13943	21503	0.8	0	11560	19991	1.03
case9	-	52157	66492	-	5	51179	61939	1.14	0	36397	55600	1.43	0	36017	58943	1.03
case10	-	9765	21338	-	0	9066	20121	1.12	0	7492	16604	1.42	0	8533	15942	1.03
Average	-	39501	48918	-	**8.3**	**38239**	**45043**	**1.13**	**2.5**	28362	**40044**	1.27	**2.5**	**27893**	41372	**1.03**

Runtime Analysis

- Take MOSAIC as an example



THANK YOU!