香 港 中 文 大 學
The Chinese University of Hong Kong

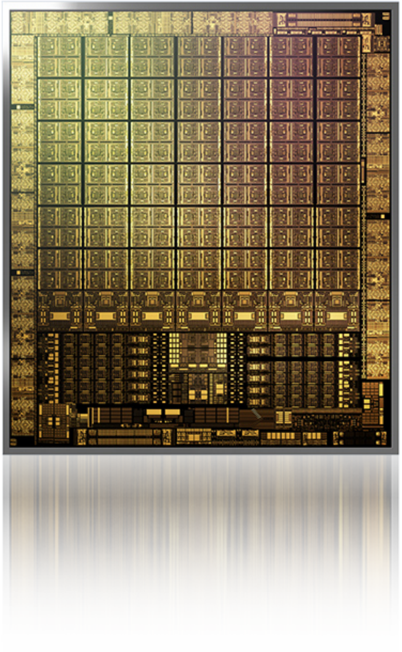
Large Scale VLSI Mask Optimization

Bei Yu
Chinese University of Hong Kong
`byu@cse.cuhk.edu.hk`

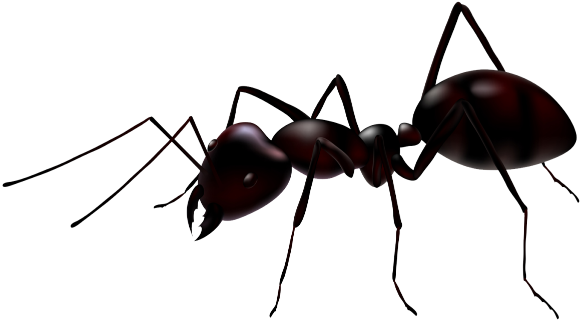
July 30, 2023

1×: Your Laptop

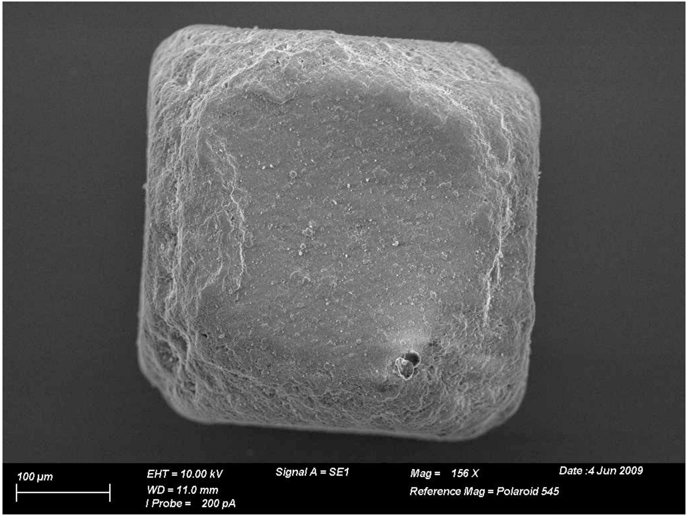
15×: NVIDIA Ampere



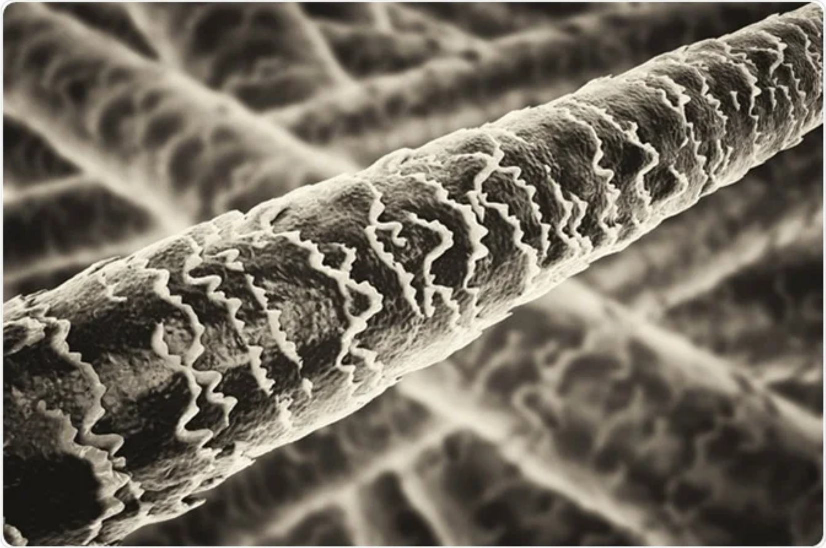
100×: An Ant



1000×: Grain of Salt

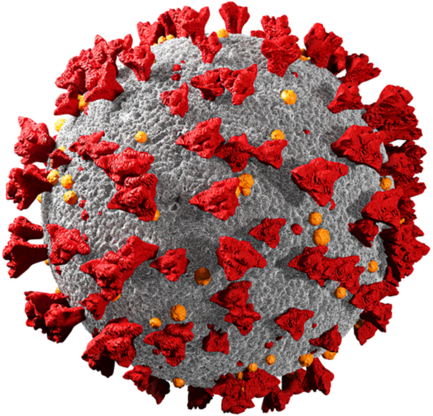


5000×: Human Hair

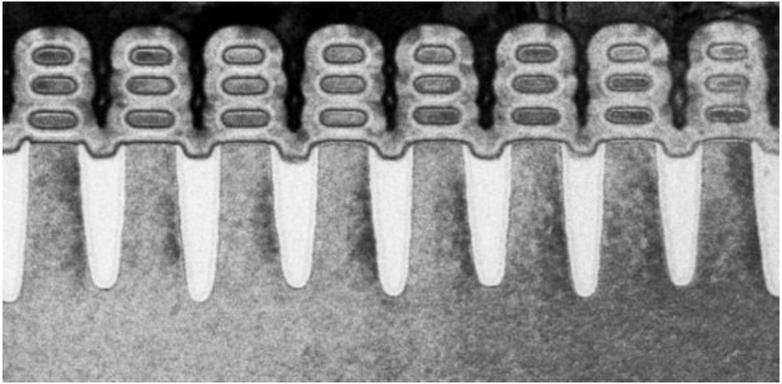


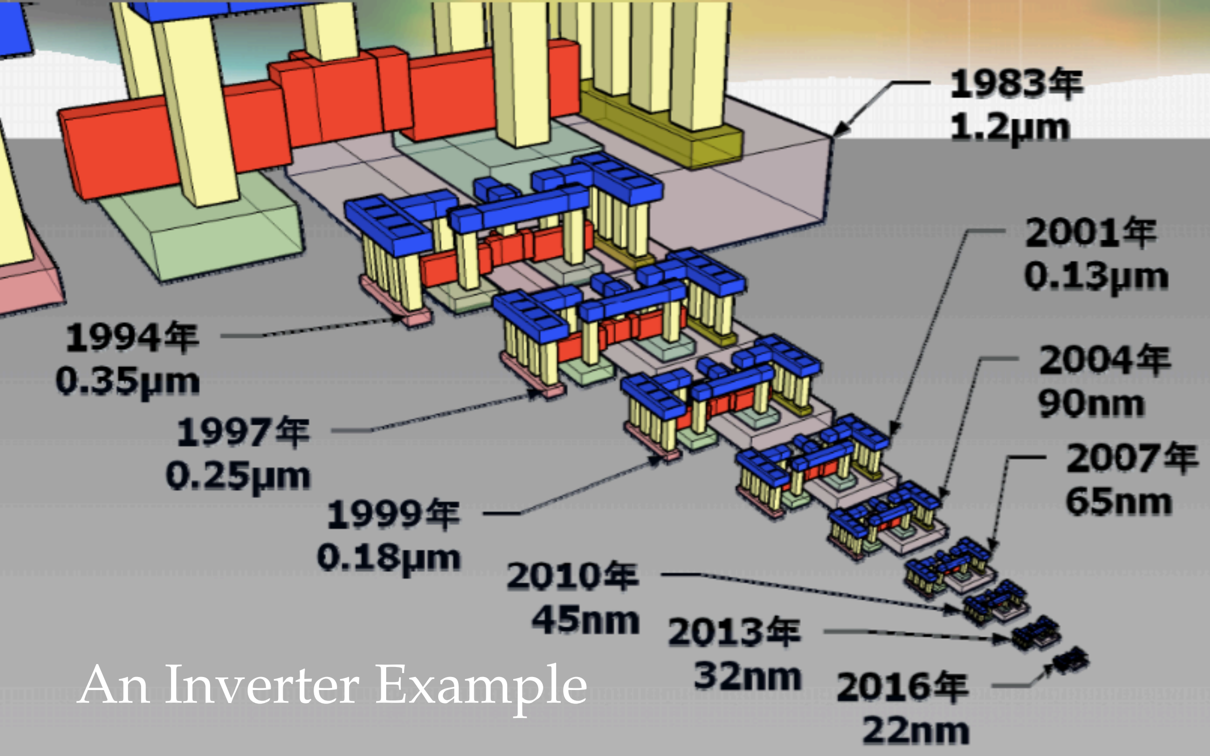
50,000×: Red Blood Cell

5,000,000×: Coronavirus



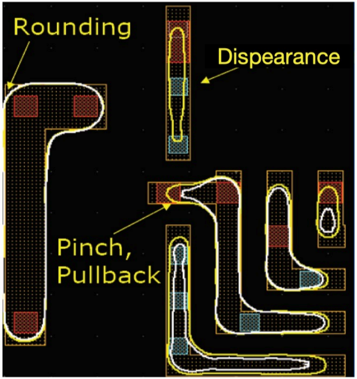
50,000,000×: 5nm Transistor

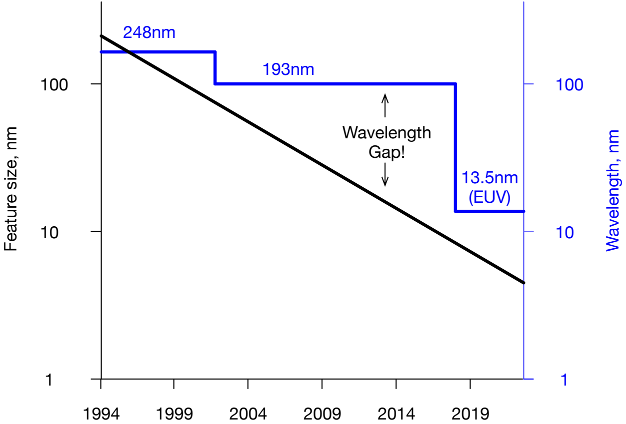




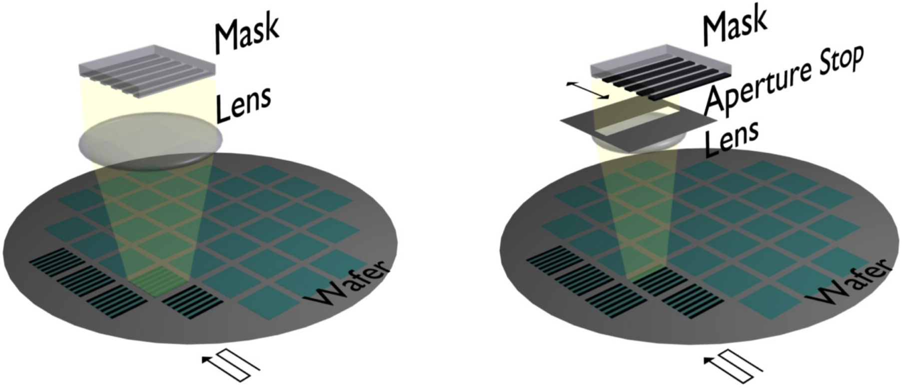
An Inverter Example

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





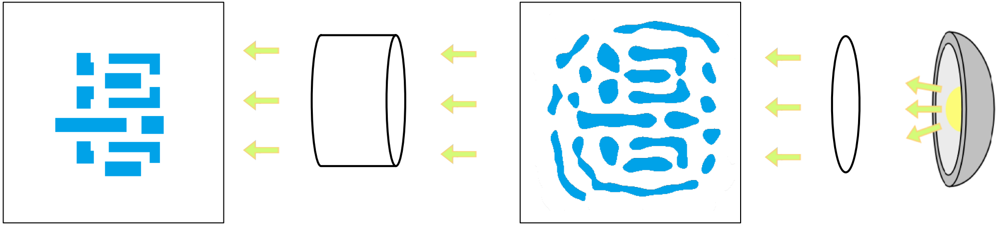
Exposure



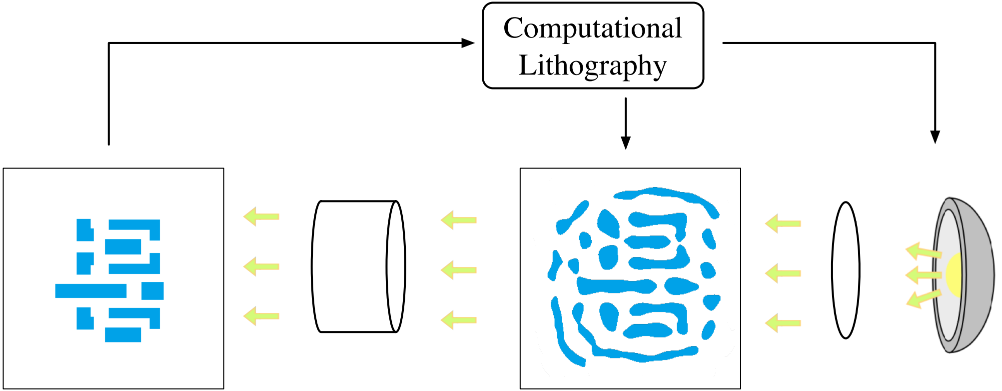
(a) (b)

Figure 2.3: Exposure procedure of state-of-the-art projection tools: (a) Step-and-repeat: the entire die is exposed in one step; (b) Step-and-scan: the die is gradually exposed by a scanning slit.

Computational Lithography

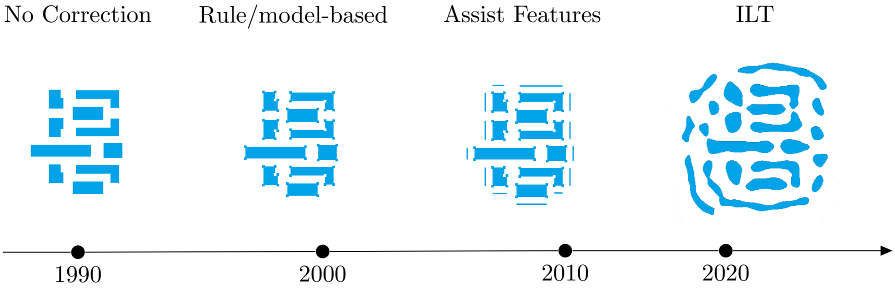


Computational Lithography


Computational Lithography

Wafer Projection Lens Mask Lens Source

Timeline



No Correction

Rule/model-based

Assist Features

ILT

1990

2000

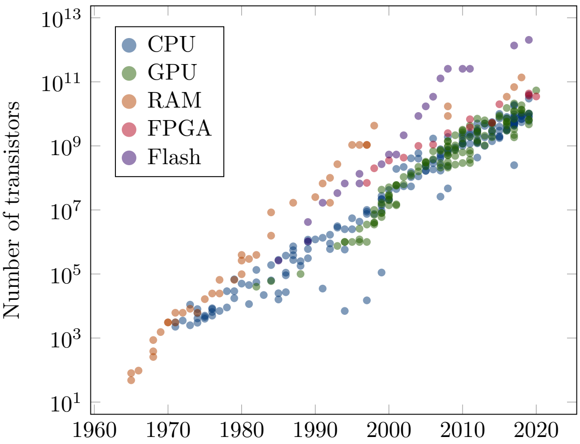
2010

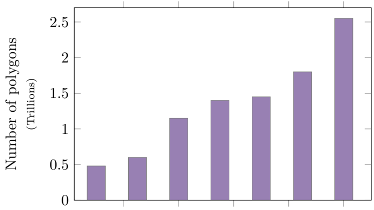
2020

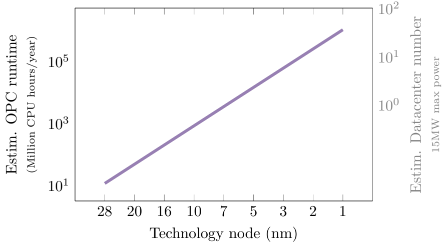
- From simple distortion to complex modification
- Computation becomes more and more complicated

Moore’s Law to Extreme Scaling

- Billions of transistors on a chip → ... Trillions of polygons
- OPC runtime goes to 10^6 (million) CPU hours/year







Outline

1. Lithography Modeling

2. Optical proximity correction (OPC)

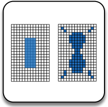
3. Full-chip OPC

Lithography Modeling

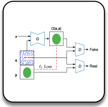
Lithography Modeling

Lithography Modeling

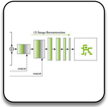
MOSAIC
ICCAD13



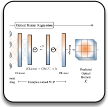
LithoGAN
DAC19

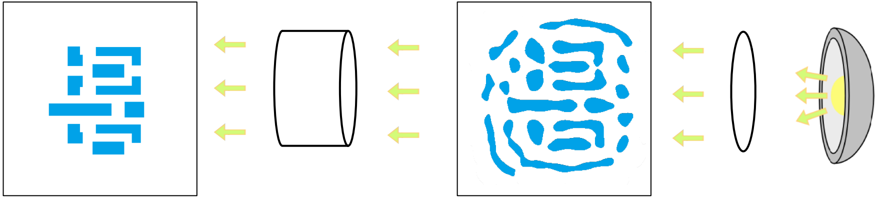


DOINN
DAC22



Nitho
DAC23





Wafer

Projection Lens

Mask

Lens

Source

Recap on rigorous lithography model

Hopkins Model and Transmission Cross-Coefficient (TCC)

The imaging equation:

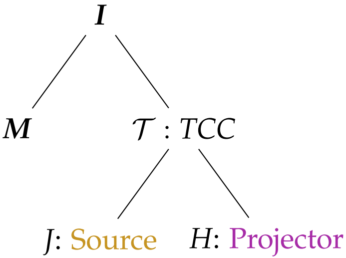
$$\mathcal{F}(\boldsymbol{I})(f,g) = \iint_{-\infty}^{\infty} \underline{\mathcal{T}\left((f'+f,g'+g),(f',g')\right)}\mathcal{F}(\boldsymbol{M})\left(f'+f,g'+g\right)\mathcal{F}(\boldsymbol{M})^{*}\left(f',g'\right)\mathrm{d}f'\mathrm{d}g', \quad (1)$$

where $\boldsymbol{M}$ is the mask, (f,g) is its frequencies. $\mathcal{T}$ is TCC given by:

$$\mathcal{T}\left((f',g'),(f'',g'')\right) := \iint_{-\infty}^{\infty} \underline{\mathcal{F}(J)(f,g)}\mathcal{F}(H)\left(f+f',g+g'\right)\mathcal{F}(H)^{*}\left(f+f'',g+g''\right)\mathrm{d}f\,\mathrm{d}g, \quad (2)$$

where the weight factor J solely depends on effective source, H is projector transfer function.

Computation graph of aerial image



$\boldsymbol{I}$

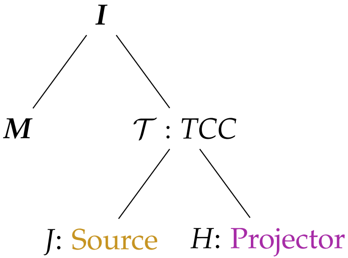
$\boldsymbol{M}$

$\mathcal{T}: TCC$

J: Source

H: Projector

Computation graph of aerial image



$\boldsymbol{I}$

$\boldsymbol{M}$

$\mathcal{T}$: TCC

J: Source

H: Projector

When the projector and source are fixed,

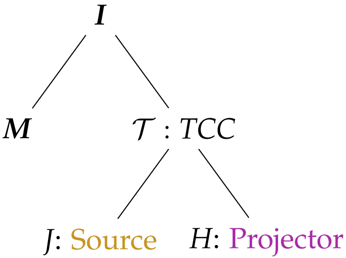
J: constant matrix

H: constant matrix

$\Downarrow$

$\mathcal{T}$: TCC is a constant matrix

Computation graph of aerial image



$\boldsymbol{I}$

$\boldsymbol{M}$

$\mathcal{T}: TCC$

J: Source

H: Projector

When the projector and source are fixed,

J: constant matrix

H: constant matrix

$\Downarrow$

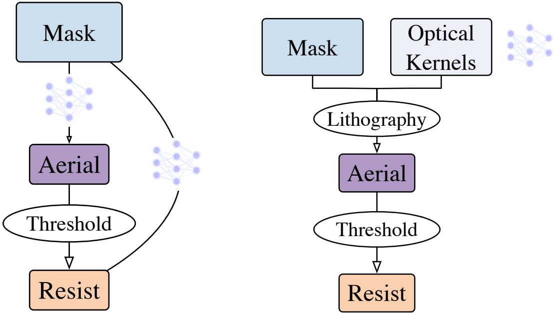
$\mathcal{T}$: TCC is a constant matrix

Instead of learning an
image-to-image mapping,

Would it be Possible to
learn the **TCC optical kernels**?

The benefits of learning optical kernels

- Get rid of negative influence of layer types & dataset distribution.
- Less training data required & smaller model size.

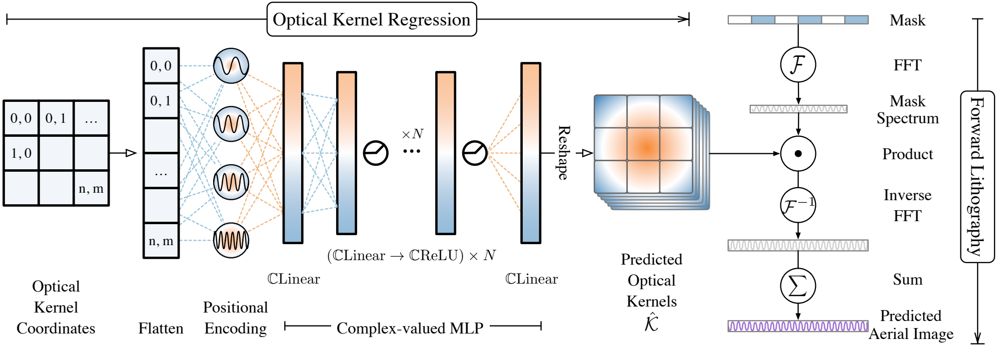


(c) Previous (d) Ours

Lithography modeling: Nitho [DAC'23][1]

Nitho

Physics-Informed Optical Kernel Regression Using Complex-valued Neural Fields



The overall aerial image prediction pipeline of Nitho framework, which separates mask-related linear operations from optical kernel regression using coordinate-based ℂMLP.

---

[1] Guojin Chen, Zehua Pei, et al. (2023). "Physics-Informed Optical Kernel Regression Using Complex-valued Neural Fields". In: *Proc. DAC*.

Lithography Model [DAC'14][2]

- SVD Approximation of Partial Coherent System [Cobb,1998]

$$\mathbf{I} = \sum_{k=1}^{N^2} w_k |\mathbf{M} \otimes \mathbf{h}_k|^2. \tag{3}$$

- Reduced Model [Gao+,DAC'14]

$$\mathbf{I} = \sum_{k=1}^{N_h} w_k |\mathbf{M} \otimes \mathbf{h}_k|^2. \tag{4}$$

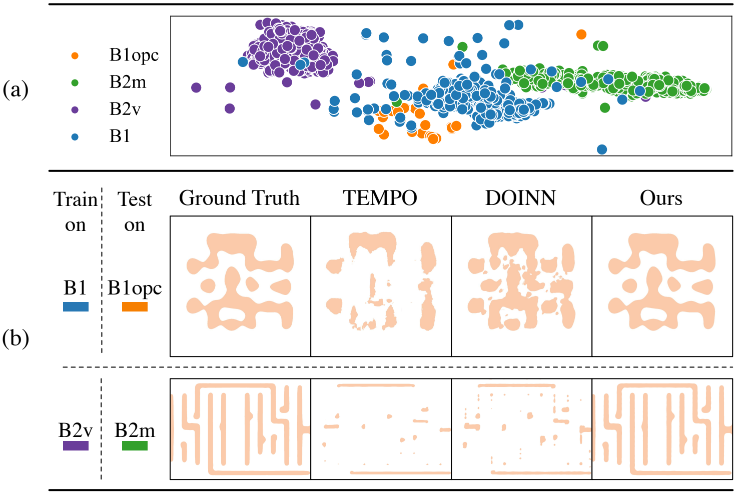
- Etch Model

$$\mathbf{Z}(x,y) = \begin{cases} 1, & \text{if } \mathbf{I}(x,y) \geq I_{th}, \\ 0, & \text{if } \mathbf{I}(x,y) < I_{th}. \end{cases} \tag{5}$$

---

[2] 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.

Nitho: Generalization on Out-of-distribution (OOD) data.



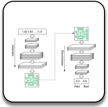
(a) t-SNE distribution of datasets; (b) Comparison of generalization capability on out-of-distribution (OOD) datasets.

Optical proximity correction (OPC)

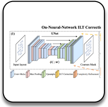
Optical proximity correction (OPC)

OPC

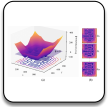
GAN-OPC
DAC18



NeuralILT
ICCAD20

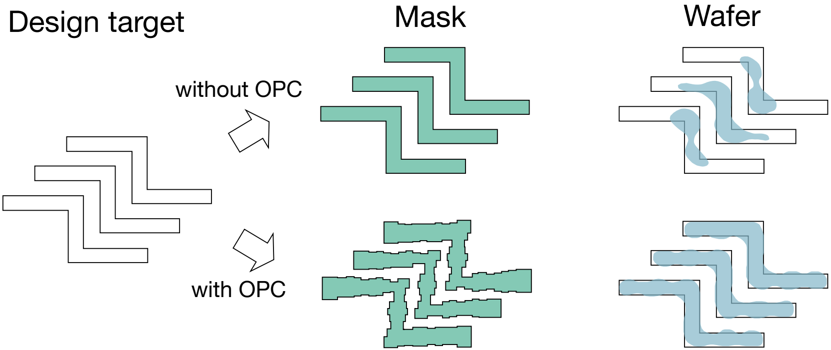


DevelSet
ICCAD21



MultiILT
DAC23

OpenILT
Now



Design target

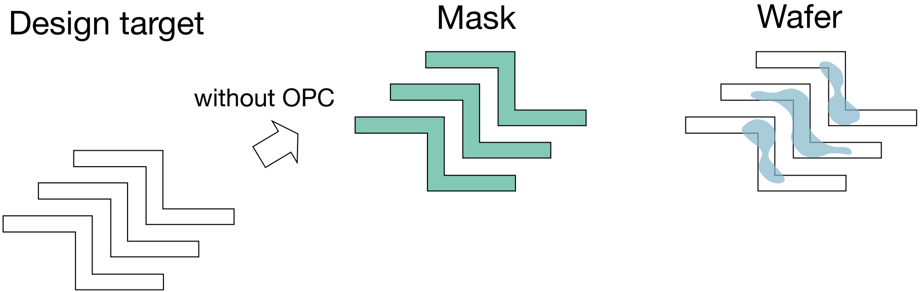
without OPC

with OPC

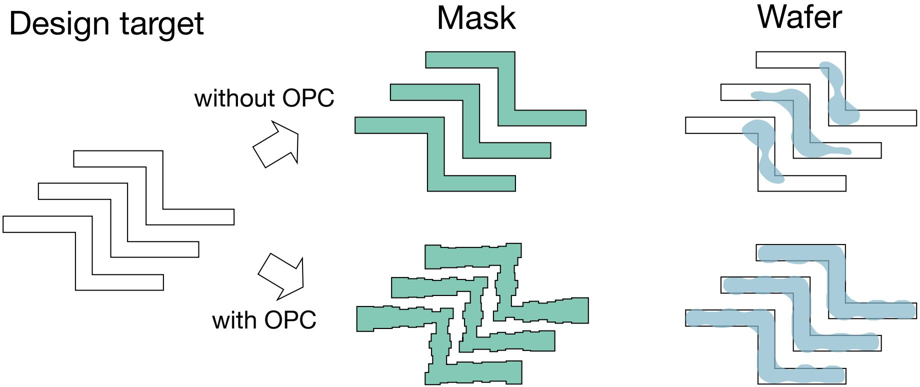
Mask

Wafer

Mask Optimization



Mask Optimization



Inverse Lithography Technique (ILT)

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

$$E = ||\mathbf{Z}_t - \mathbf{Z}||_2^2, \tag{6}$$

where $\mathbf{Z}_t$ is the target and $\mathbf{Z}$ is the wafer image of a given mask.
Apply translated sigmoid functions to make the pixel values close to either 0 or 1.

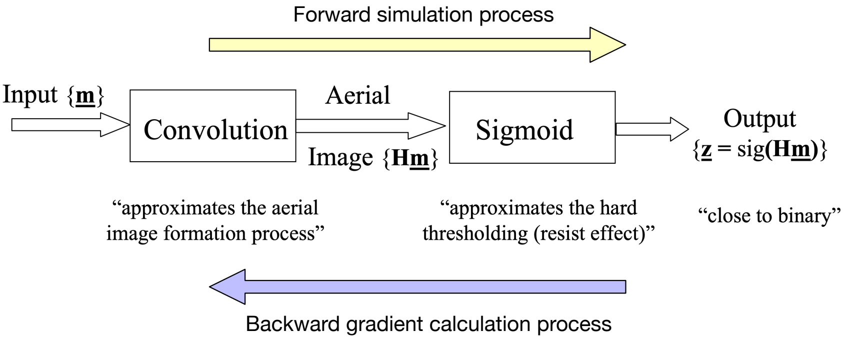
$$\mathbf{Z} = \frac{1}{1 + \exp[-\alpha \times (\mathbf{I} - \mathbf{I}_{th})]}, \tag{7}$$

$$\mathbf{M}_b = \frac{1}{1 + \exp(-\beta \times \mathbf{M})}. \tag{8}$$

Combine Equations (3)–(8) and the analysis in [Poonawala,TIP'07],

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{M}} =& 2\alpha\beta \times \mathbf{M}_b \odot (1 - \mathbf{M}_b) \odot \\ & (((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H}^*)) \otimes \mathbf{H} + \\ & ((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H})) \otimes \mathbf{H}^*).\end{aligned} \tag{9}$$

ILT flow



Forward simulation process

Input $\{\underline{\mathbf{m}}\}$

Convolution

Aerial Image $\{\mathbf{H}\underline{\mathbf{m}}\}$

Sigmoid

Output $\{\underline{\mathbf{z}} = \mathrm{sig}(\mathbf{H}\underline{\mathbf{m}})\}$

"approximates the aerial image formation process"

"approximates the hard thresholding (resist effect)"

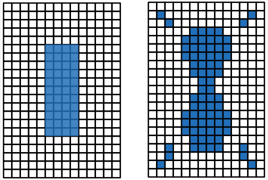
"close to binary"

Backward gradient calculation process

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.

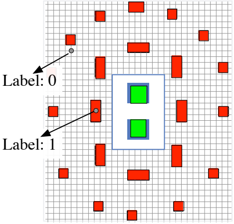
[2] Yuzhe Ma et al. (2017). "A Unified Framework for Simultaneous Layout Decomposition and Mask Optimization". In: *Proc. ICCAD*, pp. 81–88.

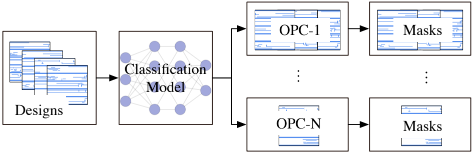
[3] Ziyang Yu et al. (2021). "A GPU-enabled Level Set Method for Mask Optimization". In: *Proc. DATE*.

Leaning-based Mask Optimization

Discriminative models [TCAD’20][4] [ASPDAC’20][5]

- Pixel-wise classification
- Printed image estimation/quality estimation



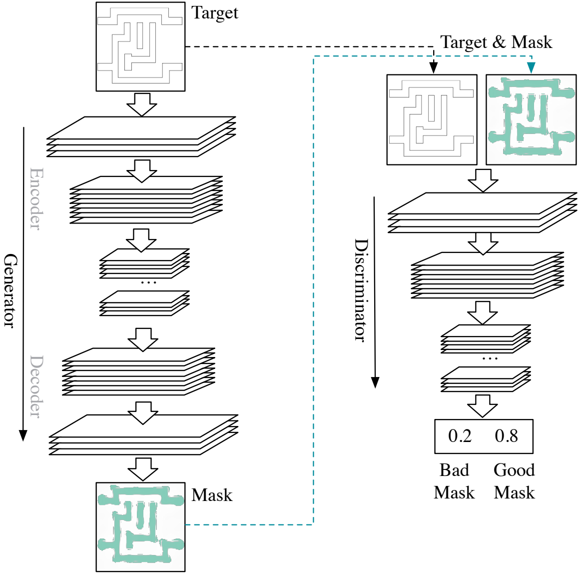


---

[4] Hao Geng et al. (2020). “SRAF Insertion via Supervised Dictionary Learning”. In: *IEEE TCAD*.

[5] Haoyu Yang, Wei Zhong, et al. (2020). “VLSI Mask Optimization: From Shallow To Deep Learning”. In: *Proc. ASPDAC*, pp. 434–439.

GAN-OPC [DAC'18][6]

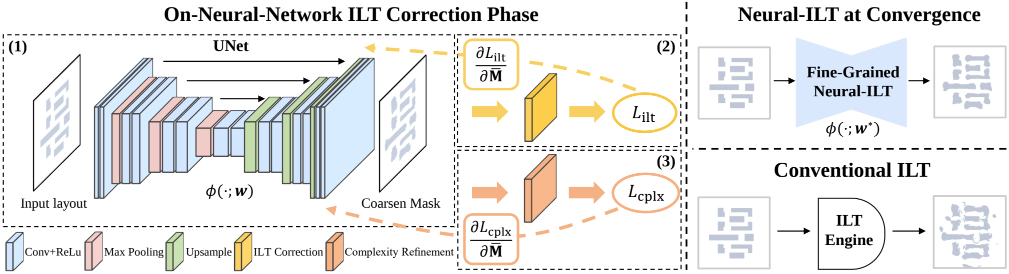


---

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

Neural-ILT [ICCAD’20][7]

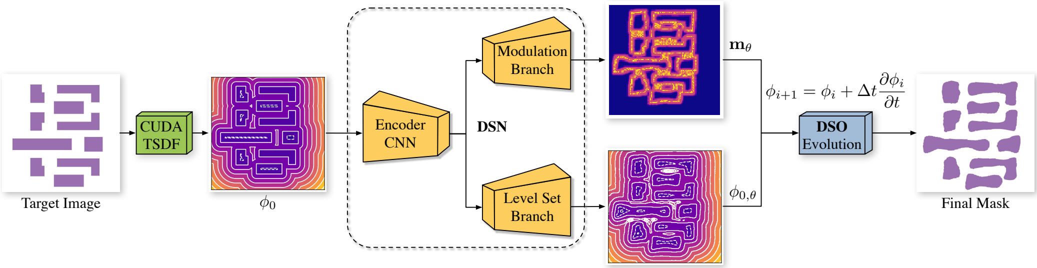
- A pre-trained UNet for performing layout-to-mask translation.
- An ILT correction layer for minimizing inverse lithography loss.
- A mask complexity refinement layer for removing redundant complex features.

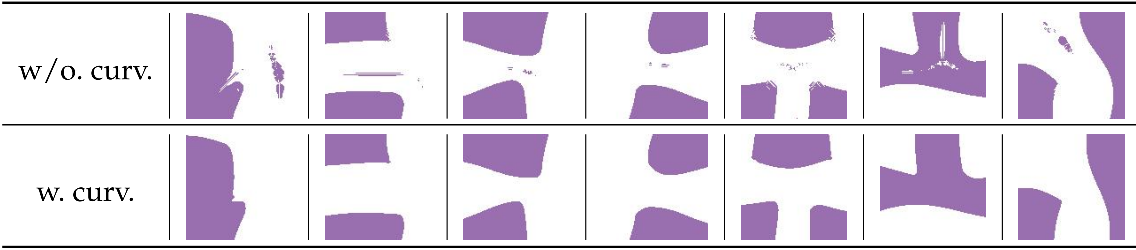


---

[7] Bentian Jiang et al. (2020). “Neural-ILT: Migrating ILT to Nerual Networks for Mask Printability and Complexity Co-optimizaton””. In: *Proc. ICCAD*.

DevelSet Architecture [ICCAD'21][8]



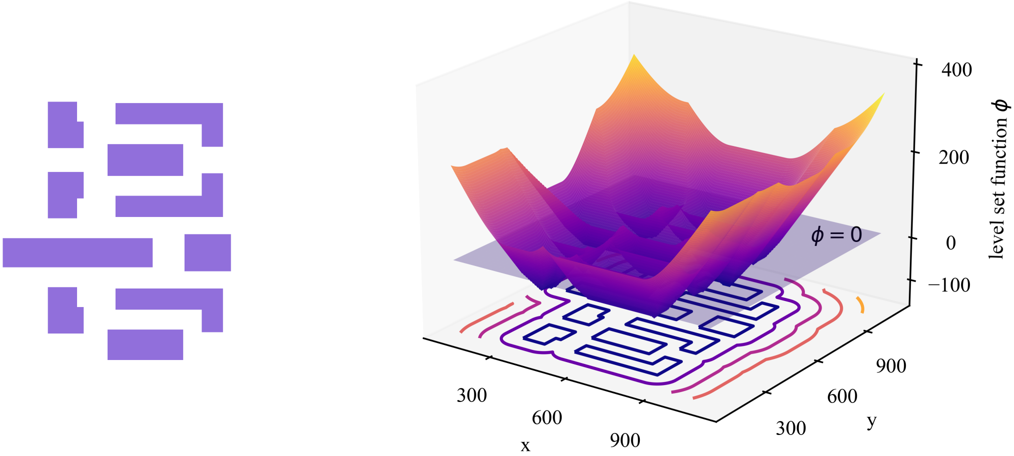


---

[8] Guojin Chen, Ziyang Yu, et al. (2021). "DevelSet: Deep neural level set for instant mask optimization". In: *Proc. ICCAD*.

Level Set of Mask

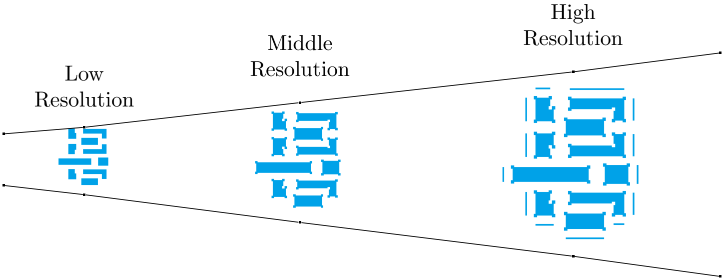
The level set function is defined as: min distance of each point to the boundary.



(a) target (b) levelset function

Multi-level ILT [DAC'23][9]

- From low resolution to high resolution
- Save runtime due to less complex computation in low resolution

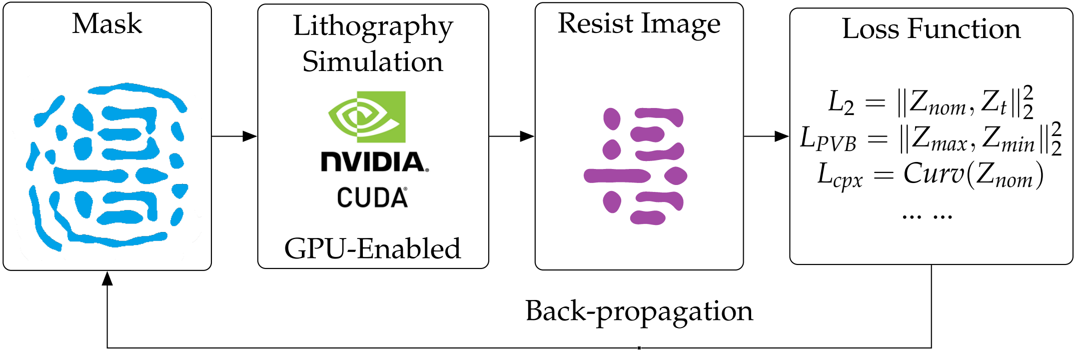


---

[9] Shuyuan Sun et al. (2023). "Efficient ILT via Multi-level Lithography Simulation". In: *Proc. DAC*.

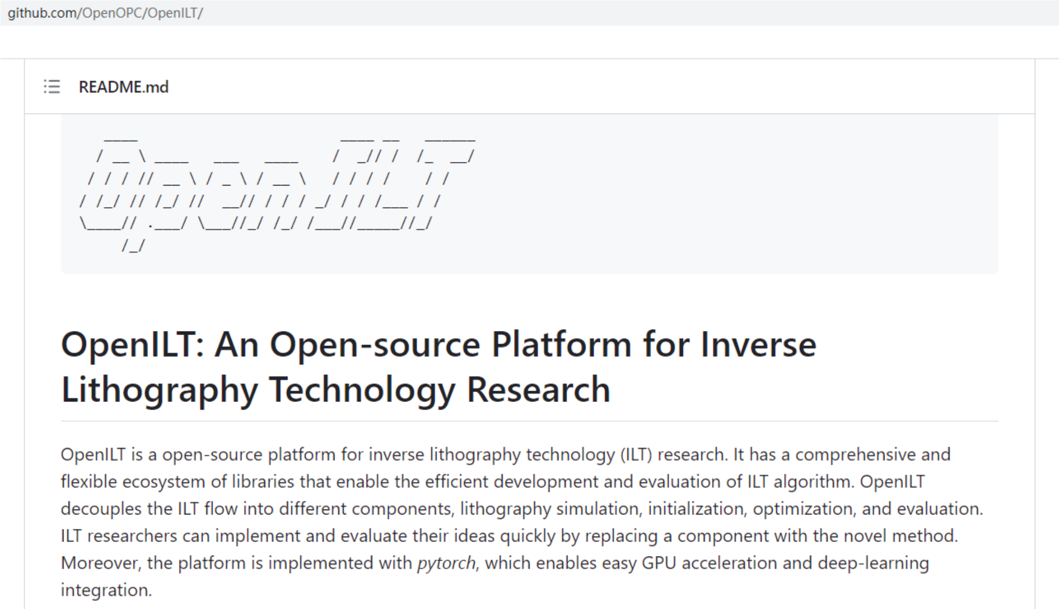
OpenILT [ASICON'23]

- GPU Acceleration → Better Runtime
- PyTorch-Based → Easier Development
- Open-Source → More Accessible



$$L_2 = \|Z_{nom}, Z_t\|_2^2$$
$$L_{PVB} = \|Z_{max}, Z_{min}\|_2^2$$
$$L_{cpx} = Curv(Z_{nom})$$
$$\dots\ \dots$$

OpenILT [ASICON'23]



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 [ASICON'23]

- 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)
```

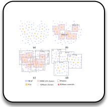
(a) Target Image (b) LevelSet (c) MultiLevel

Full-chip OPC

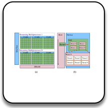
Full-chip OPC

Fullchip OPC

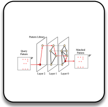
DAMO
ICCAD21



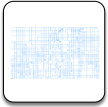
CUDA-Litho
DATE21

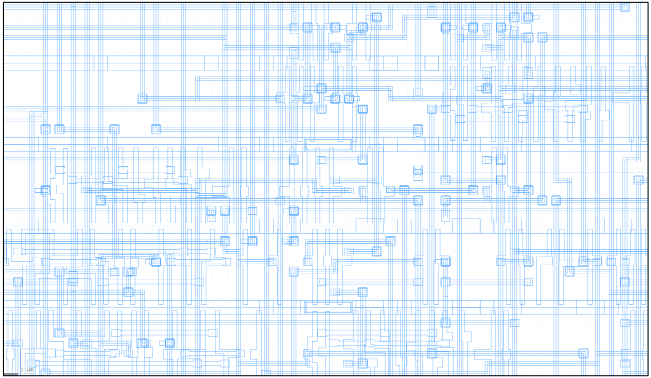


AdaOPC
ICCAD22

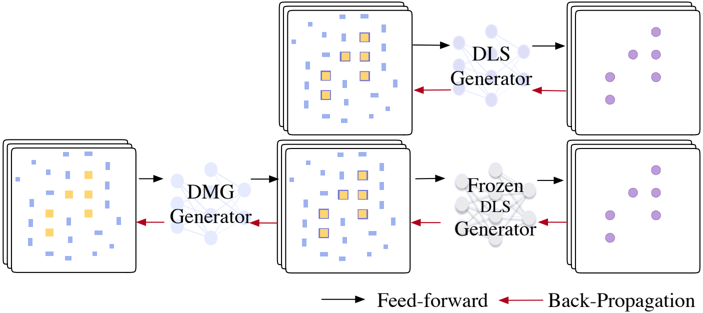


CUDA-Fullchip
Now





Deep Mask Optimization [ICCAD'20][8]

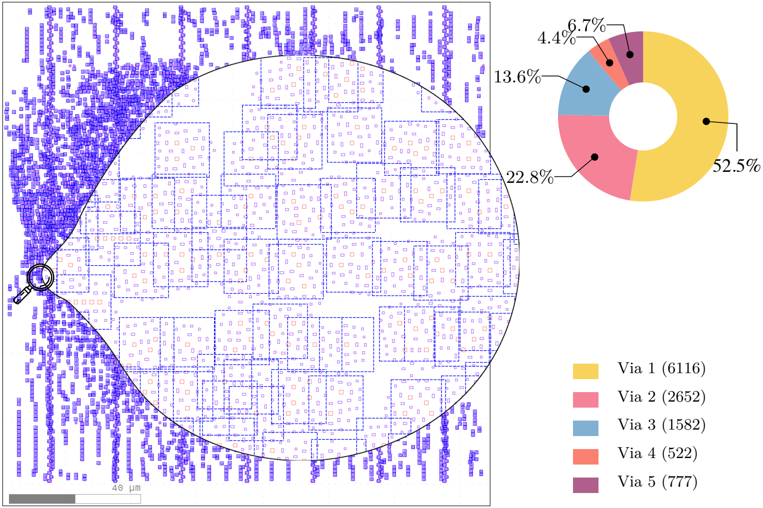


	GAN-OPC			Calibre			**DMO**		
	L_2 (nm)	PV Band (nm^2)	Runtime (s)	L_2 (nm)	PV Band (nm^2)	Runtime (s)	L_2 (nm)	PV Band (nm^2)	Runtime (s)
case 1	7456	11424	284	5159	11671	1417	**4631**	**11166**	352
case 2	7321	11215	281	4987	11463	1406	**4432**	**10955**	336
case 3	7102	11265	285	5420	11516	1435	**4802**	**11032**	367
case 4	8032	11642	322	5382	11910	1606	**4835**	**11265**	399
Average	7478	11386	293	5237	11640	1466	**4675**	**11104**	363
Ratio	1.60	1.03	0.80	1.12	1.05	4.04	**1.00**	**1.00**	1.00

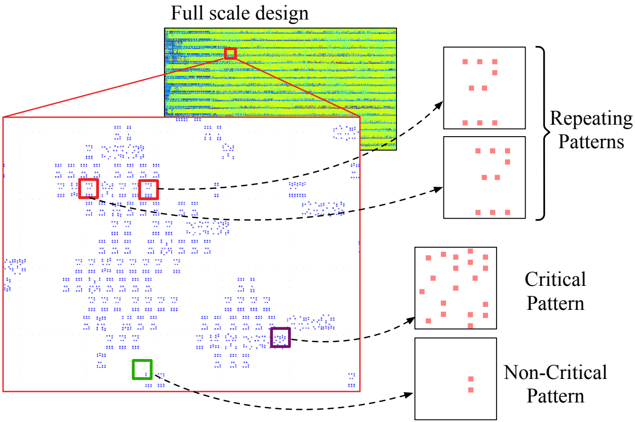
---

[8] Guojin Chen, Wanli Chen, et al. (2020). "DAMO: Deep Agile Mask Optimization for Full Chip Scale". In: *Proc. ICCAD*.

Results on ISPD 2019 datasets



Database Query in Real Design pattern [ICCAD'22][9]



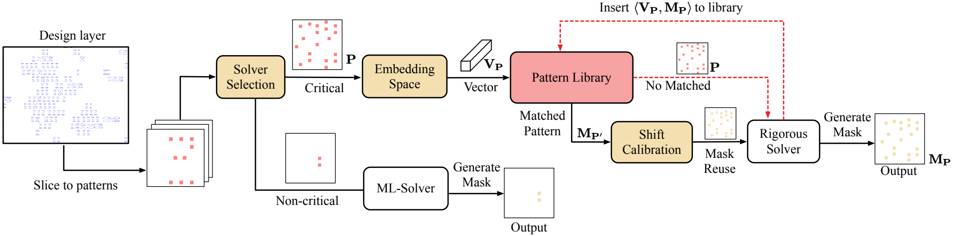
- Uneven scattering. → Solver selection
- Large ratio of repetition. → Mask Reuse

---

[9] Wenqian Zhao et al. (2022). "AdaOPC: A Self-Adaptive Mask Optimization Framework For Real Design Patterns". In: *Proc. ICCAD*.

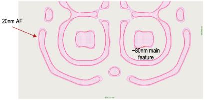
Adaptive flow for full-chip layout

- Superior in both speed& performance



Test Case ID	DAMO-DGS #EPE	DAMO-DGS PVB (nm^2)	DAMO-DGS RT (s)	ILT-GPU #EPE	ILT-GPU PVB (nm^2)	ILT-GPU RT (s)	AdaOPC #EPE	AdaOPC PVB (nm^2)	AdaOPC RT (s)
1	22	23323	5.20	23	23329	41.15	22	23232	5.50
2	26	26729	5.26	25	26762	48.5	24	26580	5.41
3	27	26938	5.22	24	26720	55.92	24	26718	5.37
4	36	27975	5.18	29	28127	70.57	25	27934	5.40
5	35	28805	5.32	30	28925	66.89	30	28927	5.44
6	30	26960	5.31	25	26762	55.81	24	26775	5.38
7	33	26382	5.23	28	26453	59.47	28	26281	5.43
8	32	30646	5.38	25	29450	54.88	27	29341	5.42
9	25	24054	5.25	24	24053	70.62	23	24022	5.43
10	24	21939	5.29	23	21701	37.59	22	21644	5.53
Avg.	29.0	26375	**5.26**	25.6	26228	56.14	**24.9**	**26145**	5.43
Ratio	1.165	1.009	**0.970**	1.028	1.003	10.340	**1.000**	**1.000**	1.000

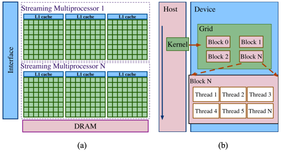
Roadmap of Full-chip OPC Research



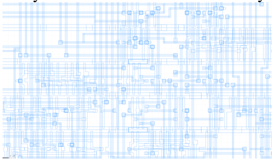
SPIE, Leo Pang
Academic Guidelines

DATE21

GLS-ILT
CUDA-Accelerated Lithography Modeling



Metal Layer: ML solution for industrial layout



2010s

2021

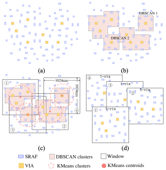
Now

2020

2022

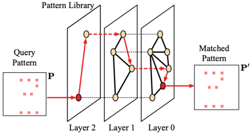
ICCAD20

DAMO
First Full-chip
ML solution



ICCAD22

AdaOPC
First Adaptive
Full-chip
OPC solution

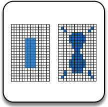


Conclusion

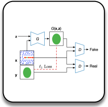
Summary of Research Progress

Lithography Modeling

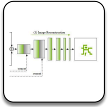
MOSAIC
ICCAD13



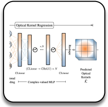
LithoGAN
DAC19



DOINN
DAC22

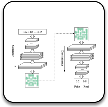


Nitho
DAC23

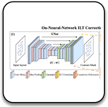


OPC

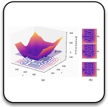
GAN-OPC
DAC18



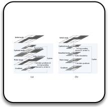
NeuralILT
ICCAD20



DevelSet
ICCAD21



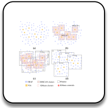
MultiILT
DAC23



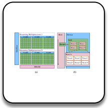
OpenILT
Now

Fullchip OPC

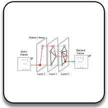
DAMO
ICCAD21



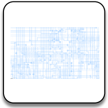
CUDA-Litho
DATE21



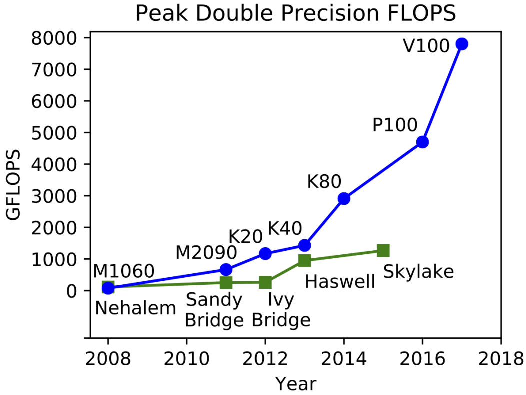
AdaOPC
ICCAD22



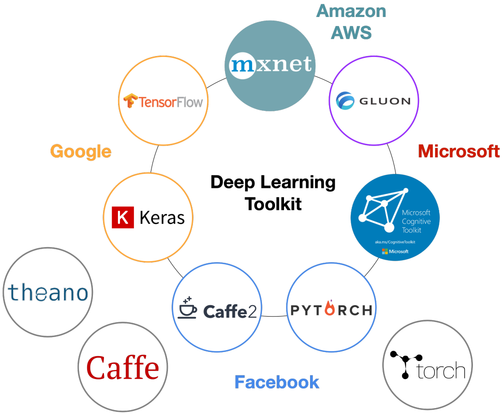
CUDA-Fullchip
Now



Machine leaning and CUDA acceleration

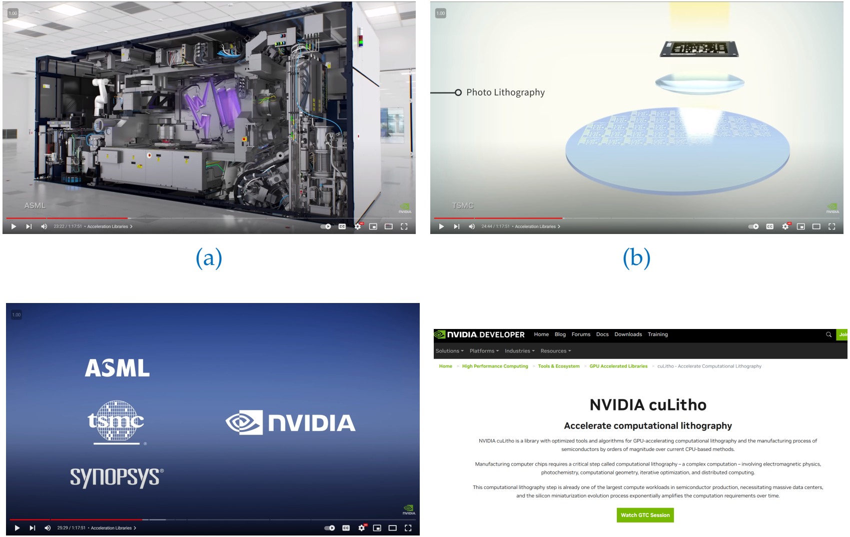


Over **60x** speedup in neural network training since 2013



Deep learning toolkits

Start from NVIDIA GTC CuLitho



(a) (b)

(c) (d)

NVIDIA DEVELOPER Home Blog Forums Docs Downloads Training

Solutions Platforms Industries Resources

Home > High Performance Computing > Tools & Ecosystem > GPU Accelerated Libraries > cuLitho - Accelerate Computational Lithography

NVIDIA cuLitho

Accelerate computational lithography

NVIDIA cuLitho is a library with optimized tools and algorithms for GPU-accelerating computational lithography and the manufacturing process of semiconductors by orders of magnitude over current CPU-based methods.

Manufacturing computer chips requires a critical step called computational lithography – a complex computation – involving electromagnetic physics, photochemistry, computational geometry, iterative optimization, and distributed computing.

This computational lithography step is already one of the largest compute workloads in semiconductor production, necessitating massive data centers, and the silicon miniaturization evolution process exponentially amplifies the computation requirements over time.

Watch GTC Session

(a) EUV lithograph machine (ASML). (b) Photo lithography. (c) and (d) Nvidia-cuLitho has 40× accleeration.

EUV: Extreme-Ultraviolet Lithography

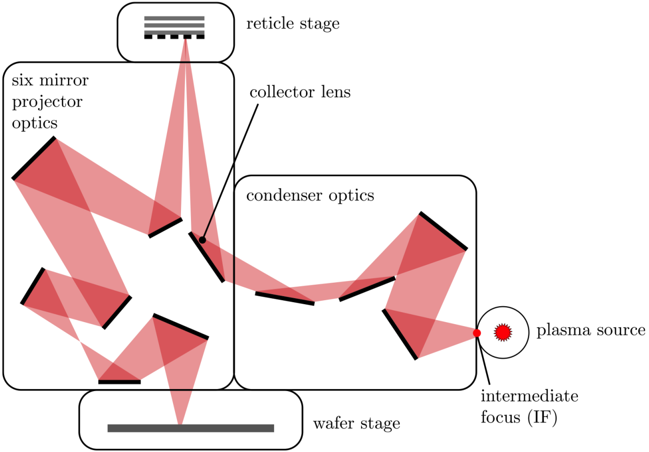


Figure 2.16: EUVL projection system employing all-reflective optical components. In this example, a six-mirror projection system suited for NAs of up to 0.35 is depicted. In order to reduce down-times in case of maintenance, condenser, projector, and the mask and wafer stages are placed into individual vacuum chambers.

How will this mask print?