TorchResist: Open-Source Differentiable Resist Simulator

Zixiao Wang[1], Jieya Zhou[1], Su Zheng[1], Shuo Yin[1], Kaichao Liang[2], Shoubo Hu[2], Xiao Chen[2], and Bei Yu[1]

[1] The Chinese University of Hong Kong, Hong Kong SAR
[2] Huawei, Hong Kong SAR

ABSTRACT

Recent decades have witnessed remarkable advancements in artificial intelligence (AI), including large language models (LLMs), image and video generative models, and embodied AI systems. These advancements have led to an explosive increase in the demand for computational power, challenging the limits of Moore's Law. Optical lithography, a critical technology in semiconductor manufacturing, faces significant challenges due to its high costs. To address this, various lithography simulators have been developed. However, many of these simulators are limited by their inadequate photoresist modeling capabilities. This paper presents **TorchResist**, an open-source, differentiable photoresist simulator.TorchResist employs an analytical approach to model the photoresist process, functioning as a white-box system with at most twenty interpretable parameters. Leveraging modern differentiable programming techniques and parallel computing on GPUs, TorchResist enables seamless co-optimization with other tools across multiple related tasks. Our experimental results demonstrate that TorchResist achieves superior accuracy and efficiency compared to existing solutions. The source code is publicly available at https://github.com/ShiningSord/TorchResist.

Keywords: Photoresist Simulation, Differentiable Programming, Analytical Modeling

1. INTRODUCTION

The rapidly growing demand for computational density in modern AI applications, such as LLMs[1–3] and generative AI (GenAI)[4,5] models, poses significant challenges to the electronics industry. As semiconductor nodes continue to shrink and transistor counts rise, optical lithography,[6] a critical technology in semiconductor manufacturing, has become indispensable in current integrated circuit (IC) fabrication processes, accounting for approximately 30% to 40% of production costs. To reduce these costs, various lithography simulators have been developed.[7–11] Recent advancements leverage the computational power of GPUs and machine learning (ML)[10,12–15] techniques to accelerate simulations. However, their inadequate resist modeling capabilities reduce the overall effectiveness and accuracy of these lithography simulators.

The basic principle behind the operation of a photoresist is the change in solubility of the resist in a developer upon exposure to light.[6] The modeling of the resist process[16,17] commonly involves multiple steps after exposure. These steps include post-exposure bake (PEB), development, and postbake. A simplified process is illustrated in Figure 1. During exposure, a strong acid is generated. However, this acid alone does not change the solubility of the resist. During the PEB process, the photogenerated acid catalyzes a reaction that alters the solubility of the polymer resin in the resist. This reaction creates a solubility differential between the exposed and unexposed regions of the resist. Additionally, the PEB process facilitates acid diffusion, which helps to remove standing waves. Development[18] is one of the most critical steps in the photoresist process. In this step, the resist dissolves in the developer, leaving behind the designed patterns on the photoresist. These patterns are used for further pattern transfer. Finally, postbake is applied to harden the resist image. This step ensures that the resist can withstand harsh environments, such as implantation or etching.

In the early stages of lithography, researchers focused on accurately modeling the physical-chemical processes to predict and control the resist behavior, achieving significant success at larger nodes.[19,20] However, as lithography progresses to advanced, smaller nodes, strict analytical modeling of the resist process becomes increasingly difficult due to the intricate physical-chemical interactions[21] at the nanometer scale and the growing complexity of production techniques.[22] In recent years, a variety of threshold-based[23] and network-based resist models[24] have been developed. Threshold-based methods are valued for their simplicity and computational

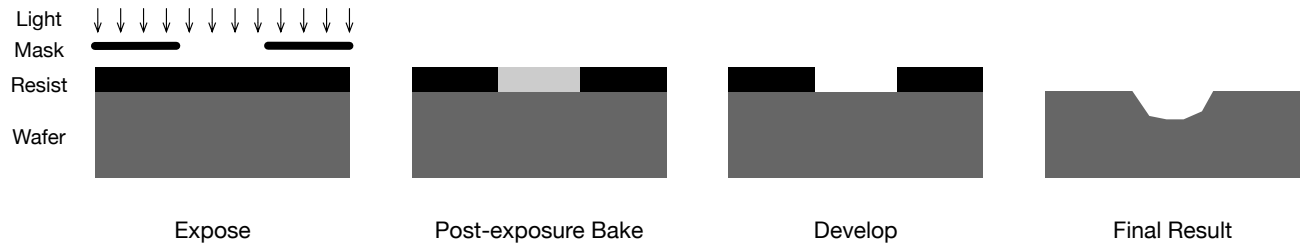

Figure 1. Illustration of a positive resist process.

efficiency but are constrained by limited model capacity. In contrast, network-based methods leverage neural networks to model the resist process, iteratively updating the network's parameters using existing aerial-wafer image pairs. While network-based methods excel at predicting data with distributions similar to the training set, their generalization to unseen data remains uncertain, and the resulting models often lack interpretability and explainability.

The core principle of TorchResist is to regularize the resist model using off-the-shelf analytical formulations while delegating parameter calibration to well-established numerical methods, leveraging a given calibration dataset. This strict formulation ensures sufficient model capacity while serving as a strong regularizer, enabling TorchResist to achieve robust generalization on unseen data. Unlike existing network-based methods, which often involve millions of trainable parameters, TorchResist is designed with fewer than twenty interpretable parameters, allowing it to converge efficiently on relatively small datasets. Additionally, by utilizing modern differentiable programming tools and harnessing the parallel computing power of GPUs, TorchResist significantly accelerates both calibration and inference processes.

Differentiable Programming and Parallel Computation. Differentiable programming, supported by tools like TensorFlow,[25] PyTorch,[26] and JAX,[27] enables efficient optimization of computational models through automatic differentiation, with forward and backward methods for gradient computation. Its applications span probabilistic programming, Bayesian inference, robotics, and computational lithography. GPU parallel computing accelerates these tasks by processing multiple operations concurrently, significantly reducing computation time in areas such as semiconductor lithography, where it handles mask and resist results simultaneously, enhancing efficiency in high-performance computing.

2. ALGORITHMS

2.1 Resist Models

Exposition Model. The exposure of the resist is the initial step in the process. As described in previous work,[16] when light passes through the resist without reflection, the Lambert-Beer law can be applied to characterize the optical absorption:

$$\frac{d\mathbf{I}}{dh} = -\mathbf{I}\sum_i a_i m_i, \tag{1}$$

where $\mathbf{I}$ represents the light intensity, h denotes the distance from the resist-air interface, and a_i and m_i are the molar absorption coefficient and molar concentration of the i-th component, respectively. We consider three absorbing species: the inhibitor, the base resin, and the reaction products. For a positive photoresist, Equation (1) can be further specified as:

$$\frac{\partial \mathbf{I}(h,t)}{\partial h} = -\mathbf{I}(h,t)\left[a_1 \boldsymbol{m}_1(h,t) + a_2 \boldsymbol{m}_2(h,t) + a_3 \boldsymbol{m}_3(h,t)\right], \tag{2}$$

where a_1, a_2, and a_3 are the molar absorption coefficients of the inhibitor, base resin, and reaction products, respectively. Similarly, $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$ represent the molar concentrations of the inhibitor, base resin, and

reaction products. The variables h and t denote the depth in the film and the exposure time, respectively. The destruction of inhibitor can be obtained via,

$$\frac{\partial \boldsymbol{m}_1(h,t)}{\partial t} = -\boldsymbol{m}_1(h,t)\mathbf{I}(h,t)C, \tag{3}$$

where C is the fractional decay rate of inhibitor per unit intensity.

Assuming the boundary condition of light intensity, the aerial image $\mathbf{R}(x,y)$, is given by optical lithography simulation and the lamp intensity is consistent during the exposure, we have,

$$\mathbf{I}(0,t) = \mathbf{R}. \tag{4}$$

Considering the initial inhibitor uniformity, resin uniformity and resin does not bleach,

$$\begin{aligned} \boldsymbol{m}_1(h,0) &= m_{10}; \\ \boldsymbol{m}_2(h,t) &= m_{20}. \end{aligned} \tag{5}$$

And reaction product is generated from inhibitor, and the amount of substance is conserved.

$$\boldsymbol{m}_3(h,t) = m_{10} - \boldsymbol{m}_1(h,t). \tag{6}$$

By substituting Equations (4) to (6) into Equations (2) and (3), normalizing, and replacing constants, we obtain:

$$\begin{aligned} \frac{\partial \mathbf{I}(h,t)}{\partial h} &= -\mathbf{I}(h,t)[A\mathbf{M}(h,t) + B]; \\ \frac{\partial \mathbf{M}(h,t)}{\partial t} &= -\mathbf{I}(h,t)\mathbf{M}(h,t)C, \end{aligned} \tag{7}$$

where $\mathbf{M}(h,t) = \frac{\boldsymbol{m}_1(h,t)}{m_{10}}$ is fractional inhibitor concentration. $A = (a_1 - a_3)m_{10}$, $B = (a_2 m_{20} + a_3 m_{10})$ and C are the constant that should be further calibrated. Equation (7) can be further solved with the following initial conditions and boundary conditions,

$$\begin{aligned} \mathbf{M}(h,0) &= 1; \\ \mathbf{M}(0,t) &= \exp(-\mathbf{R}Ct); \\ \mathbf{I}(h,0) &= \mathbf{R}\exp[-(A+B)h]; \\ \mathbf{I}(0,t) &= \mathbf{R}. \end{aligned} \tag{8}$$

Development model. The bulk development model proposed in[17] can describe the reaction of developer with the resist. Assuming k_D is the rate of diffusion of developer to resist surface, k_R is the rate constant, the rate of development can be described with,

$$\boldsymbol{r} = \frac{k_D k_R \mathbf{D}\boldsymbol{m}_3^n}{k_D + k_R \boldsymbol{m}_3^n}, \tag{9}$$

where $\mathbf{D}$ is the bulk developer concentration and we assume n molecules of product $\boldsymbol{m}_3$ react with the developer to dissolve a resin molecule. By using Equation (6) and the fractional inhibitor concentration $\mathbf{M}(h,t)$, Equation (9) can be rewritten as,

$$\boldsymbol{r} = \frac{k_D \mathbf{D}(1-\mathbf{M})^n}{k_D/k_R m_{10}^n + (1-\mathbf{M})^n}. \tag{10}$$

As described in Equation (8), when resist unexposed ($t = 0$), we have $\mathbf{M} = 1$ and the rate is zero. When resist completely exposed, we have $\mathbf{M} = 0$ and the rate is equal to $r_{\max}$,

$$r_{\max} = \frac{k_D \mathbf{D}}{k_D/k_R m_{10}^n + 1}. \tag{11}$$

Let a be a constant,

$$a = k_D/k_R m_{10}^n. \tag{12}$$

The physical meaning of a is an inflection point in the rate curve. By letting,

$$\frac{d^2\boldsymbol{r}}{d\mathbf{M}^2} = 0, \tag{13}$$

we have,

$$a = \frac{n+1}{n-1}(1 - m_{\mathrm{TH}})^n, \tag{14}$$

where m_{TH} is the value of $\mathbf{M}$ at the inflection point. By replacing the constant and taking the finite dissolution rate of unexposed resist ($r_{\min}$) into consideration, the final rate model is,

$$\boldsymbol{r} = r_{\max}\frac{(a+1)(1-\mathbf{M})^n}{a + (1-\mathbf{M})^n} + r_{\min}, \tag{15}$$

where m_{TH}, $r_{\max}$, and $r_{\min}$ should be determined experimentally.

Once the development rate is obtained, the front of developer can be computed by finding the time required to reach each point $\mathbf{T}(z, x, y)$, and we have,

$$|\nabla \mathbf{T}(z, x, y)| = \frac{1}{\boldsymbol{r}(z, x, y)}. \tag{16}$$

If we use a simplified model where we only consider the vertical development path, the time can be computed as,

$$\mathbf{T}(z, x, y) = \int_0^h \frac{dh}{\boldsymbol{r}(z, x, y)}. \tag{17}$$

However, the development path is general not strictly vertical, the fast-marching level-set methods and their variants[18,28,29] can be employed to solve this problem. Given the speed field $\boldsymbol{r}$, the time required to reach each point in the field can be computed. And the final development result is the envelop $\mathbf{T}(z, x, y) = t_{\mathrm{dev}}$, where t_{dev} is the development time.

In summary, there is a group of parameters should be further decided in TorchResist. Then we decide the values with popular numerical methods.

2.2 Parameter Optimization

We have formulated the resist process in previous subsection, and denote it as $f_\theta(\cdot)$, several parameters θ still wait for further calibration. In our case, the calibration is conducted on a dataset of gray-scale aerial image and binary wafer image pairs $\{(\mathbf{R}_i, \mathbf{W}_i)\}_{i=1}^N$. The target of the optimization is to minimize the difference of the prediction and target by optimizing the parameter θ and τ,

$$\theta, \tau = \arg\min_{\theta,\tau} \|\mathbf{W} - \Gamma(f_\theta(\mathbf{R}), \tau)\|_0, \tag{18}$$

where $\Gamma(\cdot, \tau)$ is a threshold function to binaryize the output of resist simulator and τ is an adjustable threshold. Obviously, directly optimize Equation (18) is difficult due to the exist of $L0$ norm and threshold function Γ.

Differentiable Object Functions. To address is issue, a widely applied trick is employing the sigmoid function $\sigma(\cdot)$ and replacing the $L0$-norm with the binary cross-entropy (BCE) between the predicted value and groundtruth.

$$\sigma(h) = \frac{e^h}{1 + e^h},$$
$$\mathrm{BCE}(h_1, h_2) = -[h_2 \log(h_1) + (1 - h_2)\log(1 - h_1)],$$

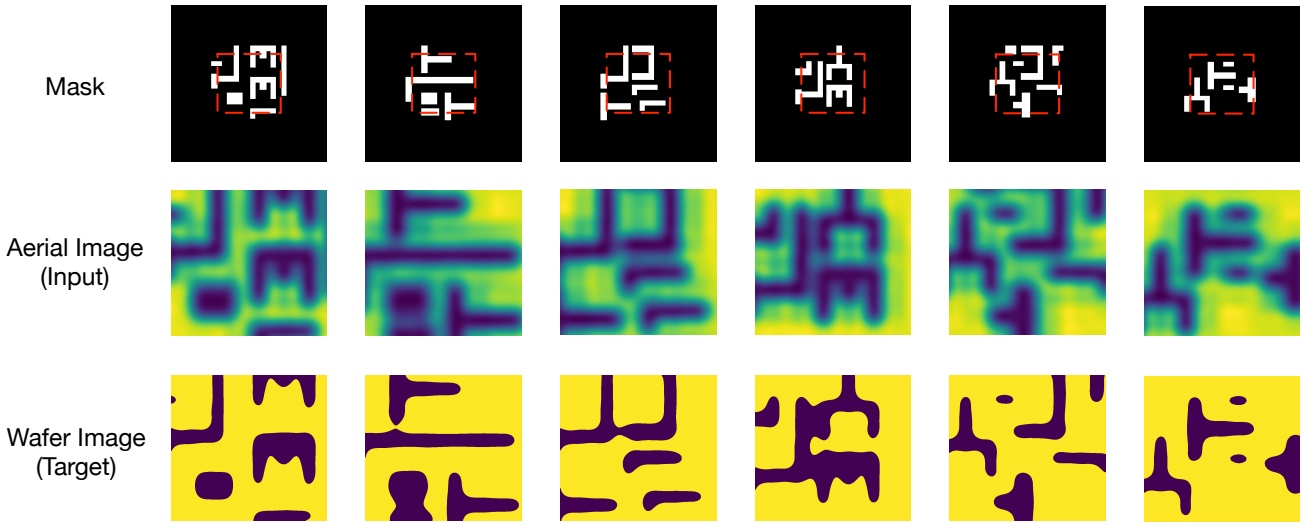

Figure 2. Illustration of the utilized dataset. The region of interest within the mask is highlighted by red dashed lines.

where $h_1 \in [0, 1]$ is the prediction and $h_2 \in \{0, 1\}$ is the target. And the differentiable object function gives,

$$\text{loss} = \text{BCE}(\sigma(sf_\theta(\mathbf{R}) - s\tau), \mathbf{W}), \tag{19}$$

where s is a scale factor to sharpen the boundary of prediction. We should note that every step in the formulation of resist model f_θ is differentiable, and we implement the process with modern automatic differentiation framework PyTorch.[26] Therefore, the optimization of Equation (19) can be easily achieved with popular gradient-decent methods, which we will further detailed in next section.

3. NUMERICAL RESULTS

3.1 Experiment Details

Dataset. LithoBench[30] is a well-known layout pattern dataset consisting of 133,496 tiles organized into several subsets. For our work, we focus exclusively on the MetalSet subset, which is generated from the ICCAD-13 benchmark[7] and contains 16,472 tiles, each measuring 2048×2048 nm^2. For each tile, we crop the central region with dimensions of 812×756 nm^2 and utilize commercial tools to generate both the aerial image and the corresponding resist image. The aerial image is a grayscale representation indicating the intensity distribution in the region of interest, while the resist image is a binary representation that illustrates the resist result. We show several examples in Figure 2. Both the commercial resist tools and our TorchResist framework ensure that the resolutions of aerial and resist images are consistent at $7nm$ per pixel during the resist simulation process. We treat the generated aerial images and resist images as the inputs and target outputs of our TorchResist and baseline resist methods. We randomly select 20% of whole dataset as calibration set and treat the remaining ones as the test set. The calibration/test split is uniform across all resist methods.

Parameter Optimization. Some of the parameters are pre-determined based on domain knowledge. For example, in our settings, we set the absorption coefficient of the resist, $\alpha = A + B$, to 6.186/nm, with $A = 0$. The resist thickness is fixed at 75 nm. The number of n in the development model is set to five for our experiments. Additionally, we always set the boundary sharpness factor s to six.

To calibrate the remaining parameters in TorchResist, we use the popular gradient descent method. We adjust the parameters on the training set for a total of 9 epochs. The learning rate is initialized at 1e-2 and is scaled by a factor of 0.3 after every 3 epochs. The optimizer used is Adam[31] with $\beta_1, \beta_2 = 0.9, 0.999$. The batch size is set to 16. The entire training process takes approximately 1 hour on a single NVIDIA A100 GPU.

Table 1. The performance comparison of the resist model on LithoBench. The lithography model is a commercial tool.

Resist Model	Pixel Difference (%)	EPEMean (nm)	EPEmax (nm)	Differentiable	Depth Simulation
Fixed Thres.[7]	0.59	1.52	4.45	✗	✗
Variable Thres.[23]	0.49	1.21	3.95	✗	✗
TorchResist	0.22	0.73	2.87	✓	✓

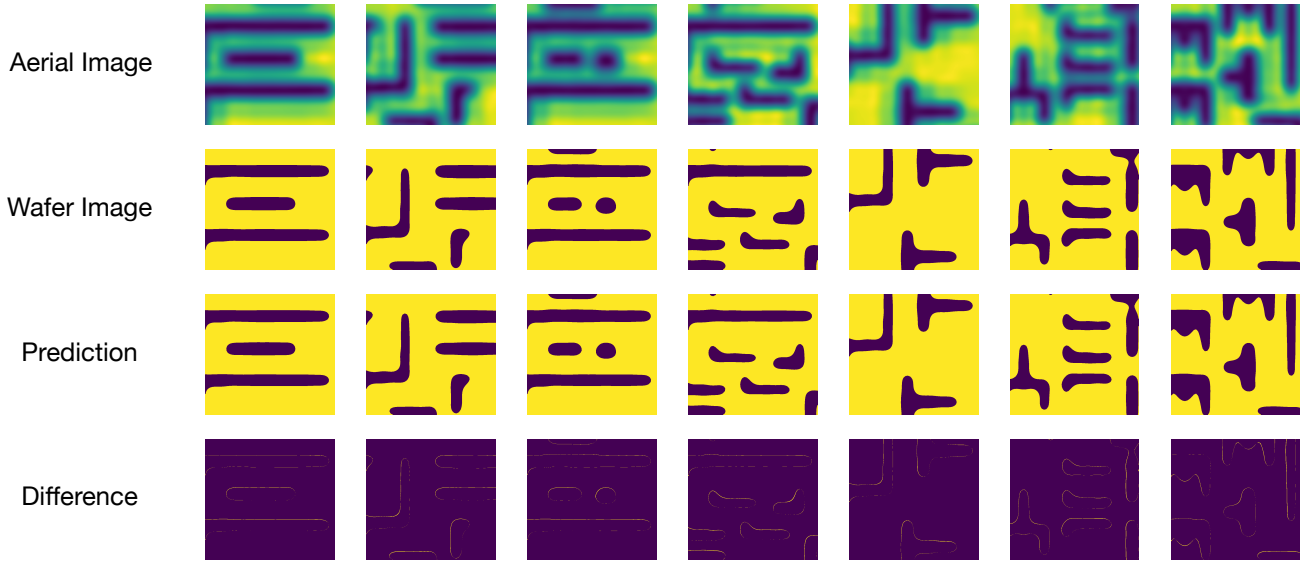

Figure 3. Illustration of the predictions of TorchResist. We also compare the predictions with groundtruth for the reference. The resolution of all the figures are 1 nm/pixel.

3.2 Evaluation

Evaluation Metrics. After training is completed, we fix the parameters of TorchResist. The predicted developed depth image is first up-sampled to 1 nm per pixel and then thresholded by the threshold τ, as explained in Equation (18). The up-sampling algorithm used is bilinear interpolation. We compare the predictions of TorchResist on the test set with the corresponding ground truth to evaluate its performance. Quantitatively, we employ three evaluation metrics: Pixel Difference, Edge Placement Error (EPE)-mean, and EPE-max.

Pixel Difference is the normalized L_0-norm between the prediction and the ground truth:

$$\text{Pixel Difference} = \frac{\|\mathbf{W} - \Gamma(f_\theta(\mathbf{R}), \tau)\|_0}{\#\text{pixels in total}} \times 100\%. \tag{20}$$

EPE estimates the difference between the edges of the ground truth and the predicted edges. EPE-mean is the average EPE value across all edges in a tile, while EPE-max is the maximum EPE value in a tile. The reported values represent the averages across all tiles in the test set.

Baseline Methods. We compare our TorchResist with two baseline methods: Fixed Threshold[7,30] and Variable Threshold.[23] The Fixed Threshold method is the simplest resist method, which applies a fixed threshold to the aerial images to obtain the final binary resist results. The Variable Threshold method uses an adaptive threshold to determine the resist result, given by:

$$\tau_{var} = M_1 + M_2 R_{\max}, \tag{21}$$

where M_1 and M_2 are constants to be determined, and $R_{\max}$ is the maximum value of the aerial image in a local region. We fine-tune the constants for both baselines on the training set and evaluate their performance on the test set.

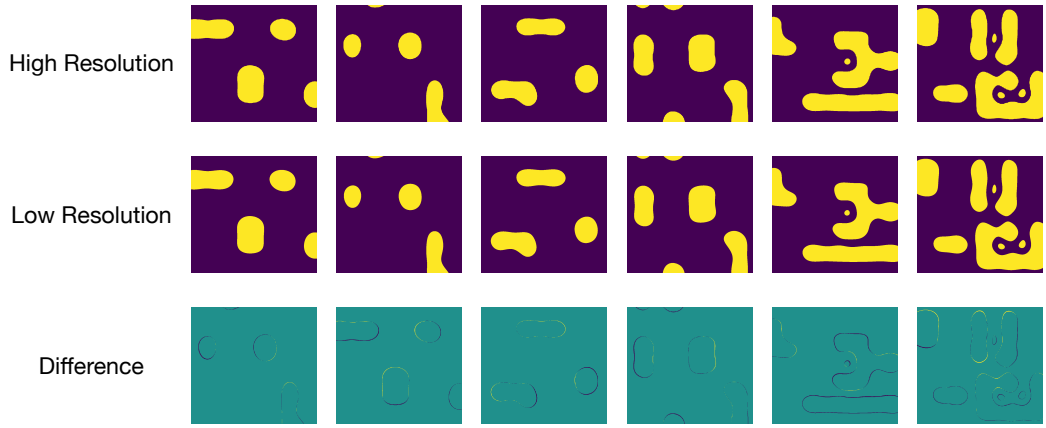

Figure 4. Comparison between results obtained at different resolutions.

Table 2. Comparison of model efficiency, measured by the average time required to process a 2µm × 2µm patch at different resolutions. Inference is performed using TorchResist on a single NVIDIA 3090 GPU.

Resolution	Cost Time(s)	Ratio
7 nm/pixel	0.04	1.00
1 nm/pixel	1.98	48.96

Table 3. The performance of TorchResist with different open-source lithography models.

Litho Model	Resist Model	Pixel Difference (%)	EPEMean (nm)	EPEmax (nm)
FUILT[9]	TorchResist-F	1.77	7.03	31.91
ICCAD13[7]	TorchResist-I	3.39	10.62	59.98

Results. We predict the resist values for the aerial images provided by commercial tools using all three resist methods and compare the results in Table 1. We also show visualizations of the predicted results in Figure 3. The results indicate that TorchResist outperforms both threshold-based methods, demonstrating its superior model capacity. Furthermore, TorchResist can simultaneously output both the binary resist and the development depth, which is beneficial for downstream tasks that require 3D simulation results.

Efficiency and Scale Robustness. We also evaluate the efficiency of TorchResist on a single NVIDIA 3090 GPU. We conduct experiments at both 1nm/pixel and 7nm/pixel resolution and report the average processing time for a single mask. The results are summarized in Table 2. We also test the scale robustness of TorchResist by comparing the results at different resolutions. We should note the all the model parameters keep the same under different resolutions. The average pixel difference between them is 0.17%. Some examples at different resolutions are shown in Figure 4. The result shows the scale robustness of TorchResist, and the inference cost can be largely reduced without an obvious trade-off in precision.

Extension on Open-Source Lithography Models. There are two popular open-source lithography simulators that are widely used: FUILT[9] and ICCAD13.[7] However, both simulators lack reliable resist modeling, which limits their overall capabilities. To address this limitation and support further research tasks that depend on accurate resist modeling, we introduce two additional variants of TorchResist: TorchResist-F and TorchResist-I. These variants are derived from the outputs of the respective lithography simulators and commercial resist results.

The evaluation results for these variants are provided in Table 3 for reference. It is important to note that the only difference between these variants and the original TorchResist lies in the parameter values, which are adjusted to account for the differences in lithography simulations. Consequently, the evaluation metrics for these variants should not be directly compared with the results presented in the main table.

4. CONCLUSION

In this paper, we introduced TorchResist, an open-source, differentiable photoresist simulator designed to address the limitations of existing resist modeling approaches in lithography simulation. By leveraging analytical formulations and modern differentiable programming techniques, TorchResist achieves high accuracy and efficiency compared to traditional methods. Our experimental results on the LithoBench dataset demonstrate that TorchResist significantly reduces pixel difference and edge placement errors, while also providing the capability for 3D depth simulation. Furthermore, the integration of TorchResist with open-source lithography models, such as FUILT and ICCAD13, highlights its versatility and potential for broader applications in semiconductor manufacturing. As an open-source tool, TorchResist is expected to facilitate further research and development in computational lithography, enabling more robust and efficient resist modeling for advanced semiconductor nodes.

REFERENCES

[1] Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al., "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437* (2024).

[2] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al., "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774* (2023).

[3] Wang, Z., Zhang, J., Zhao, W., Farnia, F., and Yu, B., "Moreaupruner: Robust pruning of large language models against weight perturbations," *arXiv preprint arXiv:2406.07017* (2024).

[4] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," in [*Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*], 10684–10695 (2022).

[5] Wang, Z., Farnia, F., Lin, Z., Shen, Y., and Yu, B., "On the evaluation of generative models in distributed learning tasks," *arXiv preprint arXiv:2310.11714* (2023).

[6] Mack, C., [*Fundamental principles of optical lithography: the science of microfabrication*], John Wiley & Sons (2007).

[7] Banerjee, S., Li, Z., and Nassif, S. R., "Iccad-2013 cad contest in mask optimization and benchmark suite," in [*2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*], 271–274, IEEE (2013).

[8] Chen, G., Geng, H., Yu, B., and Pan, D. Z., "Open-source differentiable lithography imaging framework," in [*DTCO and Computational Patterning III*], **12954**, 118–127, SPIE (2024).

[9] Yin, S., Zhao, W., Xie, L., Chen, H., Ma, Y., Ho, T.-Y., and Yu, B., "Fuilt: Full chip ilt system with boundary healing," in [*Proceedings of the 2024 International Symposium on Physical Design*], *ISPD '24*, 13–20, Association for Computing Machinery, New York, NY, USA (2024).

[10] Watanabe, Y., Kimura, T., Matsunawa, T., and Nojima, S., "Accurate lithography simulation model based on convolutional neural networks," in [*Optical Microlithography XXX*], **10147**, 137–145, SPIE (2017).

[11] Fühner, T., *Artificial evolution for the optimization of lithographic process conditions*, PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) (2014).

[12] Yu, P., Shi, S. X., and Pan, D. Z., "True process variation aware optical proximity correction with variational lithography modeling and model calibration," *Journal of Micro/Nanolithography, MEMS and MOEMS* **6**(3), 031004–031004 (2007).

[13] Wang, Z., Shen, Y., Zhao, W., Bai, Y., Chen, G., Farnia, F., and Yu, B., "Diffpattern: Layout pattern generation via discrete diffusion," in [*2023 60th ACM/IEEE Design Automation Conference (DAC)*], 1–6, IEEE (2023).

[14] Wang, Z., Shen, Y., Yao, X., Zhao, W., Bai, Y., Farnia, F., and Yu, B., "Chatpattern: Layout pattern customization via natural language," in [*Proceedings of the 61st ACM/IEEE Design Automation Conference*], 1–6 (2024).

[15] Chen, G., Wang, Z., Yu, B., Pan, D. Z., and Wong, M. D., "Ultra-fast source mask optimization via conditional discrete diffusion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2024).

[16] Dill, F. H., Hornberger, W. P., Hauge, P. S., and Shaw, J. M., "Characterization of positive photoresist," *IEEE Transactions on electron devices* **22**(7), 445–452 (1975).

[17] Mack, C. A., "Development of positive photoresists," *Journal of the Electrochemical Society* **134**(1), 148 (1987).

[18] Sethian, J. A., "Fast-marching level-set methods for three-dimensional photolithography development," in [*Optical Microlithography IX*], **2726**, 262–272, SPIE (1996).

[19] Thackeray, J., Cameron, J., Jain, V., LaBeaume, P., Coley, S., Ongayi, O., Wagner, M., Rachford, A., and Biafore, J., "Pursuit of lower critical dimensional uniformity in euv resists," *Journal of Photopolymer Science and Technology* **26**(5), 605–610 (2013).

[20] Smith, M. D., Byers, J. D., and Mack, C. A., "The lithographic impact of resist model parameters," in [*Advances in Resist Technology and Processing XXI*], **5376**, 322–332, SPIE (2004).

[21] Thackeray, J. W., Nassar, R. A., Brainard, R., Goldfarb, D., Wallow, T., Wei, Y., Mackey, J., Naulleau, P., Pierson, B., and Solak, H. H., "Chemically amplified resists resolving 25 nm 1: 1 line: space features with euv lithography," in [*Emerging Lithographic Technologies XI*], **6517**, 394–404, SPIE (2007).

[22] Thackeray, J. W., "Materials challenges for sub-20-nm lithography," *Journal of Micro/Nanolithography, MEMS, and MOEMS* **10**(3), 033009–033009 (2011).

[23] Randall, J., Ronse, K. G., Marschner, T., Goethals, A.-M., and Ercken, M., "Variable-threshold resist models for lithography simulation," in [*Optical Microlithography XII*], **3679**, 176–182, SPIE (1999).

[24] Zach, F. X., "Neural-network-based approach to resist modeling and opc," in [*Optical Microlithography XVII*], **5377**, 670–679, SPIE (2004).

[25] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015). Software available from tensorflow.org.

[26] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems* **32** (2019).

[27] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., "JAX: composable transformations of Python+NumPy programs," (2018).

[28] Osher, S. and Fedkiw, R. P., "Level set methods: an overview and some recent results," *Journal of Computational physics* **169**(2), 463–502 (2001).

[29] Dai, Q., Guo, R., Lee, S.-Y., Choi, J., Lee, S.-H., Shin, I.-K., Jeon, C.-U., Kim, B.-G., and Cho, H.-K., "A fast path-based method for 3-d resist development simulation," *Microelectronic engineering* **127**, 86–96 (2014).

[30] Zheng, S., Yang, H., Zhu, B., Yu, B., and Wong, M., "Lithobench: Benchmarking ai computational lithography for semiconductor manufacturing," *Advances in Neural Information Processing Systems* **36** (2024).

[31] Kingma, D. P., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980* (2014).