Intelligent OPC Engineer Assistant for Semiconductor Manufacturing

Guojin Chen[1*], **Haoyu Yang**[2], **Bei Yu**[1], **Haoxing Ren**[2]

[1]Chinese University of Hong Kong
[2]NVIDIA
gjchen21@cse.cuhk.edu.hk, haoyuy@nvidia.com, byu@cse.cuhk.edu.hk, haoxingr@nvidia.com

Abstract

Advancements in chip design and manufacturing have enabled the processing of complex tasks such as deep learning and natural language processing, paving the way for the development of artificial general intelligence (AGI). AI, on the other hand, can be leveraged to innovate and streamline semiconductor technology from planning and implementation to manufacturing. In this paper, we present *Intelligent OPC Engineer Assistant*, an AI/LLM-powered methodology designed to solve the core manufacturing-aware optimization problem known as optical proximity correction (OPC). The methodology involves a reinforcement learning-based OPC recipe search and a customized multi-modal agent system for recipe summarization. Experiments demonstrate that our methodology can efficiently build OPC recipes on various chip designs with specially handled design topologies, a task that typically requires the full-time effort of OPC engineers with years of experience.

1 Introduction

Recent advancements in chip design and manufacturing have enabled the handling of intricate tasks such as deep learning and natural language processing, bringing us closer to achieving artificial general intelligence (AGI). Meanwhile, AI is revolutionizing semiconductor technology by optimizing every stage of the semiconductor lifecycle, from conceptual planning to execution and manufacturing. By leveraging AI-driven innovations, the industry can achieve greater efficiency, precision, and performance, potentially accelerating the development of next-generation chips.

In this paper, we utilize AI to address the complexities of optical proximity correction (OPC), a critical process in semiconductor manufacturing. As shown in Figure 1, OPC involves adjusting chip designs to counteract lithography distortions, ensuring that the final patterns on the silicon wafer closely match the intended design with high precision. As depicted in Figure 2, a complete OPC solution comprises both the solver and the recipe. The OPC solver includes the core OPC algorithms, such as lithography imaging computation, mask database management, gradient calculation, and shape perturbation, among other processes (Granik and Cobb 2002; Lei et al. 2014). Foundries and EDA (Electronic Design Automation) vendors each have their own OPC solver implementations, which are built on advanced algorithms and follow similar workflows (Mentor Graphics, Siemens 2024; Synopsys, Inc. 2024). The OPC recipe, however, contains specific configurations tailored to optimize a particular design. It includes common optimization parameters such as step size, maximum iterations, shape movement constraints, polygon fragmentation policies, and error control strategies. Recipes also incorporate specialized rules for handling unique chip design patterns that cannot be effectively addressed through standard optimization settings. These specialized rules are crucial in OPC and are typically developed by experienced OPC engineers through extensive trial and error.

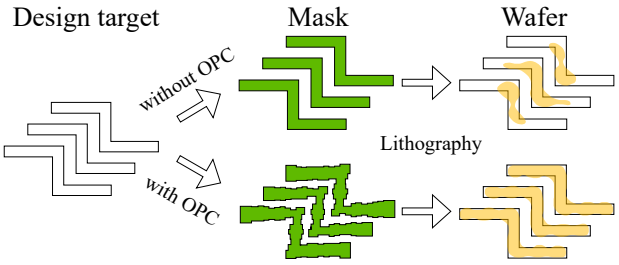


Figure 1: Motivation for optical proximity correction (OPC) in semiconductor manufacturing process.

To enhance the efficiency of chip development, we present the *Intelligent OPC Engineer Assistant*, an AI-driven framework designed to assist human engineers in the rapid development of OPC recipes. This methodology integrates a reinforcement learning (RL)-based approach for optimizing objective searches and a multi-modality large language model (MLLM)-backboned agent system to facilitate spatial reasoning and recipe summarization.

The remainder of the manuscript is organized as follows. In Section 2, we delve into the background of OPC and recipe development, highlighting several works that address related challenges using RL and LLM agents. In Section 3, we present the core framework of our algorithm, structured in a two-stage approach. Experimental results and conclusions are provided in Section 4 and Section 5, respectively.

[*]Work accomplished during internship at NVIDIA.

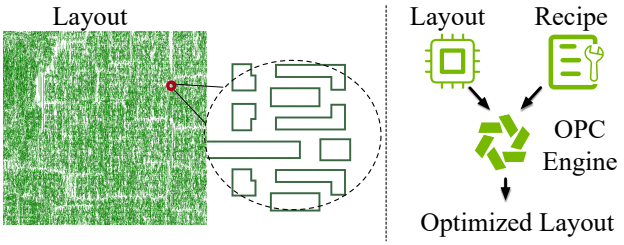


Figure 2: Left: The full-chip layout. Right: The relationship between the layout, OPC Recipe and OPC Engine.

2 Preliminaries

2.1 Related Works

OPC and Reinforcement Learning As device geometries shrink to the nanometer scale, the limitations of traditional optical lithography become apparent, necessitating advanced computational methods to achieve the desired fidelity. As illustrated in Figure 1, OPC is a critical component of computational lithography, addressing distortions and proximity effects that arise during the lithographic process. By systematically adjusting the mask design to counteract these effects, OPC ensures that the final printed patterns on the wafer closely match the intended design. This correction is pivotal for maintaining device performance, yield, and reliability in the semiconductor industry. The integration of computational lithography and OPC is thus crucial for advancing semiconductor technology, enabling the production of increasingly smaller and more complex integrated circuits. In recent years, numerous studies have utilized AI and ML to enhance OPC algorithms (Yang and Ren 2024; Zhu et al. 2023a; Chen et al. 2020). Notably, (Liang et al. 2023, 2024) have proposed reinforcement learning-based OPC approaches that integrate spatial correlations and OPC-specific principles, enhancing performance and efficiency across both metal and via layers.

OPC Recipe Development and LLM Agents As illustrated in Figure 2, the primary motivation for OPC recipe development is to enhance the productivity of OPC engineers in the context of modern semiconductor manufacturing. As depicted in Figure 3, once the OPC engine is well-established, it becomes necessary to adjust the recipe based on various factors such as pattern characteristics, location, and process nodes, in addition to optimizing the algorithm's intrinsic parameters. Figure 3 illustrates two pertinent examples from this study. The first example concerns the adjustment of EPE measurement points, depicted as red dots in Figure 3. Initially, these measurement points are distributed along the boundaries, leading to excessive optimization at the corners of the patterns. This over-optimization can degrade the overall effectiveness of the OPC process. Therefore, an optimized EPE measurement recipe involves relocating the measurement points either outward or inward from the corners to prevent excessive optimization at specific locations. Traditionally, this adjustment relied heavily on the engineers' experience and experimental tuning, which was time-consuming and costly, especially with the advent

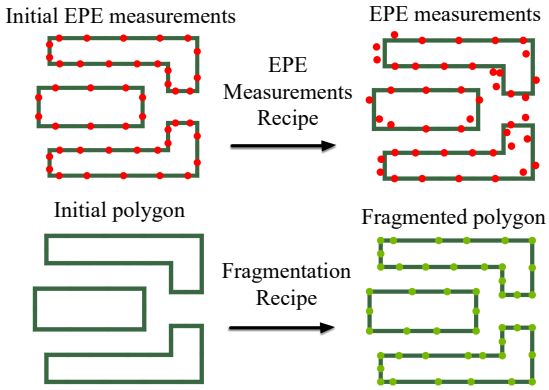


Figure 3: OPC recipe development. The upper figure shows the recipe for optimizing EPE measurement points, while the lower figure shows the edge fragmentation recipe.

of new technology nodes. The second example involves the fragmentation of polygons, which is essential for the OPC engine to function. The recipe plays a crucial role in fragmenting different polygons according to their characteristics. The illustration in Figure 3 shows the initial polygon and its fragmented version post-recipe application, demonstrating how the fragmentation recipe is tailored to ensure optimal performance of the OPC engine. As design complexity and the number of device and mask layers increase, traditional OPC methods have become inadequate for advanced nodes, necessitating the creation of more customized OPC recipes. This growing complexity has led to a surge in the engineering workload, demanding more OPC engineers to handle the sophisticated new OPC recipes (Wu et al. 2016; Asthana, Wilkinson, and Power 2016; Liu and Zhang 2010). Recently, The integration of LLMs as agents to automate the EDA process has recently attracted considerable research interest. For instance, RTLFixer (Tsai, Liu, and Ren 2023) focuses on automated debugging and code repair, while ChipNeMo (Liu et al. 2023) serves as an engineering assistant chatbot, facilitating EDA script generation and bug analysis. Additionally, ChatEDA (Wu et al. 2024) incorporates LLMs into traditional design workflows, allowing designers to interact with EDA tools through a conversational interface using natural language. These advancements significantly enhance the automation of EDA processes and improve engineering efficiency.

2.2 Evaluation Metrics for OPC

In this paper, we use domain-specific process variation band (PVB) and edge placement error (EPE) as two typical metrics to evaluate OPC performance. As shown in Figure 4, these metrics provide a comprehensive assessment of the quality of the OPC mask, capturing different aspects of the lithographic process.

Edge placement error (EPE) quantifies the geometric distortion of the resist image (Banerjee, Li, and Nassif 2013). It is calculated by sampling points along the edges of target shapes and counting the number of points where

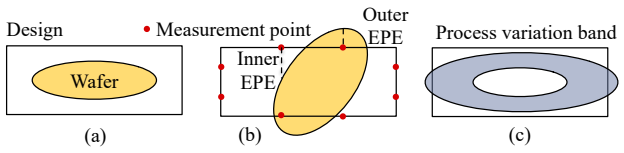


Figure 4: Evaluation metrics for OPC. (a) Definition of design target and the wafer pattern. (b) Illustration of EPE measurement points, including inner and outer EPE violations for calculating the total EPE count and EPE distance. (c) Process variation band (PVB) calculation.

the distance between the target and printed pattern exceeds a predefined threshold. The smaller the EPE, the better the OPC mask quality. In this paper, we calculate the EPE number and EPE distance to evaluate EPE. The EPE number (EPE N) is counted by the number of points where the distance between the target and printed pattern exceeds 1 nm. The EPE distance (EPE D) is summed by the distance between the target and printed pattern where the distance exceeds 1 nm.

PVB evaluates the robustness of the mask against different process conditions (Banerjee, Li, and Nassif 2013). PVB represents the range of deviations that can occur during the patterning of semiconductor features. As illustrated in Figure 4(c), this band defines the tolerances within which the process must operate to achieve the desired feature dimensions and ensure product quality. A narrower process variation band indicates better process control, crucial for manufacturing advanced semiconductor devices. It is computed by measuring the bitwise XOR region between the printed images from the maximum and minimum process conditions ($\pm 2\%$) $\boldsymbol{Z}_{max}, \boldsymbol{Z}_{min}$. $\text{PVB}(\boldsymbol{Z}_{max}, \boldsymbol{Z}_{min}) = \|\boldsymbol{Z}_{max} - \boldsymbol{Z}_{min}\|_2^2$.

2.3 Problem Formulation

By automating these processes, this paper seeks to reduce the time and cost involved in OPC recipe development, thereby improving the overall efficiency and effectiveness of computational lithography in semiconductor manufacturing. The study has two primary objectives: first, to develop an automated method for generating a comprehensive set of recipes applicable to full-chip patterns, including the optimization of EPE measurement points and layout fragmentation points; second, to ensure that these recipes minimize both the PVB and the EPE.

3 Methodology

3.1 Overview: A Two-stage Approach

In this section, we present a two-stage framework, as depicted in Figure 5, with a focus on the synergistic integration of reinforcement learning (RL) and large language models (LLM) in the development of OPC recipes. In the first stage, RL is employed to explore and identify optimal solutions for OPC recipe generation, taking into account the shapes and positions of patterns along with their surrounding contexts. Once the RL policy is trained, it can generate a set of recipes tailored to specific pattern features by adjusting EPE measurement points and fragment points. However, the RL process is computationally intensive and impractical for full-chip applications. To overcome this limitation, the second stage harnesses the capabilities of LLM to efficiently derive recipes by synthesizing the results produced by RL. This stage involves generating a zero-shot feature pool using LLM, annotating features with a vision-language model, and constructing a decision tree. The decision tree is then utilized to produce the final OPC recipes with enhanced efficiency. This two-stage framework leverages the complementary strengths of RL and LLM to optimize OPC recipe development, striking a balance between accuracy and computational efficiency.

3.2 Exploration with Reinforcement Learning

At this stage, RL can be utilized in conjunction with any OPC engines. The RL algorithm explores the parameter space and conducts a global OPC optimization, customized for various design rules and scenarios. This process aims to automate the identification of effective, fine-tuned parameter combinations, thereby reducing the time and expertise needed for manual OPC recipe adjustment.

Reinforcement Learning Framework Proximal policy optimization (PPO) (Schulman et al. 2017) is an advanced RL algorithm that has shown significant potential in optimizing complex processes. In the context of computational lithography, particularly in the development of OPC recipes, PPO offers a powerful framework for improving the precision and efficiency of photomask patterning. The objective of OPC is to adjust the mask design so that the printed patterns on the wafer closely match the intended design after exposure and development. In contrast to traditional heuristic-based OPC recipe development methods (Wu et al. 2016; Asthana, Wilkinson, and Power 2016), PPO leverages deep learning models to iteratively enhance OPC recipe development through agent-environment interactions. The environment in this context includes the polygon space defined by polygon coordinates and their corresponding rasterized images, which serve as feature embeddings in the observation space, as illustrated in Figure 5. The agent, guided by PPO, learns to adjust fragment points and EPE measure points to minimize the OPC loss function.

The state of the environment at time t, denoted as s_t, includes both the polygon coordinates and the rasterized image features. The agent takes an action a_t, which adjusts the positions of these points. The environment then transitions to a new state s_{t+1} and the agent receives a reward r_t based on the OPC loss. The goal of the PPO algorithm is to maximize the expected cumulative reward, defined as the return R_t:

$$R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k, \tag{1}$$

where γ is the discount factor and T is the time horizon. PPO optimizes a policy $\pi_\theta(a_t|s_t)$, parameterized by θ, by interacting with the environment and updating θ to maximize the

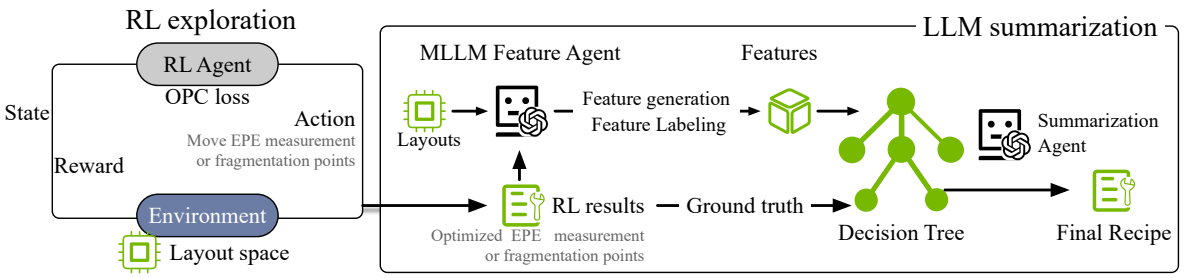


Figure 5: The two-stage approach for OPC recipe generation. The first stage employs RL to optimize OPC recipes. The second stage utilizes multi-modal LLM agents to efficiently summarize the results generated by RL and generate the final OPC recipes.

expected return. The policy update is constrained by a proximity term to ensure stability:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t \Bigg[\min \Bigg(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \Bigg) \Bigg], \tag{2}$$

where $\hat{A}_t$ is the advantage estimate and ϵ is a clipping parameter. The advantage estimate $\hat{A}_t$ is calculated as:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_T, \tag{3}$$

with the temporal difference error δ_t given by:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{4}$$

In the OPC context, the state s_t includes the current positions of the measurement points, the fragment points, and the rasterized image features. The action a_t consists of permissible adjustments to these points within a specified range of $\pm 40nm$. The reward function r_t, critical to the RL training process, is derived from the OPC loss $\mathcal{L}_{\text{OPC}}$, which quantifies the alignment between the corrected mask pattern and the target design post-lithography. To align with the RL paradigm where higher rewards are preferred, the reward is defined as the negative of the OPC loss:

$$r_t = -\mathcal{L}_{\text{OPC}}. \tag{5}$$

Mathematically, the OPC loss can be expressed as:

$$\mathcal{L}_{\text{OPC}} = \alpha \cdot \mathcal{L}_2(v, z) + \beta \cdot \text{EPE}(v, z) + \gamma \cdot \text{PVB}(v, z), \tag{6}$$

where v represents the rasterized image representation from vertices of the polygon, z is the target pattern, $\mathcal{L}_2$ is the Euclidean distance metric. EPE and PVB are edge placement error and process variation band. The coefficients α, β, and γ are weights that balance the contributions of each term to the overall loss.

Additionally, the value function $V(s_t)$ is approximated using a neural network parameterized by ϕ, and is trained to minimize the following loss:

$$\mathcal{L}_V(\phi) = \mathbb{E}_t \left[\left(V_\phi(s_t) - R_t \right)^2 \right]. \tag{7}$$

The overall training objective combines the clipped surrogate objective for policy optimization and the value function loss, along with an entropy bonus $S[\pi_\theta](s_t)$ to encourage exploration:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_t \left[\mathcal{L}^{\text{CLIP}}(\theta) - c_1 \mathcal{L}_V(\phi) + c_2 S[\pi_\theta](s_t) \right], \tag{8}$$

where c_1 and c_2 are coefficients that balance the importance of the value loss and the entropy bonus, respectively.

3.3 Recipe Summarization with LLMs

In the second stage, LLMs are used to summarize the RL-generated OPC recipes. By leveraging the summarization capabilities of LLMs, we derive more effective and generalizable OPC recipe rules. The use of LLMs in this stage ensures that the insights gained from the RL-generated recipes are translated into practical and applicable rules for OPC engineers. LLMs possess advanced natural language processing capabilities (Zhu et al. 2023b), allowing them to understand and generate human-like text. This makes them well-suited for the task of summarizing (Brown et al. 2020) and reasoning (Kojima et al. 2022) over complex OPC recipes. The summarization process involves condensing the large volume of generated recipes into a coherent set of rules that can be easily understood and applied by OPC engines and engineers. The reasoning process, on the other hand, involves analyzing the summarized rules to identify patterns and relationships that can lead to the discovery of new, more effective OPC recipes.

Although the initial phase of employing RL produced better results than the baseline, it encountered significant challenges when applied to full-chip scenarios due to the need to process millions of clips, leading to an impractically long runtime for RL. Additionally, in OPC recipe development, the final step requires the extraction of recipe rules based on pattern shapes, which are then used by commercial OPC software for pattern matching and retargeting operations.

The second phase of the framework leverages the capabilities of multimodal large models to bridge the gap between superior RL exploration outcomes and the generation of OPC recipes. To address the hallucination issues often associated with LLMs, our approach is divided into four steps: ① During the data processing stage, we convert the RL results into two distinct formats: JSON and image clips; ② We utilize LLMs to perform feature generation and zero-shot data labeling. ③ We construct a decision tree based on the labeled features. ④ Finally, the decision tree serves as

a retrieval source for the LLM, facilitating the generation of the OPC recipe. This structured methodology ensures the effective translation of RL exploration results into practical OPC recipes, enhancing the overall efficiency and accuracy of the recipe generation process.

Data Representation Transformation Since the RL action space primarily focuses on two aspects—EPE measurement movement and edge fragment point movement—the data structure can be uniformly represented in two major parts. The first part indicates whether the point's movement direction aligns with the positive direction, and the second part specifies the exact movement distance. This structured representation ensures that the RL-optimized layout information is effectively retained and utilized in OPC recipe development. More specifically, for each point e_i, we will record the RL-adjusted movement vector $\boldsymbol{\delta}$ with normal direction $\boldsymbol{i}$ and the distance δ. To facilitate the LLM's understanding of the RL results, we convert the data into a JSON format.

Zero-shot Feature Pool Generation and Feature Labeling To explicitly express the "optimization algorithm" embedded in the RL results, a straightforward approach is to input the RL outputs directly into the LLM as coordinates of segments. However, this method presents two major issues: first, the LLM cannot comprehend the spatial relationships between edges or polygons based solely on coordinates; second, due to the limitations of the LLM's context window, it cannot process exceedingly long coordinate representations, leading to judgments based only on the first few points, often resulting in incorrect assessments. To maximally preserve the layout information optimized by RL, we transcribed the information, recording location-related details and geometry features for each point.

In traditional OPC recipe development, engineers manually set EPE measurement points and fragment points based on pattern shape, position, surrounding pattern characteristics, and layer number. This is followed by forward lithography and OPC simulation to obtain evaluation results, which are iteratively adjusted. This process is time-consuming, labor-intensive, and monotonous. Recent studies, such as (Gilardi, Alizadeh, and Kubli 2023), have demonstrated that LLMs outperform human annotators in tasks related to labeling and annotation, offering higher efficiency and lower costs. Automating this process with LLMs is a straightforward idea that can significantly enhance engineers' efficiency. Our method of utilizing LLMs for classification labeling in OPC recipe development involves two steps: first, feature pool mining and generation. Second, feature labeling.

Feature Mining and Generation Raw images of EPE points and fragment points are input into a multi-modality large language model (MLLM), which analyzes the images and extracts features. For different points, the features generated by the LLM are pooled and deduplicated. The data format includes feature names and descriptions, resulting in a comprehensive feature pool. Part of the feature pool is shown below and the full feature pool is available in the supplementary material. Those features will be fed into next step for the labeling agents to label the features for each EPE measurement point and fragment point.

Feature pool examples

{`on_jog_long_edge`: "it is on the jog, but on the long edge of the jog",
`on_jog_short_edge`: "it is on the jog, but on the short edge of the jog",
`on_horizontal_edge`: "the point is located on a horizontal edge",
`on_vertical_edge`: "the point is located on a vertical edge", ... }

Feature Labeling Using the feature pool generated by MLLMs in the first step, for each EPE measurement point and fragment point, we employ the MLLMs to label the input images with corresponding prompts and evaluate each feature in the pool. By labeling each point, we obtain a series of feature information for each point, which is then used to construct our decision tree. This automated approach not only preserves the intricate details of the RL-optimized layouts but also streamlines the OPC recipe development process, making it more efficient and effective. Another critical aspect is the labeling of ground truth for decision trees. To ensure the recipe does not become overly complicated, the RL-movement vector $\boldsymbol{\delta}$ is divided into different intervals. These intervals serve as classification boundaries. For example, in a given direction, the vector is categorized into C intervals, where the furthest positive interval is labeled as $+C$ and the furthest negative interval as $-C$. Consequently, the ground truth labels range from $-C$ to $+C$, encompassing a total of $2C + 1$ categories. A label of 0 indicates no movement.

LLM Labeling Example

{`epe_id`: 10,
`features` : { `on_jog_long_edge`: false, `on_jog_short_edge`: false, `on_horizontal_edge`: true, `on_vertical_edge`: false, ... }, `result`: $+C$} ...
{`epe_id`: 15,
`features` : { `on_jog_long_edge`:, true `on_jog_short_edge`: false, `on_horizontal_edge`: false, `on_vertical_edge`: true, ... }, `result`: $-C$}

Self-improvement System Upon generating the decision tree, we can rank the features by importance and perform importance-based feature selection. As illustrated in Figure 6, the features `on jog long edge` and `face convex corner` rank the lowest, with a feature importance of 0. By removing the unimportant features and recycling them back into the LLM's input, we update the feature pool and continue constructing new decision trees. This iterative process is repeated several times, enhancing feature extraction and development, ultimately making the system self-improving.

This approach mirrors the workflow of human OPC engineers, who continuously experiment, validate results, and

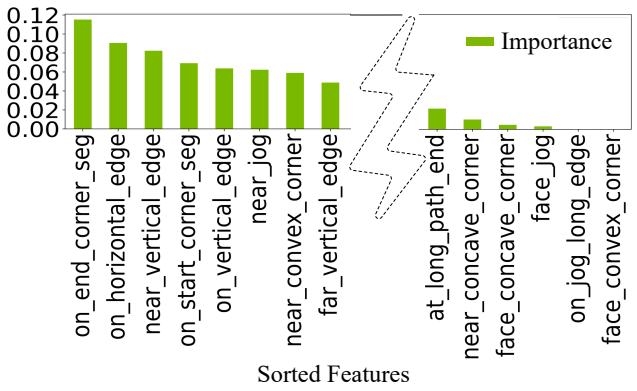
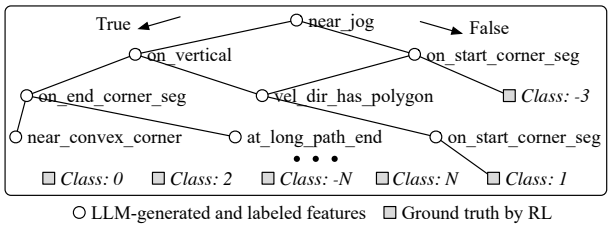


Figure 6: Feature importance ranking. The features are ranked based on their importance in the decision tree. The less important features are removed and fed back into the LLM for further improvement.

refine recipes. By automating this process into a symbolic improvement and generation system based on decision trees, we streamline and enhance the traditional OPC recipe development process. This automation not only accelerates the creation of optimized OPC recipes but also improves their accuracy and reliability, demonstrating the significant potential of integrating advanced computational techniques into the domain of computational lithography.

Recipe Generation Based on Decision Tree After annotating with the LLM, we can intuitively construct a decision tree for each pattern by combining the original polygon information with the extracted features. Based on the results from the first RL phase, we label each leaf node of the decision tree with the corresponding recipe type. Once the decision tree is trained, each branch serves as a reference for the LLM to write the corresponding rules, ultimately generating a complete OPC recipe, as shown in Figure 7(a) and Figure 7(b). Additionally, using the decision tree as an intermediate representation has the benefit that the LLM can, after abstracting the decision tree, generate different recipe expressions according to the syntax of various downstream OPC software, as shown in Figure 7(c).



(a) Decision tree example. The leaf nodes of the decision tree are labeled with ground truth based on RL results. The non-leaf nodes are features generated and labeled by the LLM.

```
...{condition: [near_jog, on_vertical, not
vel_dir_has_polygon, not on_start_corner
_seg ], type: EPE, class: 1} {condition: [not
near_jog, not on_start_corner_seg ], type:
EPE, class: 3} {condition: [near_convex
_corner, next_to_concave_corner ], type:
FRAG, class: -2} ...
```

(b) LLM-generated recipe example in `jsonl` format. Conditions are tree feature labels, and the type determines the tasks. The class represents the RL ground truth.

```
NEWTAG edge A short_len corner1 convex
corner2 convex -out line_end
NEWTAG neighbor both line_end short_len
corner convex -out line_end_adj
fragment_corner A convex concave mid
_length 0.03
fragment_corner convex concave long
_lenght 0.04 breakinhalf
retarget_layer A pattern0 curve0
pattern_epe curve1 emulate
```

(c) Downstream OPC software recipe example also includes statements for defining feature labels and movement distances.

Figure 7: Decision tree and recipe generation example. (a) Decision tree constructed from LLM-labeled features and RL ground truth. (b) Simplified LLM-generated recipe example in `jsonl` format. (c) Downstream OPC (Mentor Graphics, Siemens 2024) software recipe example in Tcl language.

4 Experimental Results

4.1 Dataset

To evaluate the effectiveness of our framework, we utilized datasets from two distinct processes. The first dataset is derived from the 2013 ICCAD contest (Banerjee, Li, and Nassif 2013), which includes a test set of ten $2\mu m \times 2\mu m$ metal layer patterns fabricated using a 32nm process. This dataset is widely employed in various OPC engines and semiconductor lithography research. We utilized the training set provided by (Yang et al. 2020) for our experiments. The second dataset is sourced from the NVIDIA Deep Learning Accelerator (NVDLA) (NVIDIA 2024), an open architecture designed to standardize deep learning inference accelerators. From the full-chip layout of the NVDLA, fabricated using NanGate $45nm$ standard cells (Stine et al. 2007), we extracted nearly one million clips. We then randomly selected 800 clips for the training set and 200 clips for the test set. This diverse dataset collection enabled a comprehensive evaluation of our OPC recipe development and its application within computational lithography.

4.2 Model and OPC Engine

In this study, we utilize GPT-4o (OpenAI 2024), an optimized version of GPT-4 with multi-modal capabilities that process both text and images, enhancing performance and versatility. The feature labeling component relies on these multi-modal capabilities, while other parts of the framework can use a purely language-based model. The OPC model is

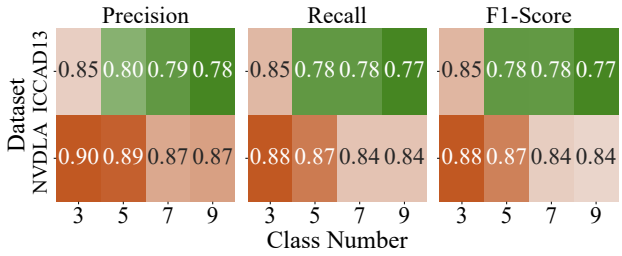


Figure 8: Decision Tree Efficiency on both ICCAD13 and NVDLA datasets.

built on top of the open-source OPC engine (Zheng et al. 2023), which is widely used in the OPC community. The methodology can be applied to any OPC engine. Implementation details and prompt scripts are provided in the supplementary material. The hyperparameters of the OPC loss are set to $\alpha = 1$, $\beta = 100$, and $\gamma = 1$.

4.3 Decision Tree Efficiency

As a critical basis for generating recipes using LLMs, decision tree models indirectly influence the final OPC performance. In our study, we demonstrate the precision, recall, and F1-score of the final decision tree generated for ICCAD13 and NVDLA datasets in Figure 8. The horizontal axis in Figure 8 represents the number of segments into which we divided the displacement distance, indicating the number of classes. The average precision ranges from 0.77 to 0.90, with some examples achieving a precision of 100%. This indicates that the decision tree constructed using features automatically mined by the LLM agent can achieve 100% accuracy on certain patterns. However, as the complexity and number of patterns increase, the overall average precision falls below 0.9. Despite this, for overall OPC effectiveness, a higher number of classes results in more precise outcomes. Therefore, in the recipe optimized by the LLM, we utilized 9 classes to compare the OPC results.

4.4 Efficiency of the Overall Framework

In Table 1, we present the results of three different scenarios: the pure OPC engine (Zheng et al. 2023), the OPC engine after first-stage RL optimization, and the OPC engine utilizing an LLM-generated recipe. The metrics PVB, EPE N, and EPE D are introduced before, with smaller values indicating better performance. The runtime is measured in seconds for the OPC engine and in hours for the RL optimization. In both the ICCAD13 and NVDLA benchmarks, RL optimization significantly reduces EPE values. Specifically, in the ICCAD13 benchmark, RL reduces the EPE number by 12% and the EPE distance by nearly 24%. Similarly, in the NVDLA benchmark, RL reduces the EPE number by 13% and the EPE distance by 15%. However, RL's major drawback is its excessive time consumption for pattern optimization, highlighting the necessity of the second stage of our framework.

The results of the second stage, shown in the OPC+LLM column of the table, indicate that the LLM-generated recipe performs comparably to RL optimization. For the NVDLA benchmark, the LLM-generated recipe reduces EPE N and EPE D by 8% and 12%, while for the ICCAD13 benchmark, it reduces EPE N by 11% and EPE D by 19%. The advantage of the rule-based LLM-generated recipe is its immediate applicability to new layouts without additional optimization time, maintaining the same runtime as the OPC engine itself while enhancing performance.

	ICCAD13		
	OPC	OPC+LLM	OPC+RL
PVBand	53328	51271	50060
ratio	1.00	0.96	0.94
EPE N	119.70	107.00	105.60
ratio	1.00	0.89	0.88
EPE D	693.10	561.70	525.90
ratio	1.00	0.81	0.76
Runtime	4s	4s	3hr
	NVDLA		
	OPC	OPC+LLM	OPC+RL
PVBand	170899	161056	162096
ratio	1.00	0.94	0.95
EPE N	159.45	146.95	139.40
ratio	1.00	0.92	0.87
EPE D	869.25	766.10	741.05
ratio	1.00	0.88	0.85
Runtime	6s	6s	3hr

Table 1: Performance of the framework. The RL stage results are shown in the OPC+RL column, and the results related to the final LLM-generated recipe are shown in the OPC+LLM column.

5 Conclusion

In this paper, we propose a two-stage framework to automate the task of OPC recipe development. In the first stage, RL mimics the process by which an OPC engineer designs a recipe. This involves exploring the optimal solutions for EPE measurement and fragmentation based on the characteristics of the patterns. In the second stage, LLMs automate the process of summarizing the recipe crafted by the OPC engineer. Utilizing the optimal solutions generated by RL, the LLM initially generates a relevant feature pool from the layout and subsequently annotates each point with its corresponding features. This annotated data is then used in conjunction with the RL results to construct a decision tree. Ultimately, the LLM retrieves this decision tree to generate a set of recipes. The experimental results show that this framework reduces the key error metric by more than 10% without increasing runtime.

References

Asthana, A.; Wilkinson, B.; and Power, D. 2016. OPC recipe optimization using genetic algorithm. In *Optical Microlithography XXIX*, volume 9780, 43–54. SPIE.

Banerjee, S.; Li, Z.; and Nassif, S. R. 2013. ICCAD-2013 CAD contest in mask optimization and benchmark suite. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 271–274.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 33: 1877–1901.

Chen, G.; Chen, W.; Ma, Y.; Yang, H.; and Yu, B. 2020. DAMO: Deep Agile Mask Optimization for Full Chip Scale. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Gilardi, F.; Alizadeh, M.; and Kubli, M. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30).

Granik, Y.; and Cobb, N. B. 2002. MEEF as a Matrix. In *Photomask*, 980–991.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Lei, J.; Hong, L.; Lippincott, G.; and Word, J. 2014. Model-based OPC using the MEEF matrix II. In *Optical Microlithography XXVII*, volume 9052, 170–178. SPIE.

Liang, X.; Ouyang, Y.; Yang, H.; Yu, B.; and Ma, Y. 2023. RL-OPC: Mask Optimization with Deep Reinforcement Learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Liang, X.; Yang, H.; Liu, K.; Yu, B.; and Ma, Y. 2024. CAMO: Correlation-Aware Mask Optimization with Modulated Reinforcement Learning. In *ACM/IEEE Design Automation Conference (DAC)*.

Liu, M.; Ene, T.-D.; Kirby, R.; Cheng, C.; Pinckney, N.; Liang, R.; Alben, J.; Anand, H.; Banerjee, S.; Bayraktaroglu, I.; et al. 2023. ChipNeMo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*.

Liu, Q.; and Zhang, L. 2010. Optimize the OPC control recipe with cost function. In *Photomask Technology 2010*, volume 7823. SPIE.

Mentor Graphics, Siemens. 2024. Calibre Computational Lithography. https://eda.sw.siemens.com/en-US/ic/calibre-manufacturing/computational-lithography/. Accessed: 2025-01-06.

NVIDIA. 2024. NVIDIA Deep Learning Accelerator. https://nvdla.org/index.html. Accessed: 2025-01-06.

OpenAI. 2024. Hello gpt-4o. https://openai.com/index/hello-gpt-4o. Accessed: 2025-01-06.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Stine, J. E.; Castellanos, I.; Wood, M.; Henson, J.; Love, F.; Davis, W. R.; Franzon, P. D.; Bucher, M.; Basavarajaiah, S.; Oh, J.; and Jenkal, R. 2007. FreePDK: An Open-Source Variation-Aware Design Kit. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 173–174.

Synopsys, Inc. 2024. Proteus Solutions. https://www.synopsys.com/manufacturing/mask-solutions/proteus.html. Accessed: 2025-01-06.

Tsai, Y.; Liu, M.; and Ren, H. 2023. RTLFixer: Automatically fixing rtl syntax errors with large language models. *arXiv preprint arXiv:2311.16543*.

Wu, H.; He, Z.; Zhang, X.; Yao, X.; Zheng, S.; Zheng, H.; and Yu, B. 2024. ChatEDA: A large language model powered autonomous agent for eda. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Wu, L.; Kwa, D.; Wan, J.; Wang, T.; John, M. S.; Deeth, S.; Chen, X.; Cecil, T.; Meng, X.; and Lucas, K. 2016. Building block style recipes for productivity improvement in OPC, RET and ILT flows. In *Design-Process-Technology Co-optimization for Manufacturability X*, volume 9781. SPIE.

Yang, H.; Li, S.; Deng, Z.; Ma, Y.; Yu, B.; and Young, E. F. Y. 2020. GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*.

Yang, H.; and Ren, H. 2024. ILILT: Implicit Learning of Inverse Lithography Technologies. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 56319–56331. PMLR.

Zheng, S.; Ma, Y.; Zhu, B.; Chen, G.; Zhao, W.; Yin, S.; Yu, Z.; and Yu, B. 2023. OpenILT: An Open-source Platform for Inverse Lithography Technique Research. https://github.com/OpenOPC/OpenILT/. Accessed: 2025-01-06.

Zhu, B.; Zheng, S.; Yu, Z.; Chen, G.; Ma, Y.; Yang, F.; Yu, B.; and Wong, M. 2023a. L2O-ILT: Learning to Optimize Inverse Lithography Techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Zhu, Z.; Xue, Y.; Chen, X.; Zhou, D.; Tang, J.; Schuurmans, D.; and Dai, H. 2023b. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*.