Efficient ILT via Multigrid-Schwartz Method

Shuyuan Sun[1,2], Fan Yang[1,2*], Bei Yu[4], Li Shang[1,3], Dian Zhou[5], Xuan Zeng[1,2]
[1]State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai, China
[2] School of Microelectronics, Fudan University, Shanghai, China
[3]China and Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai, China
[4]Chinese University of Hong Kong, Hong Kong, China
[5]University of Texas at Dallas, Texas, USA

Abstract

Inverse Lithography Technology (ILT) is an important Resolution Enhancement Technology (RET) in chip manufacturing. Due to the high computational demands of ILT, large-scale layouts are typically partitioned into smaller tiles for independent processing. In this paper, we propose a multigrid-Schwarz method to overcome challenges in tile assembly. Experimental results show that our approach achieves comparable performance to the full-chip ILT, offering increased parallelizability and speedup in parallel mode. Unlike the traditional divide-and-conquer algorithm, it effectively alleviates discontinuities of tile stitching, preventing manufacturing failures.

1 Introduction

As feature sizes continue to shrink while the wavelength of light sources remains unchanged, achieving advanced processes becomes increasingly challenging, demanding more powerful Resolution Enhancement Techniques (RETs). In this context, Inverse Lithography Technology (ILT) is gaining more attention as one of the most crucial RETs. Given the intensive computational demands of full-chip ILT, a prevalent approach involves partitioning the large-scale layout into smaller tiles. These tiles are individually optimized and subsequently assembled to obtain a correction solution for the entire layout.

Considerable efforts have been devoted to enhancing the quality and efficiency of mask optimization on individual tiles [1–4], whereas research on full-chip mask optimization remains relatively limited. Due to the non-uniqueness of ILT results, where different mask graphics can yield the same lithographic output, there is no assurance that graphics on both sides of the boundary will converge to the same position. This discrepancy may lead to significant mismatches and suboptimal solutions at the boundaries. Furthermore, as ILT simultaneously optimizes both sub-resolution assist features (SRAF) and main features, the mismatch issue is further exacerbated, as depicted in fig. 1. Hence, tile assembly poses the most significant challenge for full-chip mask optimization.

Various methods have been proposed to tackle boundary mismatching. In [5], the error norm is calculated within the overlapping region of tiles and the final results are selected based on the error. In [6], a 'stitch-and-heal' methodology is proposed to re-optimize the region near the stitching boundary. However, it also introduces new partition boundaries, potentially leading to new mismatching errors at boundaries. Additionally, they both requires additional rounds of ILT, substantially extending the runtime. In [7] and [8], a pattern library is constructed to improve runtime efficiency and graphics consistency. However, when dealing with metal layers containing many complex shapes, extracting template patterns becomes notably challenging. In [9], a dedicated ILT computing device is constructed, utilizing thousands of GPU/CPU pairs to emulate a single giant GPU/CPU pair. In [10], a pipelined approach for ILT is proposed, enabling the achievement of full-chip ILT using limited resources. However, such a configuration has a high computation cost. The communication between GPUs poses significant engineering challenges, introducing a multitude of complexities.

*Corresponding author: yangfan@fudan.edu.cn.

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3657309

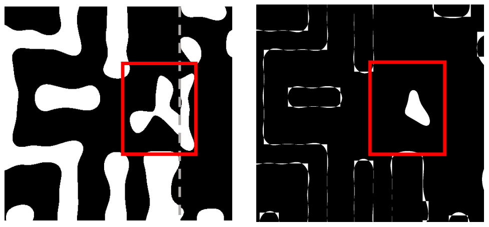
(a) Mask (b) Litho differ from target

Figure 1: Severe mismatch on the boundary.

The goal of this work is to tackle the aforementioned challenge in large-scale ILT, to achieve correction results with seamless continuity in a cost-effective manner while retaining a certain degree of parallelism. To this end, the proposed ILT framework comprises a multigrid method for performing ILT at multiple scales, a modified additive Schwarz method for addressing boundary mismatches, and a multi-color multiplicative Schwarz method for refining the final results. Specifically:

Parallelism and runtime efficiency. Inspired by the multigrid method [11], computing an approximate solution on a coarse grid facilitates communication between tiles without compromising parallelism. In our proposed method, coarse-grid ILT is independently and parallelly executed. The coarse grid covers a larger area and results in fewer partitioned tiles from the layout, enabling a faster generation of an initial solution and reducing tile stitching. Therefore, the multigrid method not only accelerates the process, but also mitigates mismatches at tile boundaries. Following the coarse-grid ILT, a fine-grid ILT is utilized to achieve more precise correction solutions. We employ a modified Additive Schwartz method [12] to break down the entire fine-grid ILT process into several stages. At the end of each stage, communication takes place between adjacent tiles to avoid mismatches at the boundaries. At last, we employ a multi-color multiplicative Schwartz method to refine the results. Through a limited number of tile-to-tile communications, we can maintain a certain degree of parallelism in a cost-effective manner.

Continuity of boundaries. Bad continuity of tile boundaries can lead to manufacturing difficulties. First, we introduce a metric to quantitatively evaluate the continuity of stitching boundary. Then, we propose to combine the Schwartz method [12] with a weighted smoothing algorithm [13] from the field of image stitching to improve the continuity of boundaries. Experimental results show that our methods greatly improve the mask continuity on our designed metric by more than 3.15×, and is comparable to the results obtained by the expensive full-chip ILT.

The rest of the paper is organized as follows. In Section 2, we present the background of lithography simulation and give the problem formulation. In Section 3, we propose the multigrid-Schwartz ILT method for boundary correction. In Section 4, experimental results are presented. In Section 5, we conclude the paper.

2 Background

2.1 Lithography Model

The transformation from mask image $\boldsymbol{M} \in \mathbb{R}^{H \times W}$ to wafer image $\boldsymbol{Z}$ consists two models: the lithography model and the photoresist model. The lithography model transforms the mask image $\boldsymbol{M}$ to the aerial image $\boldsymbol{I}$, which can be approximated by the Hopkin's model [14] as shown in Equation (1).

$$\boldsymbol{I}(x,y) \approx \sum_{i=1}^{N_i} w_i \|\boldsymbol{h}_i(x,y) \otimes \boldsymbol{M}(x,y)\|^2, \tag{1}$$

where $\boldsymbol{h}_i \in \mathbb{R}^{P \times P}$ represents the i^{th} optical kernel, and w_i is the corresponding weight. N_i is the total number of kernels used for simulation. $\otimes$ denotes the convolution operation. It is proved that convolution in the spatial domain is equivalent to multiplication in the frequency domain. Therefore, computational efficiency can be further improved by using the fast Fourier transformation (FFT), as shown in Equation (2).

$$\boldsymbol{h}_i(x,y) \otimes \boldsymbol{M}(x,y) = \mathcal{F}_N^{-1}\left(\boldsymbol{H}_i \odot [\mathcal{F}_N(\boldsymbol{M})]_P\right), \tag{2}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent FFT and inverse FFT, respectively. $\boldsymbol{H}_i$ is the representation of $\boldsymbol{h}_i$ in the frequency domain and is also referred to as the transmission cross-coefficient (TCC) kernel. $[\cdot]_P$ represents the extraction of the low-frequency portion of size $P \times P$ from a FFT spectrum of size $N \times N$.

Our litho-simulation tool, obtained from the ICCAD 2013 competition [15], is designed for layouts of size $N \times N$, where $N = 2048$. The FFT spectrum corresponding to the provided TCC kernel $\boldsymbol{H}_i$ is sampled with a minimum interval of $1/N$. While convolution does not impose restrictions on the size of the input image, we cannot directly apply $\boldsymbol{H}_i$ to larger layouts. Based on the convolution theorem, we derived Equation (3) from Equation (2) to calculate the aerial image for a larger layout. To obtain the frequency corresponding to $\boldsymbol{H}_i$, the size of the convolved image should be multiples of N, i.e., $\boldsymbol{M} \in \mathbb{R}^{sN \times sN}$, where s is an integer factor.

$$\boldsymbol{h}_i(x,y) \otimes \boldsymbol{M}(x,y) = \mathcal{F}_{sN}^{-1}\left(\boldsymbol{H}_i\left(\frac{j}{s}, \frac{k}{s}\right) \odot [\mathcal{F}_{sN}(\boldsymbol{M})]_{sP}\right). \tag{3}$$

The photoresist model transforms the aerial image $\boldsymbol{I}$ into the wafer image $\boldsymbol{Z}$. For simplicity, we use a compact photoresist model with acontant threshold, consistent with [15]. We use $f(\cdot)$ to represent all these computations from mask image $\boldsymbol{M}$ to wafer image $\boldsymbol{Z}$, as shown in

$$f(\boldsymbol{M}) = \boldsymbol{Z}. \tag{4}$$

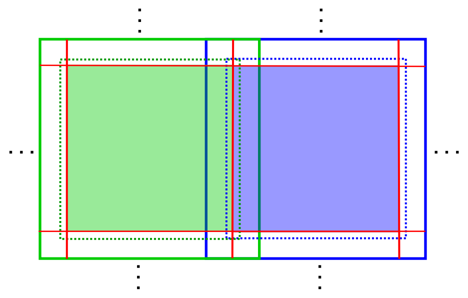

Figure 2: The green and blue lines outline the boundaries of two adjacent tiles, while the red lines outline the shared boundaries between the core sections. The filled region represents the core sections of tiles.

And the inverse lithography problem can be regarded as to find the optimal mask $\boldsymbol{M}^*$. After photolithography processing $f(\cdot)$, the obtained image should be aligned with the target image $\boldsymbol{Z}_t$ as much as possible. The mask optimization problem is stated in Equation (5).

$$\boldsymbol{M}^* = f^{-1}(\boldsymbol{Z}_t), \tag{5}$$

where $\boldsymbol{Z}_t$ represents the target image and $\boldsymbol{M}^*$ represents the optimal mask image we want to get.

2.2 Tile partitioning and Domain Decomposition

Direct solve Equation (5) is excessively complex. The traditional divide-and-conquer method involves partitioning the layout $\boldsymbol{M}$ into J overlapping tiles $\boldsymbol{M}_j$, as shown in Fig. 2. This tile partition method is similar to the domain decomposition (DD) methods that employed by restricted additive Schwartz (RAS) preconditioners [12, 16]. Both of them solve the problem in the overlapping subdomains separately, and gather the solutions from non-overlapping subdomains to obtain the final results. In ILT, we refer the non-overlapping subdomains $\tilde{\boldsymbol{M}}_j$ as 'core' sections, the overlapping subdomains $\boldsymbol{M}_j$ as 'tiles', and the remaining part $\boldsymbol{M}_j - \tilde{\boldsymbol{M}}_j$ as 'margin' sections.

After introducing tile partition, we adapt the RAS method to restate Equation (5) as Equation (6).

$$\boldsymbol{M^*} = \sum_{j=1}^{J} \tilde{\boldsymbol{R}}_j^{\top} f^{-1}(\boldsymbol{Z}_t \boldsymbol{R}_j), \tag{6}$$

where $\boldsymbol{R}_j$ is the restriction operator, acting on each overlapping subdomain $\boldsymbol{M}_j$, and $\tilde{\boldsymbol{R}}_j^{\top}$ is the interpolation operator, used to put each non-overlapping subdomain back to its corresponding position in the layout. It is proved in [17] that RAS methods converge faster than additive Schwartz (AS) methods.

2.3 Evaluation Metrics and Problem Formation

Despite its advantages, tile partitioning introduces two notable challenges. Firstly, it potentially lead to sub-optimal optimization results. The mask image obtained through single-tile ILT may differ from the one cropped from the assembled mask. This discrepancy arises because the margin section of one tile is influenced by its adjacent tiles. We conducted experiments using two ILT solvers [3, 4], tile assembly resulted in up to a 8247 and 4600 increase in $L2$ error, respectively. While the increased L2 error may not appear substantial, in cases such as the example depicted in Fig. 1, it can lead to critical errors, resulting in circuit function failures.

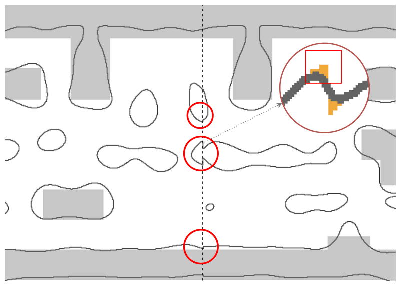

Figure 3: Using the traditional approach, stitchig errors manifest during the partitioning. A zoomed-in view highlights one stitching error, the area covered in orange represents the *stitch_loss*, which we use to evaluate the continuity of stitched graphics.

Secondly, since the ILT results in adjacent tiles are independently obtained, there will be discontinuities near the shared boundaries of core sections, such discontinuities can violate the manufacturability rule check (MRC), as shown in Fig. 3.

Definition 1 (*Stitch Loss*). *To assess the continuity of graphics near the boundaries, we employs multiple iterations of Gaussian low-pass filtering to smooth the shape contours. Then, we extract coordinates where graphics intersect with the stitching line. For each intersection, we extract a 40×40 window centered around it and compute the pixel differences within this window before and after smoothing. We take the sum of pixel differences as the Stitch Loss to evaluate the continuity of the stitched graphics. As illustrated in Fig. 3, a zoomed-in view highlights one stitching error. The area covered in orange represents the Stitch Loss.*

Definition 2 (*L2 loss*). *L2 loss is calculated by* $\|Z - Z_t\|_2^2$, *where* Z *represents the wafer image generated under the nominal dose and nominal focus.*

Definition 3 (*PVBand*). *Process variation band (PVBand) is calculated by* $\|Z_{in} - Z_{out}\|_2^2$. Z_{in} *is the wafer image generated under the defocus and* -2% *dose condition, and* Z_{out} *is the wafer image generated under the nominal focus and* $+2\%$ *dose condition.*

Problem 1 (Mask Optimization). *Given the target layout* Z_t, *our goal is to achieve a fully optimized mask* $\boldsymbol{M}$ *within an acceptable runtime, aiming to enhance the parallelism of ILT and ensure consistency of layout shapes at stitching lines.*

3 Algorithms

3.1 Overall Framework

The overall framework of multigrid-Schwartz ILT is illustrated in this section, as shown in Section 3.1. We perform coarse-grid ILT, fine-grid ILT and refine ILT in sequence, and the correction area of a single ILT changes from large to small. The coarse-grid ILT is employed to obtain approximate solutions that consider more global informations and expedite the ILT process. The fine-grid ILT is utilized to achieve a higher quality correction solution and fix boundary mismatch issues. The refine ILT is adopted to further improve the ILT solution via enabling more frequent information exchange between tiles. To preserving acceptable parallelism, we implement a multi-color Schwartz method for the 'refine ILT'. We detail the coarse-grid, fine-grid and refine ILT in Section 3.2, Section 3.3 and Section 3.4, resspectively.

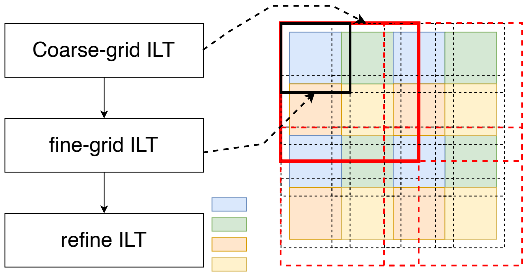

Figure 4: The overall framework.Coarse-grid ILT operates over larger areas to obtain an approximate solution for the entire chip. In contrast, fine-grid ILT operates without scaling and aims to achieve higher-quality solutions while addressing mismatches near the boundaries. Fine-grid ILT facilitates more frequent communication between tiles, yet it can be executed in parallel on tiles with the same color to preserve a certain level of parallelism.

3.2 Multigrid ILT

In our proposed framework, our first step is to determine the maximum grid size whose ILT output contributes positively to the final result. After this, we systematically shrink the correction grid to gradually approach the optimal outcome for the entire layout. In Section 3.1 we show a partition strategy for a two-grid ILT. Note that the coarse grid may not necessarily contain an integer number of finer grids, but its size must be an integer multiple of N, where N is the input size of the original lithosimulator (lines 6 in Algorithm 1). The scaling factor for coarse-grid ILT is greater than 1, and for fine-grid ILT, is equal to 1.

For each grid size, we perform tile partitioning using the method illustrated in Fig. 2. Subsequently, we downsample the obtained tiles to fit within a single GPU, and apply the existing single-tile ILT algorithm $\varphi(\cdot)$ for mask optimization (lines 9–10 in Algorithm 1). Finally, after completing ILT on all tiles of one grid, the correction results are assembled. Note that stitching errors are not addressed during coarse-grid ILT. Instead, they are corrected in the fine-grid ILT, which results in a different assembly equation (line 12 in Algorithm 1). By employing Equation (3), it is capable to compute the

Algorithm 1 Multigrid ILT

```
1:  Input : Target image $Z_t \in \mathbb{R}^{H\times W}$; the biggest scale factor $s_{max}$; single-tile ILT solver $\varphi(\cdot)$; the overlapping length between tiles $2l$; the tile partition strategy $Part(\cdot)$.
2:  Output : optimized mask $\boldsymbol{M}$.
3:  $s \leftarrow s_{max}$;
4:  $\boldsymbol{M} \leftarrow Z_t$;
5:  repeat
6:      $\{(Z_j, \boldsymbol{M}_j)\} \leftarrow Part(Z_t, \boldsymbol{M}, \text{tile\_size}=sN, \text{margin}=l)$;
7:      for $j = 1...J$ do
8:          $Z_{j,s} = \text{Downsample}(Z_j, \text{factor} = s)$;
9:          $\boldsymbol{M}_{j,s} = \text{Downsample}(\boldsymbol{M}_j, \text{factor} = s)$;
10:         $\boldsymbol{M}_{j,s} = \varphi(Z_{j,s}, \boldsymbol{M}_{j,s})$;
11:     end for
12:     Assemble tiles using Equation (6) or Equation (14);
13:     s = s / 2;
14: until $s < 1$.
```

optical image for layouts of diverse sizes. Nevertheless, the capacity

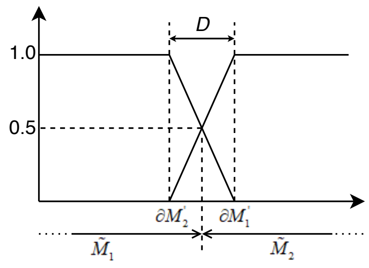

Figure 5: The weighted smoothing algorithm.

to simulate the layout area in a single iteration is limited by the available GPU memory size. Therefore, downsampling is employed to reduce the mask size, facilitating a more expansive field of view in an economical manner. In Equation (7), the relationship between spacial and frequency domain is stated:

$$\mathcal{F}_{sN}\left(\boldsymbol{M}\left(sx, sy\right)\right) = \frac{1}{s^2}\mathcal{F}_{sN}\left(\boldsymbol{M}\right)\left(\frac{j}{s}, \frac{k}{s}\right). \tag{7}$$

Exchanging the left and right sides of Equation (7) and applying a coordinate transformation yield the first line of Equation (8).

$$\begin{aligned} \mathcal{F}_{sN}\left(\boldsymbol{M}\right) &\approx s^2\mathcal{F}_{sN}\left(\boldsymbol{M}\left(sx, sy\right)\right)\left(sj, sk\right) \\ &\approx \mathcal{F}_{N}\left(\boldsymbol{M}\left(sx, sy\right)\right). \end{aligned} \tag{8}$$

The minimum interval of the FFT spectrum $\mathcal{F}_{sN}$ is $1/sN$, and the minimum interval of $\mathcal{F}_N$ is $1/N$. Every s points in $\mathcal{F}_{sN}$ corresponding one point in $\mathcal{F}_N$, yielding the second line of Equation (8).

Using the frequency transformation stated in Equation (8), we obtain Equation (9) based on Equation (3).

$$\boldsymbol{I}_i\left(sx, sy\right) \approx \mathcal{F}_N^{-1}\left(\boldsymbol{H}_i\left(\frac{j}{s}, \frac{k}{s}\right) \odot \left[\mathcal{F}_N\left(\boldsymbol{M}_s\right)\right]_{sP}\right), \tag{9}$$

where $\boldsymbol{M}_s$ represents the reduced mask. The update of mask image in coarse-grid ILT is guided by lithography results calculated using Equation (9) and Equation (2).

Note that downsampling inevitably results in information loss. Consequently, the solution obtained by coarse-grid ILT is more comprehensive in scope but less precise in accuracy. Therefore, we introduce fine-grid ILT in Section 3.3 to address stitching errors and achieve superior quality results.

3.3 Modified Schwartz Method

In Section 2.3, it was demonstrated that the traditional divide-and-conquer method yields suboptimal final optimization results due to the absence of communication between tiles. Inspired by the RAS method, we divide the ILT process into several stages. In each stage, ILT is performed independently on tile $\boldsymbol{M}_j$ to solve Equation (10).

$$f\left(\boldsymbol{u}_j\right) = Z_t\boldsymbol{R}_j, \text{ in } \boldsymbol{M}_j, \tag{10}$$

where f represents the lithography process, and $\boldsymbol{u}_j$ represents the solution being sought on tile $\boldsymbol{M}_j$. After each stage finished, we stitch all tiles using Equation (6) to update the entire layout. The next stage of ILT uses tiles cropped from the assembled layout, so that the margin section of tiles can be updated by adjacent tiles.

The RAS method places constraints on the boundary values of overlapping subdomains, ensuring their equivalence to the values in adjacent subdomains. It promotes uniform values at the joint positions of non-overlapping subdomains. Inspired by RAS method, we modified its boundary condition and make it suitable for ILT.

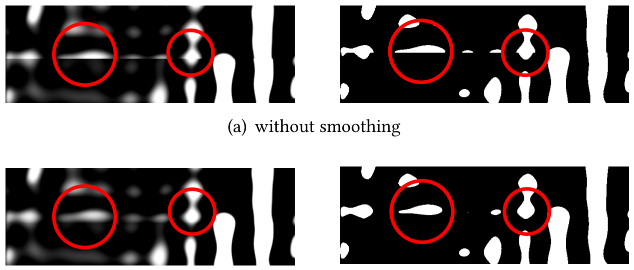

Figure 6: Weighted smoothing improves the continuity of graphics.

From Equation (1), it is evident that lithography results obtained from individual tiles near their boundaries will inevitably deviate from the results obtained through simulations for the entire chip. Hence, the boundary conditions are set in proximity to the core sections, as the ILT results in these regions are derived from accurate lithography simulations. Equation (11) is the boundary condition we modified from the RAS method.

$$\boldsymbol{u}_j = \boldsymbol{M}_l, \text{ on } \partial\boldsymbol{M}_j' \cap \boldsymbol{M}_l, l \in N_j, \tag{11}$$

where N_j is the index set of adjacent tiles to tile $\boldsymbol{M}_j$, and $\partial\boldsymbol{M}_j'$ is the overlapping boundaries that near the core section. In Fig. 2, we use dashed lines to represent $\partial\boldsymbol{M}_j'$.

However, given that the correlation between pixels is not so strong in ILT, using boundary conditions directly can still lead to boudaries discontinuities. Therefore, we incorporate the weighted smoothing algorithm [13] to ensure a seamless stitching between tiles. The idea behind the algorithm is to set up a buffer area that gradually transit the patterns to the location where the boundary conditions are set. In Equation (12) and Fig. 5, we show the weighted stitching strategy of two adjacent tiles.

$$M_{j,l} = w_j\boldsymbol{M}_j' + w_l\boldsymbol{M}_l', l \in N_j, \tag{12}$$

where $M_{j,l}$ represents the merge result of tile $\boldsymbol{M}_j$ and tile $\boldsymbol{M}_l$.

$$w_j(d_j) = \begin{cases} 1, d_j > D \\ 1 - \dfrac{d_j}{D}, d_j \leqslant D, \end{cases} \tag{13}$$

where D is the overlapping length between $\boldsymbol{M}_j'$ and $\boldsymbol{M}_l'$, and d_j is the distance from $\partial\boldsymbol{M}_j'$. The weighted smoothing can effectively remove the stitching seams and improves the continuities, as shown in Fig. 6. In Fig. 6, the left column consists of mask images before binarization, while the images in the right column are masks after binarization.

We modified Equation (6) to incorporate the weighted smoothing algorithm into the tile assembly process, as illustrated in Equation (14).

$$\boldsymbol{M}^* = \sum_{j=1}^{J}\boldsymbol{R'}_j^{\top}f^{-1}\left(Z_t\boldsymbol{R}_j\right), \tag{14}$$

where $\boldsymbol{R'}_j^{\top}$ is the weighted interpolation operator.

3.4 Multi-color Schwartz Method

In this section, we introduce the multi-color multiplicative Schwarz method [18] for the 'refine ILT', aiming to achieve a better optimization result. The learning rate of the 'refine ILT' is relatively small, implying that it will only make small adjustments. For scenarios where shapes near the boundaries exhibit strong mutual dependencies but are split across two tiles, relying solely on the fine-grid ILT is insufficient. Therefore, 'refine ILT' enable a more frequent communication between tiles. Note that 'refine ILT' is executed only a few times, so will not consume too much time.

The multi-color multiplicative Schwarz method employs a tile coloring scheme that guarantees non-overlapping regions do not share the same color, as depicted in Section 3.1. ILT can be performed in parallel on tiles that share the same color, with fixing patterns on tiles of other colors. In this way, we obtain a certain level of parallelism.

4 Experimental Results

All experiments are performed on a Linux workstation with a 2.6GHz Intel Xeon CPU and Nvidia RTX A6000 GPU (51G). The lithography simulation tool was obtained from the ICCAD 2013 competition [15], which we reimplemented in Pytorch and modified to simulate a larger area. For the via layer, we find that the method of extracting template patterns is more suitable. Therefore, we conducted experiments only on the M1 layer. We conduct experiments on 20 layout clips of size 4096×4096. This is the maximum size of a layout that our GPU can handle and optimize via ILT at once. Using the partition strategy illustrated in Fig. 2, a 4096×4096 layout can be divided into 9 tiles of size 2048×2048, with an overlap length of 2×512 between tiles.

First, we conduct experiments on a single GPU. The initial set of experiments involved employing the traditional divide-and-conquer approach, where ILT was performed independently on each tile before assembling the final correction result. Two single-tile ILT approaches, namely "Multi-level-ILT" [4] and "GLS-ILT" [3], are utilized for testing, and their results are presented in the first two columns of TABLE 1. The maximum number of iterations for both methods is set to 100 on a single tile. From the experimental results, we observe that the mask obtained by Multi-level ILT is superior to GLS-ILT in quality, with a reduction of 17.2% in L2 loss and 2.4% in PVBand. However, it exhibits a noticeable degradation in *Stitch Loss* compared to GLS-ILT. By observing the correction results, we find that it is mainly because Multi-level ILT produces more sub-resolution assist features (SRAFs) compared to GLS-ILT. The increased presence of SRAFs makes Multi-level ILT more susceptible to mismatches near the boundaries, although it helps to achieve better quality of masks.

We evaluate our proposed method on the same dataset. First, we perform 60 iterations of coarse-grid ILT with a scale factor of $s = 2$. Subsequently, we execute 40 iterations of fine-grid ILT. Note that we divide these 40 iterations of fine-grid ILT into two stages. After each stage concludes, we assemble the tiles and apply Schwartz boundary conditions to smooth the boundaries. Finally, we conduct 4 iterations of refine ILT to further improve the results. It is shown that our method can effectively reduce the *Stitch Loss* by more than 3.15×, as shown in column "Ours" of TABLE 1.

Additionally, we run 100 iterations of Multi-level ILT on the entire 4096×4096 region without tile partitioning to emulate full-chip ILT. It is referred to as "Full-chip ILT" in TABLE 1. The runtime of "Full-chip ILT" is calculated under ideal conditions without considering the communication overhead. In reality, no GPU can accommodate an entire layout at once, and there will inevitably be time overhead for CPU-GPU communication. It is shown that our method deliver comparable optimization results to the full-chip ILT. Note that the

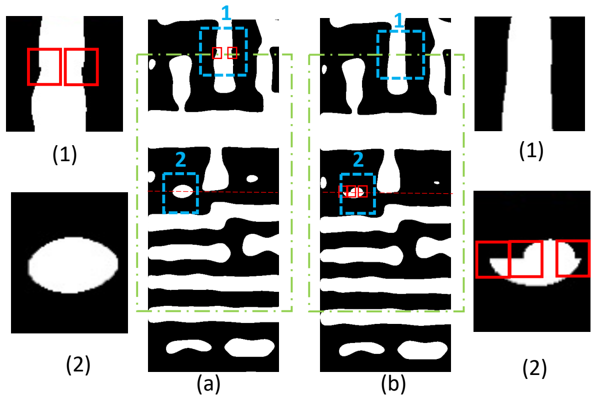

Figure 7: Stitching errors persistently appear at the newly created partition boundaries when applying the 'stitch-and-heal' method for reoptimizing the boundary region. The green box designates the re-optimized region, and the red boxes highlight areas having stitching errors. The red line represents the original stitching line. (a) presents the outcome following the re-optimization process, while (b) illustrates the result before re-optimization.

final inspection of all methods utilizes lithography results calculated using Equation (3), which is computed over the entire 4096×4096 region without tile partitioning.

Fig. 8(a) presents a segment generated by the traditional divide-and-conquer method Fig. 8(b) is the corresponding one generated by our proposed method. The red boxes in Fig. 8 indicate locations where *stitch error* exceed to 20. It is shown that our method can greatly improve the continuity of shapes near boundaries. Fig. 7 shows that the 'stich-and-heal' method cannot fully fix stitching mismatches, since it will introduce new partition edges.

For parallelism testing, our method can obtain 2.76× speedup using 4 GPUs. Note that our GPUs lack of direct connections between GPUs. The data movements between GPUs are implemented by moving data from one GPU to main memory and then to other GPUs. This may degrade the speedups for multiple GPUs. We believe the speedup can be more significant for multiple GPUs with direct connections (e.g. NVLink).

5 Conclusion

In this paper, a multigrid-Schwarz method is proposed to fix the stitch errors in tile assembly for ILT. The coarse grid covers a larger area and results in fewer tiles, enabling a faster generation of an initial solution for the entire chip and reducing tile stitching. We employ a modified Additive Schwartz method to break down the entire fine-grid ILT process into several stages. At the end of each stage, communication takes place between adjacent tiles. At last, we employ a multi-color multiplicative Schwartz method to refine the results. Experimental results show that our approach achieves comparable performance to the full-chip ILT, with significant speedup.

Acknowledgement

This research is supported partly by National Key R&D Program of China 2020YFA0711900, 2020YFA0711903, partly by the National Natural Science Foundation of China (NSFC) Research Projects 92373207 and 62090025, and The Research Grants Council of Hong Kong SAR (No. CUHK14208021).

Table 1: Comparison Results on 20 clips of size 4096×4096.

Benchmarks		GLS-ILT[3]				Multi-level-ILT[4]				Full-chip ILT				Ours			
ID	Area (nm^2)	*L2* (nm^2)	*PVB* (nm^2)	*Stitch loss* (nm^2)	TAT (s)	*L2* (nm^2)	*PVB* (nm^2)	*Stitch loss* (nm^2)	TAT (s)	*L2* (nm^2)	*PVB* (nm^2)	*Stitch loss* (nm^2)	TAT (s)	*L2* (nm^2)	*PVB* (nm^2)	*Stitch loss* (nm^2)	TAT (s)
case1	6262100	219268	402286	1204	65.03	237755	386326	2981	51.38	181913	390722	314	23.72	182645	385150	531	24.82
case2	5918645	184657	340870	447	59.93	142855	332182	1838	49.9	140922	329462	850	22.21	142490	327510	730	23.39
case3	6756478	246070	431612	201	66.97	221238	415799	1416	50.01	217571	417265	537	22.35	206092	416471	663	23.74
case4	6123117	195994	358750	1531	61.93	148811	350574	1734	49.98	147936	349684	625	23.84	148048	344276	565	23.37
case5	6895298	249377	425535	54	71.5	216506	420951	2503	50.03	209646	422329	523	22.78	207121	415247	766	23.61
case6	5889573	214195	390092	1375	62.93	167343	385179	1642	50.33	165749	387656	544	23	164324	382587	586	23.24
case7	6800017	231536	421374	273	67.09	208338	410444	1526	50.21	185919	410577	272	22.64	191293	407135	432	23.64
case8	6020502	210737	401681	177	64.28	173141	389268	1143	50.24	167485	393540	741	22.17	174227	386242	494	23.49
case9	7130023	223450	390594	1661	64.43	196909	385404	1261	50.3	188529	384135	445	22.74	189533	378575	325	23.31
case10	5910538	218678	415004	282	65.58	173957	403896	1679	50.01	165624	406165	439	22.38	170782	403409	521	23.55
case11	7238614	210456	362883	188	62.91	184013	355696	1613	50.13	177472	354928	549	22.16	179766	351473	388	23.42
case12	5833915	205383	400822	130	65.36	160615	390236	2258	49.88	157227	391330	598	22.57	155773	384810	757	23.33
case13	7009032	220800	399063	912	63.45	197219	389424	1550	50.09	184835	388272	619	22.44	189378	384253	434	23.45
case14	5950136	221100	424911	65	64.96	198236	422361	1408	49.92	163762	426933	400	22.24	172493	418160	288	23.58
case15	6953312	203975	364550	101	62.74	178465	348940	1796	50	168388	351909	716	22.2	172072	347410	535	23.75
case16	5866510	206103	393307	120	63.93	166358	379214	2100	49.95	170228	379136	832	22.14	160948	375454	839	23.54
case17	7135150	207945	350006	465	63.42	180358	338597	1996	50.09	174924	339134	696	22.19	172086	335275	726	23.44
case18	6148328	213558	399418	1221	61.97	171232	391718	2006	49.91	167363	392856	712	22.54	168484	388702	774	23.52
case19	7263044	211507	401262	662	62.8	181255	397308	1966	50.17	176519	397907	749	22.4	174415	391016	481	23.29
case20	5966107	198550	399368	131	62.81	158276	392823	2090	50.14	166786	398171	877	22.26	155494	386440	735	23.16
Average		214666.95	393669.4	560	64.201	183144	384317	1825.3	50.1335	173939.9	385605.55	601.9	22.5485	**173873.2**	**380479.75**	578.5	23.532
Ratio		1.2346	1.0347	0.9680	2.7282	1.0533	1.0101	3.1552	2.1304	1.0004	1.0135	1.0404	0.9582	**1.0000**	**1.0000**	1.0000	1.0000

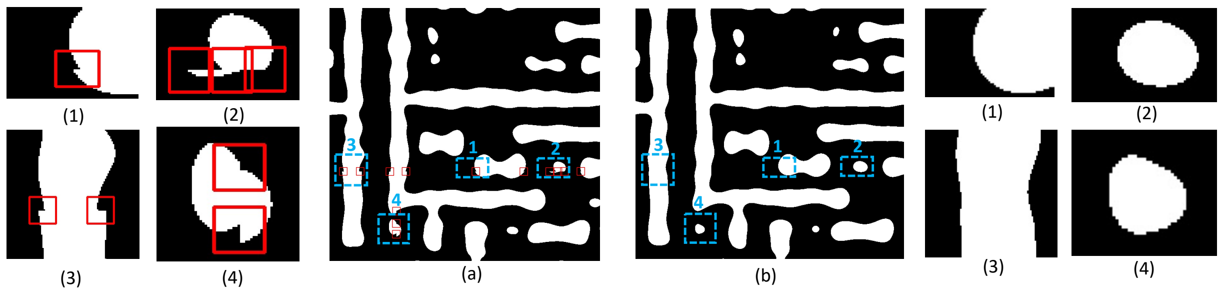

Figure 8: Stitching error comparisons of our proposed method and traditional divide-and-conquer method. (a) presents a segment generated by the traditional divide-and-conquer method, with the smaller images on the left offering enlarged views of regions exhibiting boundary mismatches. (b) is generated by our proposed method, the smaller images on the right correspond to areas where boundary mismatches are present in (a).

References

[1] Haoyu Yang and etc. Gan-opc: Mask optimization with lithography-guided generative adversarial nets. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.
[2] Bentian Jiang and etc. Neural-ilt 2.0: Migrating ilt to domain-specific and multitask-enabled neural network. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2671–2684, 2021.
[3] Ziyang Yu and etc. A gpu-enabled level-set method for mask optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(2):594–605, 2022.
[4] Shuyuan Sun and etc. Efficient ilt via multi-level lithography simulation. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.
[5] Shao Ming Yu and Yi-Ming Li. A parallel intelligent opc technique for design and fabrication of vlsi circuit. In *2005 NSTI Nanotechnology Conference and Trade Show-NSTI Nanotech 2005*, pages 724–727, 2005.
[6] Vivek Singh and etc. Making a trillion pixels dance. In *Optical Microlithography XXI*, volume 6924, pages 264–275. SPIE, 2008.
[7] Jennefir Digaum and etc. Full chip inverse lithography technology mask synthesis for advanced memory manufacturing. In *DTCO and Computational Patterning II*, volume 12495, page 1249503. SPIE, 2023.
[8] Jun Zhu and etc. Scalable hierarchy extraction of repeating structures to enhance full chip mask synthesis. In *DTCO and Computational Patterning II*, volume 12495, pages 199–207. SPIE, 2023.
[9] Linyong Leo Pang and etc. Truemask ilt mwco: full-chip curvilinear ilt in a day and full mask multi-beam and vsb writing in 12 hrs for 193i. In *Optical Microlithography XXXIII*, volume 11327, pages 145–158. SPIE, 2020.
[10] Shuo Yin and etc. Fuilt: Full chip ilt system with boundary healing. 2024.
[11] Randolph E Bank and etc. The hierarchical basis multigrid method. 52:427–458, 1988.
[12] Xiao-Chuan Cai and Marcus Sarkis. A restricted additive schwarz preconditioner for general sparse linear systems. *Siam journal on scientific computing*, 21(2):792–797, 1999.
[13] Zhijun Wang and etc. A comparative analysis of image fusion methods. *IEEE transactions on geoscience and remote sensing*, 43(6):1391–1402, 2005.
[14] Harold Horace Hopkins. The concept of partial coherence in optics. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 208(1093):263–277, 1951.
[15] Shayak Banerjee and etc. Iccad-2013 cad contest in mask optimization and benchmark suite. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 271–274. IEEE, 2013.
[16] Martin J Gander and Serge Van Criekingen. New coarse corrections for optimized restricted additive schwarz using petsc. In *Domain Decomposition Methods in Science and Engineering XXV 25*, pages 483–490. Springer, 2020.
[17] Evridiki Efstathiou and Martin J Gander. Why restricted additive schwarz converges faster than additive schwarz. *BIT Numerical Mathematics*, 43(5):945–959, 2003.
[18] Barry F Smith. Domain decomposition methods for partial differential equations. In *Parallel Numerical Algorithms*, pages 225–243. Springer, 1997.