University of California
Los Angeles

Computational Methods for Design-Assisted Mask Flows

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Abde Ali Kagalwalla

2014

Abstract of the Dissertation

Computational Methods for Design-Assisted Mask Flows

by

Abde Ali Kagalwalla

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2014

Professor Puneet Gupta, Chair

The cost per die benefit of semiconductor technology scaling that has driven Moore's law is being threatened by increasing manufacturing cost. Masks, which reproduce circuit patterns on the wafer, are the biggest contributor to this manufacturing cost. The need to print sub-wavelength patterns on the wafer has significantly increased the cost and complexity of mask manufacturing, that consists of three key steps: mask data preparation, mask write and mask inspection. In this thesis, we propose novel computational approaches to enhance mask data preparation and mask inspection that can help control mask manufacturing cost.

To reduce the pessimism of geometric approaches to estimate lithographic process window, we propose electrical process window (EPW), which accounts for electrical specifications of the circuit layout such as delay, power and static noise margin, thereby reducing pessimism by 1.5 to 8×. To reduce the pessimism in mask inspection, which can take up as much as 30% of the total mask manufacturing time, we propose design-aware mask inspection. This is accomplished by first locating non-functional features in a circuit layout, and using that information along with the timing information of the design to assign criticality to different layout shapes. This information can be exploited by mask inspection tools to reduce defect review time and first pass yield of masks. Our results indicate 39% reduction in the number of defects reported by the inspection tool and 19%-point improvement in first pass yield

of a polysilicon mask. Mask fracturing is a key component of mask data preparation that determines the e-beam shots required to write the mask. Since shot count is directly proportional to mask write time, reducing shot count is a key objective for mask fracturing solutions. To evaluate the suboptimality of modern model-based mask fracturing heuristics, we propose an ILP-based benchmarking method and an optimal benchmark generation method. Our methods show that even a state-of-the-art prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition can be suboptimal by as much as 2.3× for real ILT shapes and by 6× for generated benchmarks.

EUV lithography, a front-runner to replace the incumbent 193nm lithography, suffers from hard-to-repair defects on mask blanks. To mitigate the problem of these defects, we first propose a defect avoidance method based on random walk and gradient descent that can allow mask makers to use masks with even 30 defects without any significant yield impact. To aid the design of EUV layouts that are robust to mask defects, we propose a new metric called critical density, which can quickly evaluate the robustness of EUV layouts.

The dissertation of Abde Ali Kagalwalla is approved.

Jane Chang

Mani Srivastava

Lieven Vandenberghe

Puneet Gupta, Committee Chair

University of California, Los Angeles

2014

To my mom and dad.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Will add later.

Vita

2005–2009	Bachelor of Technology, Electrical Engineering, Indian Institute of Technology, Bombay.	2009–2010
2009–2011	Master of Science, Electrical Engineering, UCLA.	June 2011–September 2011
2011–2012	Teaching Assistant.	June 2013–September 2013
2009–2014	Graduate Student Researcher, NanoCAD Lab, UCLA.	

Publications

T.B. Chan, A. Kagalwalla, and P. Gupta. "Measurement and Optimization of Electrical Process Window," *Proc. SPIE Advanced Lithography*, February, 2010.

A. Kagalwalla, P. Gupta, C. Progler, and S. McDonald, "Design-aware Mask Inspection", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, November 2010.

A. Kagalwalla, P. Gupta, D.-H. Hur, and C.-H. Park, "Defect-aware Reticle Floorplanning for EUV Masks", *Proc. SPIE Advanced Lithography*, March 2011.

T. Chan, A. Kagalwalla, and P. Gupta, "Measurement and Optimization of Electrical Process Window", *SPIE Journal of Micro/Nanolithography, MEMS and MOEMS (JM3)*, 2011.

A. Neureuther, J. Rubinstein, M. Miller, K. Y. E. Chin, C. Levy, L. Wang, N. Xu,

C. Spanos, K. Qian, K. Poolla, J. Ghan, A. Subramanian, T.-J. K. Liu, X. Sun, K. Jeong, , P. Gupta, A. A. Kagalwalla, R. S. Ghaida, and T.-B. Chan, "Collaborative research on emerging technologies and design", *Proc. SPIE Photomask and Next-Generation Lithography Mask Technology*, 2011.

A. Kagalwalla, P. Gupta, C. Progler, and S. McDonald, "Design-Aware Mask Inspection", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012.

A. Kagalwalla, S. Muddu, L. Capodieci, C. Zelnik and P. Gupta, "Design-of-Experiments Based Design Rule Optimization", *to appear in SPIE Advanced Lithography*, 2012.

M. Gottscho, A. A. Kagalwalla, and P. Gupta, "Power Variability in Contemporary DRAMs", *IEEE Embedded Systems Letters*, 2012.

A. A. Kagalwalla and P. Gupta, "Design-Aware Defect-Avoidance Floorplanning of EUV Masks", *IEEE Transactions on Semiconductor Manufacturing*, vol. 26, 2013.

A. A. Kagalwalla, M. Lam, K. Adam, and P. Gupta, "EUV-CDA: Pattern Shift Aware Critical Density Analysis for EUV Mask Layouts", *Proc. Asia and South Pacific Design Automation Conference*, 2014.

A. A. Kagalwalla and P. Gupta, "Comprehensive Defect Avoidance Framework for Mitigating EUV Mask Defects", *SPIE Advanced Lithography Symposium*, 2014.

T. B. Chan, P. Gupta, K. Han, A. A. Kagalwalla, A. B. Kahng, and E. Sahouria, "Benchmarking of Mask Fracturing Heuristics", *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2014.

CHAPTER 1

Introduction

Over the last decade, semiconductor technology scaling has continued to keep pace with Moore's law. The biggest driver for scaling has been cost reduction due to more chips per wafer. But this cost benefit is eroding rapidly due to the huge increase in manufacturing cost with each new technology generation. This trend is illustrated in Figure 1.1.

As can be seen from Figure 1.1, a substantial component of the manufacturing cost is due to photomasks. The use of aggressive resolution enchancement techniques such as optical proximity correction (OPC), sub-resolution assist features (SRAFs) and phase-shiting masks (PSM) have led to extremely complex mask features which are hard to manufacture [GKS03]. For $45nm$ commercial digital designs, a complete mask set can easily cost more than a million dollars. Current projections from ITRS suggest that this situation is likely to get worse with newer lithography patterning technologies like double patterning or extreme ultraviolet (EUV) lithography, as Figure 1.1 shows.

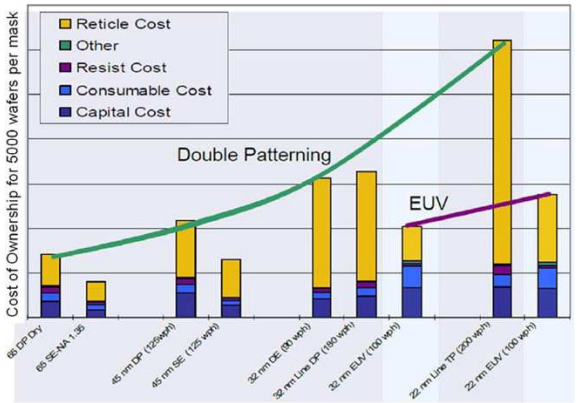


Figure 1.1: Mask cost increase with technology [itr09]

In this chapter we shall first give a brief overview of optical lithography, an integral part of semiconductor manufacturing, in Section 1.1. We shall highlight the important role played by photomasks in lithography. This will be followed by a brief description of the various steps for manufacturing masks, in Section 1.2. Finally, we shall look at one particular next generation lithography (NGL) option, EUV lithography, in Section 1.3 and highlight some key issues facing EUV mask manufacturing. We will conclude the chapter with an outline of this prospectus in Section 1.4.

1.1 Optical Lithography

Lithography has been one of the key enablers of the phenomenonal scaling seen by the semiconductor industry. The process of lithography is similar to film based photography. In photography, the desired image passes through an optical lens system and exposes a film, which is later treated to obtain a photo. In lithography, the desired target pattern is first written on a mask, which is like an optical stencil with transparent and opaque regions. When a light source shines on this mask, it creates an image which passes through an optical system and exposes a photochemical resist. This gives rise to various patterns that eventually become transistors or wires connecting transistors to make a silicon chip. This entire process is done on a wafer stepper.

A typical wafer stepper, illustrated in Figure 1.2, consists of an illumination system that comprises a light source and a set of lenses. The light obtained from the illumination system passes through a photomask which has opaque and clear regions depending on the circuit layout pattern. The light that passes through the mask pattern then exposes a chemical resist, which is deposited on the wafer. The resist is then chemically treated to leave only the unexposed part of the resist[1]. As a result, the mask pattern is transferred to the wafer.

[1]The exposed resist is chemically removed for a positive resist, which is preferred in semiconductor manufacturing. In the case of negative resist, the unexposed resist is actually removed

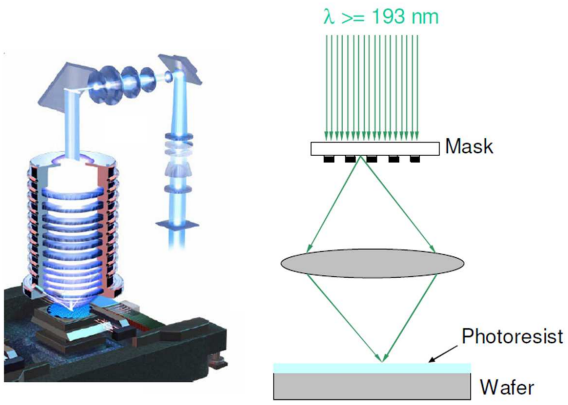


Figure 1.2: Conventional 193nm lithography system [Nau11]

This simplistic description of a wafer stepper glosses over several challenges. The single biggest issue is that the patterns that need to be printed are significantly smaller than the wavelength of light exposing it, causing diffraction which distorts the image that actually gets transferred to the wafer. Because diffraction, and even other optical phenomena like lens aberrations, are systematic phenomena that can be modeled and compensated for, a class of computational methods to rectify the distortion have emerged. These computational approaches are often referred to as resolution enhancement techniques and include OPC, retargeting and SRAF insertion.

Another critical set of problems that affect wafer steppers, that are more random in nature, is manufacturing variation. There are several sources of variation inside the stepper, but the three most important ones are exposure latitude, defocus and overlay. Exposure latitude, or dose variation, is the fluctuation in the intensity of the light source. Defocus is the error in wafer position relative to the optical system exposing it. Overlay is the alignment error between the various mask patterns that need to be reproduced on the wafer to create the final chip.

1.2 Mask Manufacturing

As mentioned earlier, a reticle (mask) is basically a stencil that determines what patterns eventually print on the wafer. The increasing aggressiveness of various resolution enhancement techniques like optical proximity correction (OPC), phase shift mask, and sub-resolution assist features (SRAF) along with decreasing feature sizes has increased the complexity, and therefore the cost, of reticles considerably [GKS03]. Keeping mask cost in control is extremely critical, especially for low volume designs. As Figure 1.1 shows, this situation will get worse for double patterning lithography which would be required at $20nm$ technology node and potentially even at $14nm$.

The various steps which are performed to convert a circuit layout to a mask that can be used to expose wafers are listed below:

- **Mask Data Preparation:** This step involves computational methods to obtain patterns that must be written on the mask from the layout pattern. As pointed out earlier, correcting the patterns for diffraction through the use of resolution enhancement techniques like retargeting, OPC, and SRAF insertion is a step. After applying these corrections, and other layout optimization techniques, the patterns are fractured into trapezoids which e-beam mask write tools can take as input.

- **Mask Writing:** Once mask patterns are obtained, mask blanks are patterned using an e-beam mask write tool. The process is very similar to optical lithography, but a narrow e-beam light source is used to expose the resist in order to achieve better accuracy.

- **Mask Inspection:** The patterned mask must be carefully checked to ensure it has no defects. This is done through a multi-step mask inspection process, which is discussed in further detail in Chapter 3. Mask inspection can take a significant amount of time, up to 30% of the total mask manufacturing time.

1.3 Replacing 193nm lithogaphy: EUV

Extreme ultraviolet (EUV) lithography is considered one of the most promising next generation lithography solutions to replace the current deep ultraviolet (DUV) lithography [itr09]. But the technology still faces several challenges before it can actually be used for volume production. The fundamental challenge limiting the adoption of EUV lithography is source power. Without adequate source power, the throughput of the expensive EUV tool is too low be to commercially viable. Resist and defect-free mask manufacturing are other significant challenges [Lev09].

High energy ultraviolet light with a wavelength of $13.5nm$, used in EUV lithography, is absorbed by all materials which prevents the use of refractive optics like DUV. As a consequence, EUV optics is reflective, as illustrated in Figure 1.3. Creating reflective masks or mirrors for EUV uses the principle of Bragg reflectors which rely on constructive interference at the interface of materials with different absorbtion rates. EUV mask blanks are constucted by stacking several molybdenum-silicon bilayer reflectors which can achieve a reflectivity of approximately 70%. The layout patterns that need to be printed on wafer are then written on the multilayer mask blank as an absorber layer.

EUV masks suffer from a unique problem we do not see in conventional lithography. The substrate on which the silicon-molybdenum bilayer is deposited has small pits or bumps on the surface. These defects can propogate to the top of the multilayer stack, as a bump or pit on the surface, causing the path and phase of the reflected EUV light to change. As a result, even a $3.5nm$ tall buried phase defect can easily print on the wafer, causing a massive critical dimension (CD) change of $20nm$ on the wafer [CN08]. Figure 1.4 illustrates the potential damage that a buried defect can cause by shorting two parallel lines. These defects are often referred to as buried defects.

EUV mask manufacturers have recently reported that they can achieve mask blanks with zero defects of size larger than $100nm$ [sem14]. However, defects smaller than $100nm$ are also capable of causing yield loss. More importantly, mask blank

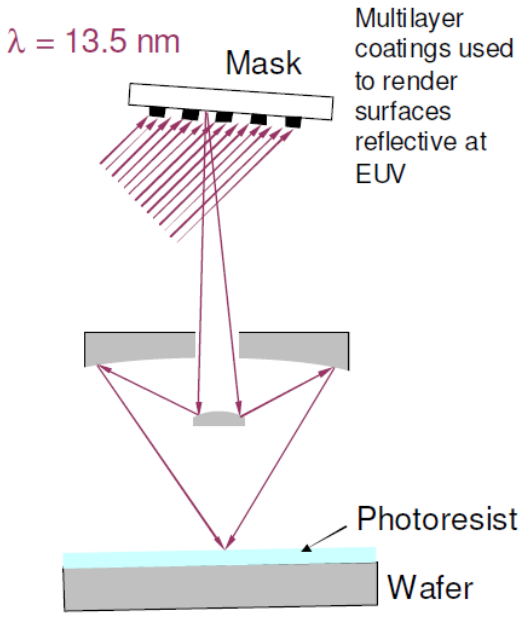


Figure 1.3: Reflective EUV lithography system[Nau11]

inspection tools tend to miss several defects [JH11], because of which the severity of this problem remains unclear. Because these mask defects are buried under multilayers, repairing them is very challenging [DL02].

1.4 Thesis Outline

In this thesis, we describe several novel computational approaches to improve the various stages of semiconductor photomask manufacturing. A key underlying theme of our work is to reduce the inherent pessimism of several mask manufacturing steps. The key contributions of this work can be summarized as follows:

- In Chapter 2, we propose a new metric to evaluate the robustness of mask patterns to the various sources of random variation on the wafer stepper. Our metric, electrical process window (EPW), reduces the pessimism of the current approach, geometric process window (GPW) by measuring variation in electrical circuit parameters like delay, leakage power and static noise margin (SNM).

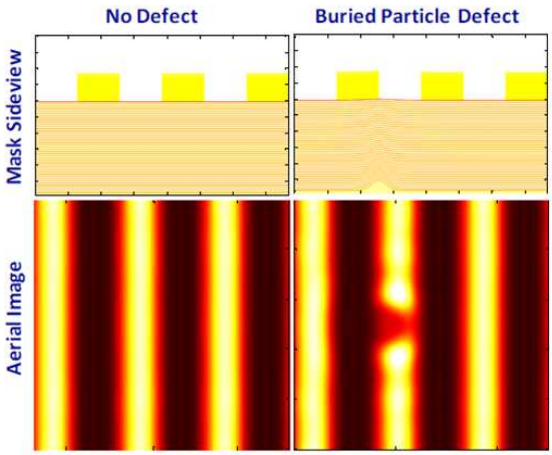


Figure 1.4: EUV mask along with its aerial image illustrating the impact of buried defects [CCN10].

- In Chapter 3, we look at methods to reduce the pessimism inherent in mask inspection steps by making it aware of non-critical and redundant features on the mask. We show how information about the criticality of mask features can help mask inspection tools reduce the reported defect count, which leads to an improvement in turnaround time and first pass yield of masks.

- In Chapter 4, we propose a method to benchmark mask fracturing heuristics. Mask fracturing determines the e-beam shots that are required to create the given mask pattern. Reducing the shot count helps reduce mask write time and consequently mask cost. Benchmarking of mask fracturing heuristics enables a better understanding of the suboptimality of the heuristics and how they can be improved.

- In Chapter 5, we propose a novel defect avoidance reticle floorplanning approach to mitigate the problem of buried defect in EUV masks. We shall show simulation results for several different scenarios and constraints based on our global optimization based method.

- In Chapter 6, we propose a metric to evaluate the robustness of EUV layouts to mask defects.

CHAPTER 2

Measurement and Optimization of Electrical Process Window

2.1 Introduction

As mentioned in Chapter 1, resolution enhancement techniques like OPC, SRAFs and phase shift masks have become a necessity to ensure the printability of sub-wavelength layout features on the wafer. Since OPC is typically performed at a nominal lithographic setup, it fails to account for variation in exposure, focus or overlay. To compensate these variations, process window OPC has been proposed in [KCG06], whereby OPCs are performed at multiple process corners. This method is, however, impractical due to long runtime. Another method, image slope OPC [CG03] optimizes slope of intensity, which is a measure of variation in dose, along with edge placement error (EPE). Retargeting [YLK09, RBF02] is a rule based technique to modify the layout before performing OPC to improve process window and is a popular approach in industry. Although these methods address the problem of lithographic variation, accurate metrics are required to quantify their benefits.

Process window is the range of process parameters such that designs produced within this range operate under desired specifications [MJL99]. Typical process window checks if the critical dimension (CD) of any feature deviates from its nominal value by more than a predefined tolerance [MJL99, LMH06] and is denoted as geometric process window (GPW) in this work. Although GPW is easy to compute or measure, it is not an accurate representation of electrical behavior of the printed circuit.

Recently, there has been some interest in reducing the pessimism due to poor correlation between design geometry and electrical performance. In [BEL08b], *electrically driven OPC* is developed based on non-rectangular transistor models for I_{on} and I_{off}. Zhang et. al. in [ZA07a] developed an analytical model to account for corner rounding in printed transistors and accounted for its impact on saturation current during OPC. Gupta et. al. in [GKS07a] used timing slack of critical paths to reduce the complexity of post-OPC mask shapes. These methods achieve smaller performance variation and reduced mask complexity despite large geometric errors [RG09]. In [ASH05], the authors propose a design-for-manufacturing methodology to compare the SNM of 6T SRAM cells printed under different defocus conditions. The method provides important feedbacks for designers at early design stage, which helps to reduce design and manufacturing costs.

Inspired by above-mentioned approaches, we propose electrical process window (EPW), which estimates PW based on delay, SNM and leakage deviation instead of variation in critical dimension (CD). In this work, we focus on PW analysis for digital VLSI circuit which has dense geometry pattern and susceptible to lithography variation [1]. To evaluate EPW, we generate post-OPC lithography contours of a given layout at different exposure, defocus and overlay (E/F/O) process points. Then, we extract transistor shapes and their electrical performances using the model in [CG10]. Finally, EPW is defined by process points that yield lithography contours with acceptable electrical performances.

The key contributions of the work we describe in this chapter are as follow:

- In contrast to the conventional GPW, we propose electrical process window defined by delay, SNM and leakage power of a design. EPW can reduce the pessimism in process control requirements as its area is 1.5$\sim$8$\times$ larger than that of GPW.

- We demonstrate that EPW can be optimized by layout transparent methods like gate length biasing and V_{th} push during manufacturing.

[1] We do not evaluate EPW for analog circuit because the layout of analog circuit is usually guardbanded with high margin to account for lithography variations.

- We propose several approximations to EPW for cases where design information is incomplete.
- We present the concept of representative layout extraction which can be used to reduce EPW evaluation runtime.

2.2 Definition of Process Windows

In this work, we focus on analyzing lithography process window for poly layer because it usually is the most critical layer in lithography. Moreover, lithography variation on poly layer has strong correlation to electrical variation as it defines transistor gate length.

2.2.1 Geometric Process Window

Definition: GPW is defined as the range of process parameters such that deviation between the CD of printed contour and circuit layout on poly layer (gate length) is within predefined tolerance, i.e.

$$(E_i, F_j, O_k) \in GPW \iff$$
$$\text{lower bound of allowed CD deviation} \leq \text{CD} \leq \text{upper bound of allowed CD deviation.} \tag{2.1}$$

In our experiments, CD deviation is estimated based on edge placement error (EPE) histogram of all transistor segments. As illustrated in Figure 2.1, EPE is defined as the displacement between printed contour and layout segments. Since EPE only measures channel length deviation on one side of transistor channel, the following scenarios are considered and CD is defined accordingly.

- Maximum EPE occurs at both edges of a transistor segment. CD = nominal channel length $\pm 2\times$ maximum EPE (worst case).
- Maximum EPE occurs at one edge of a transistor segment. We assume that the edge opposite to maximum EPE segment is not changed and CD = nominal

channel length ± maximum EPE.

Based on the definitions for CD and GPW, we consider a process point (E_i, F_j, O_k) to be within GPW if more than 99% of EPEs are smaller than predefined CD tolerance. The 1% allowance is given to avoid pessimistic GPW due to EPE outliers, which can be fixed by fine tuning mask in OPC. In subsequent sections, we use W-GPW to denote GPW with CD defined by scenario 1 (worst case) and A-GPW for GPW with CD defined by scenario 2.

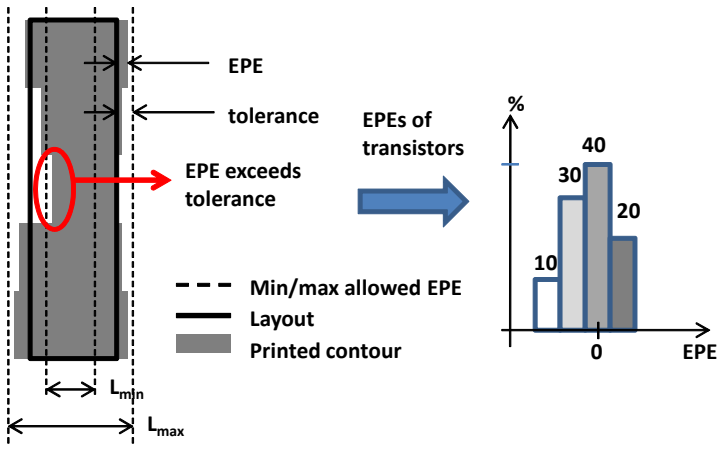


Figure 2.1: Illustration of EPE histogram.

2.2.2 Electrical Process Window

Definition: A process point (E_i, F_j, O_k) is considered within EPW if electrical performance of a printed circuit is within desired tolerance, i.e.

$(E_i, F_j, O_k) \in EPW \iff$

circuit performance lower bound $\leq$ circuit performance $\leq$ circuit performance upper bound. (2.2)

In this work, we demonstrate the evaluation of delay centric EPW (D-EPW), leakage power centric EPW (P-EPW) and static noise margin EPW (SNM-EPW) as they are commonly used electrical performance metrics. In [CGG10a], the impact of interconnect linewidth variation is found to be much smaller than the impact of

transistor gate length variation on delay. Therefore, we do not consider interconnect linewidth variation in calculating EPWs.

2.2.2.1 Delay Centric Electrical Process Window (D-EPW)

Due to sub-wavelength lithography, printed transistor channel is not rectangular despite the use of aggressive RET techniques. This imposes difficulties in EPW extraction as electrical performance of a non-rectangular gate (NRG) transistor cannot be determined from pre-characterized library. To model the impact of NRG transistors on critical path delay, we extract I_{on} of each NRG transistor using the method proposed in [CG10]. As shown in Figure 2.2, NRG transistor obtained from simulated contour is sliced into narrower transistors to approximate the non-rectangular channel. Then, the effective channel length, width and V_{th} of sliced transistors are extracted so that they can be represented as rectangular transistors[2]. Finally, the rectangular transistors are simulated using HSPICE [Hsp] and their I_{on} and I_{off} are summed up to represent total I_{on} and I_{off} of the NRG transistor. After obtaining the current, cell delay of NRG transistor is estimated by the following equation,

$$\text{Cell delay} = \frac{\sum_{j=1}^{N_i} I_{on-original-j}}{\sum_{j=1}^{N_i} I_{on-simulated-j}} \times \text{original cell delay},$$

where N_i is the total number of transistors in cell j and *original cell delay* is the delay of the cell specified in circuit's timing report. Subsequently, path delay of simulated contour ($D_{path-simulated}$) is represented as the sum of delay of every cell along the path,

$$D_{path-simulated} = \sum_{i=1}^{M} (\text{Cell delay}_i), \tag{2.3}$$

where M is the total number of cells along a critical path. Finally, D-EPW is defined as

$$(E_i, F_j, O_k) \in \text{D-EPW} \Longleftrightarrow \max(\Delta D_{path}) \leq \text{upper bound of allowed delay deviation.}$$

$$\Delta D_{path} = [\frac{D_{path-simulated}}{D_{path-original}} - 1] \times 100\%, \tag{2.4}$$

[2]We use SPICE-based method in [CG10] to calibrate parameters for NRG transistor model.

where $D_{path-original}$ is the delay of the critical path obtained from circuit's timing report.

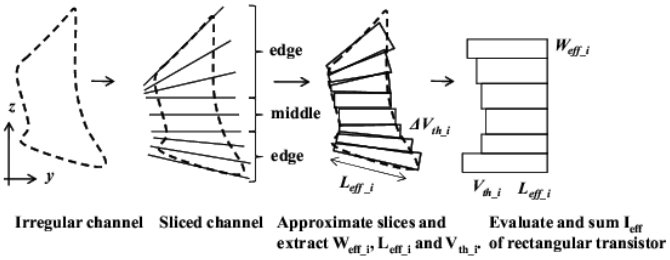


Figure 2.2: Non-rectangular gate transistor I_{on} and I_{off} extraction.

2.2.2.2 Leakage Power Centric Electrical Process Window (P-EPW)

As already mentioned, leakage current of NRG transistors at different process points ($I_{off-simulated}$) are obtained using the method in [CG10]. The method is also used for calculating leakage current of each transistor in pre-OPC layout ($I_{off-original}$) to evaluate leakage power deviation of a circuit ($\Delta power$).

$$\Delta power = [\frac{\sum_{j=1}^{T} I_{off-simulated-j}}{\sum_{j=1}^{T} I_{off-original-j}} - 1] \times 100\%, \tag{2.5}$$

where T denotes the total number of transistors in a design. Note that Equation (2.5) does not account for cell topology, i.e. stacked transistor has less leakage power compared to non-stacked transistors. This leads to an estimation error whenever CD variations are different for stacked and non-stacked transistors. Since the PEPW is a function of relative leakage power instead of the absolute value, the estimation error is only significant when stacked and non-stacked transistors have different CD variations. In other words, the estimation error is negligible if stack and non-stack transistors have similar CD distributions. For random digital logic, CD variation is affected by surrounding pattern which has no direct correlation with cell topology. Therefore the estimation error due to cell topology is unlikely a major source of error.

Since there is no lower bound for leakage power, P-EPW is defined as

$$(E_i, F_j, O_k)\in\text{P-EPW} \Longleftrightarrow \Delta power \leq \text{upper bound of allowed leakage power deviation.} \tag{2.6}$$

2.2.2.3 Signal Noise Margin Electrical Process Window (SNM-EPW)

To capture the impact of lithography imperfection on a SRAM cell, we replace each NRG transistor in the cell by an equivalent transistor which has the same I_{on} as the NRG transistor. Since there can be many width and length combinations for a given I_{on}, we choose the equivalent transistor which has channel width equal to the average width of the NRG transistor.

After obtaining the equivalent transistors for a SRAM cell, we run SPICE simulation to get the voltage transfer curves of inverter pairs in a SRAM cell. We evaluate only read noise margin, since it is typically more critical compared to hold or write noise margin. The SNM of a cell is defined by the diagonal length of maximum square within butterfly curves as shown in Figure 2.3. Due to the regular layout of SRAM array, the printed contour of each cell are similar. Therefore, we evaluate SNM-EPW based on the SNM value of a SRAM cell. SNM-EPW is defined as

$$(E_i, F_j, O_k)\in\text{SNM-EPW} \Longleftrightarrow \Delta SNM \geq \text{lower bound of allowed signal noise margin deviation,} \tag{2.7}$$

$$\Delta SNM = [\frac{SNM_{simulated}}{SNM_{original}} - 1] \times 100\%.$$

2.2.2.4 Combined Electrical Process Window (C-EPW)

Whenever there are more than one electrical performance metrics, the combined electrical PW can be easily computed by finding the intersections of the EPWs,

$$\text{C-EPW} = \bigcap_{i=1}^{Q}(EPW_i), \tag{2.8}$$

where Q is the total number of electrical performances. In this work, C-EPW is defined as the intersection between D-EPW and P-EPW.

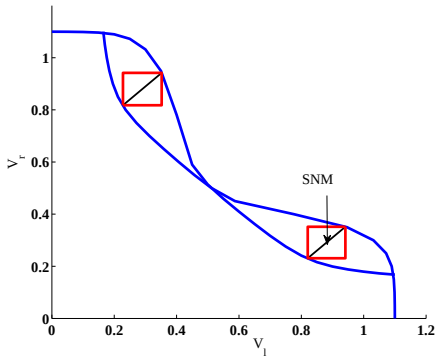


Figure 2.3: SNM extraction based on voltage transfer curves of a 6T SRAM bit cell. V_r and V_l are the internal node voltage of inverter pairs in a bit cell.

2.2.3 Relation Between GPW and EPW Tolerances

Since GPW and EPWs are defined differently, we need to figure out the relation between the two for fair comparison. To obtain the worst case corners of GPW, we simulate an inverter with 4 times fanout and a 6T SRAM cell at (nominal length $\pm$ (2 $\times$ EPE tolerance))[3] using SPICE [Hsp] and transistor model provided by Nangate Open Cell Library[nan]. The worst case delay, leakage power and SNM deviations are extracted to represent D-EPW, P-EPW and SNM-EPW tolerances, respectively. Table 2.1 summarizes the corresponding deviations in delay and leakage power for different EPE tolerances. E.g. $\pm$5% EPE (2.5nm of 50nm nominal channel length) corresponds to 11%, 54% and -24% deviations in delay, power and SNM, respectively. Hence, W-GPW with 2.5% EPE tolerance corresponds to A-GPW with 5% EPE tolerance, D-EPW with 11% delay tolerance, P-EPW with 54% leakage power tolerance and SNM-EPW with -30% SNM tolerance.

When channel length deviates more than 10%, the SNM of a 6T SRAM cell reduces to zero. Therefore, the maximum allowed geometrical deviation is 10% for SRAM. The tolerance for leakage power is very high compared to channel length and EPE tolerances because leakage power increases exponentially as channel length decreases. Note that, the tolerances in Table 2.1 are strongly dependent on the process technology.

[3]Vdd=1.1V , Temperature = 25 $^\circ$ Celsius

Table 2.1: Tolerances of GPW and EPW

ΔChannel length (%)	W-GPW Δ EPE (%)	A-GPW Δ EPE (%)	D-EPW Δ delay (%)	P-EPW Δ power (%)	SNM-EPW Δ SNM (%)
5	2.5	5	11	54	-24
10	5.0	10	21	311	-61
15	7.5	15	30	2476	N/A

2.3 Comparison Between GPW and EPW for Digital Logic

2.3.1 Experimental Setup

To show the differences between GPW and EPW for digital logic, five ISCAS-85 [ISCa] and a microprocessor (MIPS) benchmark [ope] circuits were implemented using 45nm Nangate Open Cell Library (PDK v1.2 v2008) [nan]. After synthesis, placement and routing, we define the paths within 20% of setup time constraint as critical paths. The layouts of benchmark circuits were scaled to 65nm for OPC and lithography simulation due to limitations in our optical models. After that, the simulated contours are scaled down to 45nm for leakage and drive current extraction. To emulate variations in lithography system, we simulate image for poly layer with different exposure and defocus values using Mentor Calibre[cal08]. In this work, we only analyze the PW for poly layer. During EPW extraction, we use the active layer patterns in layout, i.e. we evaluate the PW for poly layer when active layer is printed at its nominal value.

Overlay error is emulated by shifting printed active layer along vertical direction (Z direction in Figure 2.2) during transistor shape extraction. Process parameters in our experiments are as follow.

- Exposure (%) ∈ {80, 90 , 100 , 110, 120}
- Defocus (nm) ∈ {0, 40, 80, 160}
- Overlay (nm) ∈ {-20, -10, 0, 10, 20}

All process points for which any printed transistor is open or short are excluded from EPWs and GPW. This defines maximum feasible process window. To evaluate GPW, we generate EPE histogram for each process point by comparing printed contours to original layout using Mentor Calibre [cal08]. To evaluate EPW, we translate the extracted channel shapes into an OpenAccess database [oa]. After that, I_{on} and I_{off} of every transistor are extracted using the method in [CG10] to obtain deviations in delay and leakage power as mentioned in Section 2. The analysis of EPW (including NRG transistor current extraction) was implemented in C++ and the experiment was carried out on a 64bit machine running at 2GHz with 16GB memory.

2.3.2 Results

Results in Table 2.2 show that W-GPW is very pessimistic as it has zero area for all tolerances. Compared to W-GPW, A-GPW has less constrained CD definition and larger PW as expected.

Table 2.2: GPWs and EPWs area for ISCAS-85 benchmark circuits.

	W-GPW			A-GPW			D-EPW			P-EPW			C-EPW (delay,power)			Feasible
Tolerance %	2.5	5	7.5	5	10	15	11	21	30	54	311	2476	(11,54)	(21,311)	(30,2476)	Area
c432	0	0	0	0	300	1276	1538	2086	2460	882	1720	2107	0	1086	1846	2760
c499	0	0	0	0	117	1375	1559	2105	2508	921	1718	2076	9	1103	1864	2760
c880	0	0	0	0	196	1278	1390	1956	2332	825	1464	1969	0	890	1770	2565
c1355	0	0	0	0	95	1313	1665	2204	2560	847	1569	2052	35	1052	1891	2760
c1908	0	0	0	0	139	1253	1388	1937	2309	841	1493	1988	1	900	1767	2565
MIPS	0	0	0	0	0	190	921	1209	1426	334	599	823	0	248	690	1590
average	0	0	0	0	141	1114	1410	1916	2266	775	1427	1836	7	880	1638	2500

Figure 2.4 shows the scatter plots of W-GPW, A-GPW, D-EPW, P-EPW and C-EPW for benchmark circuit c1908. Although the experiments are carried out for different E/F/O, the overlay axis is excluded in these plots because it is observed that the PW is insensitive to overlay for the layouts we have. To reduce lithography simulation runtime, we estimate delay, leakage power and EPE values between sampled data points by interpolation. The experiment results for other circuits are not

displayed but the area of the PWs are stated in Table 2.2. [4] From Figure 2.4 [5], we can clearly notice the area of A-GPW is smaller than the areas of EPWs with corresponding tolerances. This implies, there are process points where printed circuit can meet electrical tolerances although its CD violates geometric constraints. GPW is a more pessimistic metric compared to EPW because of the following reasons:

- GPW requires at least 99% EPE to be within tolerable range. In contrast, EPW only restricts the total power and delay of a circuit which is the average of deviation of each transistor segment. Therefore, some of the transistor segments can vary significantly but the entire transistor is still able to meet EPW tolerance due to the averaging.

- All transistors are not equally important in EPW. For instance, delay constraints are applied only for transistors on critical paths instead of all transistors in a design.

- Averaging across multiple transistors in a critical path for delay or all transistors for power.

It is observed that at 100% exposure and 80nm defocus (circled in Figure 2.4), A-GPW with 15% EPE tolerance is within tolerance (shaded) but P-EPW with corresponding leakage power tolerance is not. This happens whenever the actual channel length deviation (combined EPE on both edges) is larger than 7.5nm (15% of channel length) but none of the EPEs exceeds 7.5nm. As a result, the process point is considered valid in A-GPW but the actual leakage power is greater than pre-defined leakage power constraints. This example shows that *A-GPW is generally pessimistic compared to EPW but it does not guarantee the electrical performance of circuit printed within its PW.*

[4] The result of W-GPW is not included in Figure 2.4 as its has zero area in all cases.

[5] It is noticed that, the ideal process point at 100% exposure and 0nm defocus lies outside P-EPW at 54% tolerance. Meanwhile, process points at 90% exposure and 0∼80nm defocus meets the tightest delay and leakage power tolerance. We believe this is due to imperfect calibration of our OPC setup.

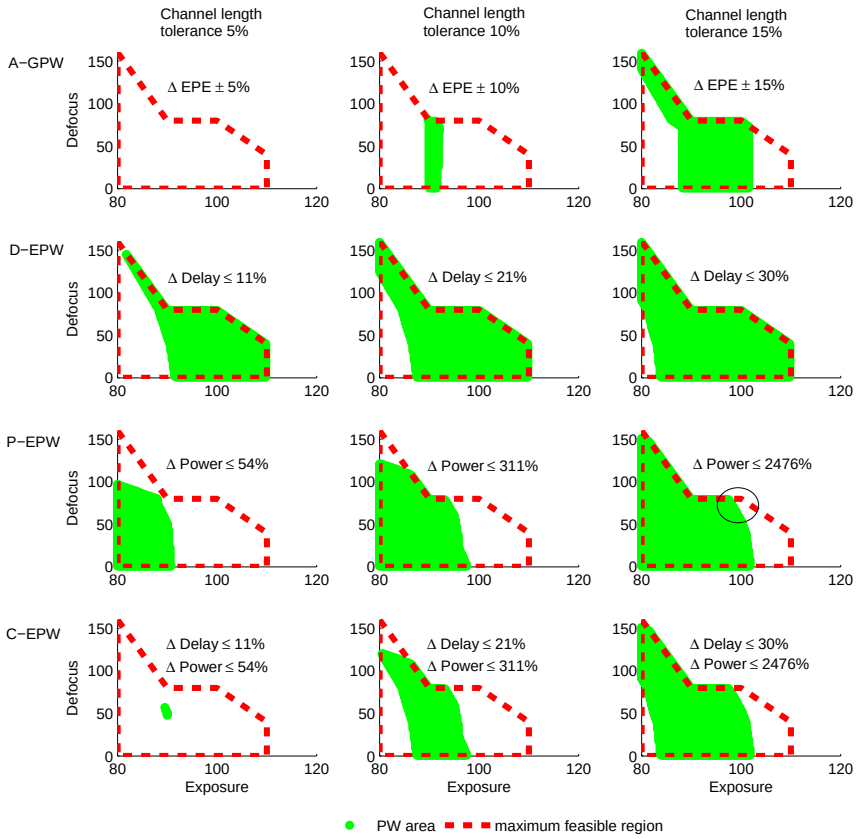


Figure 2.4: Scatter plots of A-GPW, D-EPW, P-EPW, C-EPW for ISCAS-85 benchmark circuit c1908.

When both leakage power and delay are considered, C-EPW can be much smaller than D-EPW or P-EPW as shown in the 4^{th} row in Figure 2.4. C-EPW is valuable as it clearly defines the acceptable process range, ensuring printed design can meet both delay and power requirements. In cases where A-GPW and C-EPW have comparable tolerances as mentioned in Table 2.1, the area of C-EPW is 1.5∼8× larger than that of A-GPW.

2.4 Optimization of Electrical Process Window

With EPW, the impact of process tuning on PW can be estimated from simulated contours. This enables fast and extensive exploration of process tuning approaches for maximizing PW. Since C-EPW is defined as the intersection of D-EPW and

P-EPW, it is possible to improve C-EPW by increasing D-EPW or P-EPW. But any change in gate lengths or V_{th} has opposite effects on D-EPW and P-EPW. E.g. P-EPW increases along with transistor gate lengths (leakage power reduced) but vice versa for D-EPW. Therefore, there is always a trade-off between D-EPW and P-EPW. As long as the sensitivities of P-EPW and D-EPW to the intentional gate length or V_{th} perturbation are different, they can be leveraged to improve C-EPW.

In this work, we assume ± 2nm gate length biasing and ± 20mV V_{th} push are allowed. To emulate gate length biasing, we adjust gate lengths of transistors during I_{on} and I_{off} extraction and the adjustment is conformal to gate's edges. Meanwhile V_{th} push is implemented by adjusting the nominal V_{th} of each transistor during I_{on} and I_{off} extraction.

Figure 2.5 shows that reducing the gate lengths or lowering V_{th} enlarges D-EPW as expected. Meanwhile they reduce the area of P-EPW because total leakage power is increased when gate length or V_{th} of transistors are reduced. Since D-EPW only considers delay deviation on critical paths, reducing gate lengths on critical cells or all cells has identical impact on D-EPW. For benchmark circuits $c880$ and $MIPS$, however, this is not true because one or more of the reduced gate lengths on non-critical cells in the circuits are smaller than the minimum acceptable gate length (30nm). Any transistor smaller than this minimum gate length is considered as electrically shorted and it is a catastrophic circuit failure. As a result, the process points which print the shorted transistor are treated as not feasible points which reduce the D-EPW for circuit $c880$ and $MIPS$.

Alternatively, one can improve P-EPW by increasing gate length (non-critical or all cells) or V_{th} of transistors. Figure 2.5 shows that the approaches have similar improvements for P-EPW but the impacts of these approaches on D-EPW vary significantly. Since increasing gate length or V_{th} of all transistors also increases critical path delays, D-EPW of these approaches are smaller compared to D-EPW of optimization approach which increases gate length of non-critical cell only. There are cases ($c880$ and $MIPS$) where increasing gate lengths of non-critical cells have comparable impact to that by increasing gate lengths of all cells because the number

of critical cells is relatively small compared to the number of total cells as indicated in Table 2.3.

Table 2.3: Ratio of critical cells to total cells in benchmark circuits.

Circuits	Critical cells/total cells
c432	50%
c499	24%
c880	16%
c1355	49%
c1908	26%
MIPS	3%
Average	24%

On average, biasing gate lengths selectively improves C-EPW while biasing gate lengths of all cells reduces the area of C-EPW. Besides, reducing V_{th} also improves C-EPW and vice-versa for increasing V_{th}. Based on this analysis, reducing V_{th} seems to be a good approach in absence of any design information, as it improves C-EPW consistently for all benchmark circuits. Moreover, it can be done without knowing the locations of critical cells.

2.5 EPW Approximations

In practice, critical paths of the design may not be available to the foundry. Instead of reverting to GPW, which is very pessimistic as already mentioned, we propose two methods to estimate EPW using purely geometric means.

2.5.1 Method I: Use EPE Histogram of Entire Design

This method uses the EPE histogram generated during OPC to approximate EPW without extracting channel shape of each transistor. We assume that average delay and leakage power deviation induced by EPEs of all transistors are approximately

the same as that of an artificial *equivalent transistor* with the EPE histogram of entire design. As illustrated in Figure 2.6, based on the EPE histogram extracted for entire design, each non-zero EPE bin is translated into a transistor edge which has the corresponding EPE. Consequently the channel width of each transistor segment is proportional to the percentage count [2] of its EPE bins.

Since EPE can happen on both sides of a transistor,

$$\text{channel length} = \text{nominal channel length} + 2\times EPE \text{ (worst case}^3\text{)}. \tag{2.9}$$

After constructing the *equivalent transistor*, its I_{on} and I_{off} can be estimated by NRG current extraction method mentioned earlier. Note that, the histogram is mainly constructed by the EPE of the middle part of transistor channel. These transistor sections have uniform V_{th} as they are not affected by *narrow width effects* which happens at transistor edges. During NRG current extraction, we assign each segment in the equivalent transistor to their corresponding uniform V_{th} value. Therefore the extracted current is independent of slices ordering [4].

Since delay is inversely proportional to I_{on}, we estimate delay deviation as the ratio of the I_{on} of a reference transistor to the calculated $I_{on-equivalent-transistor}$. As shown in Figure 2.6, the reference transistor has nominal channel length and its total channel width is same as the one of *equivalent transistor*. Meanwhile leakage power deviation is estimated by the ratio of $I_{off-equivalent-transistor}$ to the I_{off} of the reference because leakage power is proportional to I_{off}. The approximated EPWs are called *histogram-EPWs* in the remaining text and their definitions are given as

[2] If all edge fragments are not of equal width, the histogram can be weighed appropriately.

[3] Based on our experiment results, defining "channel length=nominal channel length + EPE" leads to over-optimistic approximations that cover large area out of reference EPWs. Therefore, we assume worst case EPE condition for this approximate method.

[4] We ignore the error due to V_{th} location dependency as it is not a major source of error compared to the simple approximations in EPW definitions. A better accuracy is possible by using complex layout extraction to keep track of edge vs. center EPE

follow,

$$(E_i, F_j, O_k) \in \text{histogram-D-EPW} \Longleftrightarrow$$
$$[\frac{I_{on-reference-transistor}}{I_{on-equivalent-transistor}} - 1] \times 100\% \leq \text{upper bound of allowed delay deviation}$$
$$(E_i, F_j, O_k) \in \text{histogram P-EPW} \Longleftrightarrow$$
$$[\frac{I_{off-equivalent-transistor}}{I_{off-reference-transistor}} - 1] \times 100\% \leq \text{upper bound of allowed power deviation} \tag{2.10}$$

In our experimental setup, EPE histogram included edge displacement of PMOS and NMOS transistors together. To estimate transistor current correctly for static CMOS, the width ratio of PMOS and NMOS is taken into account when we calculate I_{on} and I_{off},

$$I = \frac{K \times I_{PMOS} + I_{NMOS}}{K + 1},$$

where K is the ratio of PMOS to NMOS channel width. In our experiments, we use the average K across different logic cells in Nangate Open Cell library [nan] which is ≈ 1.7.

2.5.2 Method II: Use Shape of Every Transistor

Given the shape of every transistor, as mentioned in previous sections, we can extract I_{on} and I_{off}. Thus, we can calculate P-EPW based on the definitions in Equation (2.6) and no approximation is required. On the other hand, exact D-EPW cannot be determined as the information of critical cells is not available. Clearly, a strict D-EPW can be defined by the worst case delay variation of all transistors. But this definition is pessimistic as it ignores averaging effect along a critical path, which usually contains more than a single cell. To reduce the pessimism, we approximate D-EPW by averaging the delay deviation of R number of transistors with the largest delay deviation. The delay deviation of each transistor is given by

$$\Delta Delay = [\frac{I_{on-original}}{I_{on-simulated}} - 1] \times 100\%,$$

where $I_{on-original}$ is the I_{on} of the pre-OPC transistor obtained from layout and $I_{on-simulated}$ is the I_{on} of NRG transistor from simulated contour. The approximated D-EPW is named as shape-D-EPW and its definition is given as follows:

$$\begin{aligned} &(E_i, F_j, O_k) \in \text{shape D-EPW} \iff \\ &\frac{\sum_{n=1}^{R} \Delta Delay_n}{R} \leq \text{upper bound of allowed delay deviation.} \end{aligned} \tag{2.11}$$

Based on the critical paths of our benchmark circuits, we found that the average transistor stages along a critical path is about 30. Therefore, we used R=30 in our experiment for pessimistic approximation. Note that this definition does not guarantee a strict lower bound as there might be cases where the logic stages along critical paths are less than R and they contain some of the transistors with the worst delay deviations.

Alternatively, we assume that the EPE distribution of transistors along critical path is similar to that of all transistors in a design. In this case, we can estimate D-EPW by averaging the delay deviation of all transistors, i.e. R= total number of transistors.

2.5.3 Results

Figure 2.7 shows that histogram D-EPW is similar to the reference D-EPW but the area of histogram P-EPW is significantly smaller than that of reference P-EPW. As a result, the approximated histogram C-EPW only covers a small region of reference C-EPW. The error in histogram P-EPW is mainly due to the definition of channel length in Equation (2.9), where worst case condition is assumed. To make matters worse, the error is exaggerated in P-EPW as leakage power grows exponentially when channel length shrinks.

Meanwhile, Figure 2.7 shows that shape D-EPW and shape C-EPW with R=30 is much smaller than that of reference EPWs. The accuracy of the approximation improves when R is increased to total number of transistors. Since the evaluation of shape P-EPW is same as the one for reference P-EPW, there is no difference between them.

In Figure 2.8, we can see that all approximation methods cover higher EPW area compare to A-GPW on average. When both leakage and delay are considered, shape C-EPW with R=*all transistors* has the highest area coverage among the approximated C-EPWs. Although histogram D-EPW shows the highest percentage coverage compared to shape D-EPWs, the covered EPW region for histogram C-EPW is low due to the poor coverage of histogram P-EPW. It is observed that the EPW area covered by shape D-EPW with R=*all transistors* is slightly less than histogram D-EPW although both approximation used the average delay deviation of all transistors to define D-EPW. This discrepancy is due to the difference between the lumped EPE histogram and actual transistor shape.

It is observed that there are several cases where histogram D-EPW has region out of D-EPW. This happens because histogram D-EPW is evaluated based on the EPE histogram of entire design while D-EPW only consider the transistors along critical paths. In contrast, shape D-EPW with R=*all transistors* has no area out of D-EPW.

In summary, *EPW extracted based on the shape of each transistor (with R=all transistors) is the best approximation among these approaches as it has no area out of EPW and the covered EPW areas are larger than 70% on average.*

2.6 Runtime Reduction Through Representative Layout Extraction

The above mentioned process window evaluation methods (GPW and EPW, including approximation methods) require lithography simulation of a single design at multiple process points, which is very slow for large design. The problem worsens if we want to evaluate PW by considering process points at a finer level of granularity. To reduce this lithography simulation runtime, we propose an efficient PW analysis flow depicted in Figure 2.9. First, we extract representative layouts (RLs) which contains relevant shapes for EPW analysis. For D-EPW, all critical cells are selected while only 5% of the total cells are selected for P-EPW analysis. Second,

we check the printed image of original layout for all process points to filter out the process points which have pinching/bridging features. This can be done efficiently by using a less accurate but fast lithography simulation setup, which is sufficient to detect pinching/bridging [7]. In case the selected cells are too many for efficient lithography simulation, we can apply an additional clustering procedure to further reduce total number of cells. Lithography simulation runtime is reduced because these RLs have a smaller feature count compared to original layout.

2.6.1 Representative Layout Extraction

For constructing representative layouts, the key thing to observe is that for delay estimation we only need to consider transistors on critical cells because they are more likely to cause timing violation under process variation instead of the entire design. We take a $2um \times 2um$ square snippet centered at each transistor's channel (of each critical cell) to form basic layout snippets. The size of snippets is chosen to account for optical proximity effects on the transistor under consideration. These layout snippets are then tiled in a separate layout, which we shall call the Delay Representative Layout (DRL) of the design.

For power analysis, there is no obvious selection scheme to extract "critical" shapes as each transistor contributes to total leakage power. To avoid analyzing entire layout, we sample 5% cell instances from each cell type. We can adjust the sampling rate for obtaining better accuracy. $2um \times 2um$ snippets for each transistor of each of the chosen cells are then used to construct a Power Representative Layout (PRL) in a similar manner to DRL. This approach of sampling cells for PRL construction reduces runtime while minimizing estimation error because standard cells with the same cell type are likely to have similar leakage power deviation.

Only DRL and PRL of a design layout then undergo lithography simulation at different process corners to evaluate EPW. Note that, we use neighboring shapes of a transistor during RL extraction but we only perform EPW analysis on the

[7] Note that identifying PW to avoid bad pinching/bridging patterns is not sufficient as there are patterns which can only tolerate small errors due to timing and they are design dependent.

transistor in the middle of the snippet for both DRL/PRL. We apply the approximate EPW methods discussed earlier to the representative layouts because complete EPW analysis requires detailed information of the critical path. The total lithography simulation runtime of these two RLs of a design is still substantially less than that of the entire design layout as shown in Table 4 [8] for one large Mips processor layout. We can further reduce total transistor shapes that need to undergo lithography simulation by clustering the chosen layout snippets using the method outlined in [GMM09]. The runtime improvement due to clustering is also shown in Table 4 [9].

Figure 2.10 shows the accuracy of our DRL+PRL extraction method compared to evaluation of EPW for the entire design. Both EPE-histogram and shape approximation methods were tried for the RLs. The results show that the PW estimated using representative layout method is similar to the one which uses entire design. The shape approximation method is slightly optimistic and overestimated P-EPW. This is due to the fact that the random sampling misses out some critical patterns that cause leakage power failure. Note that there is no area out of EPW for the histogram method. This happens because the error in sampling is compensated for by the pessimistic estimation of the histogram method as mentioned in Section 5. In summary, the RL extraction method reduces lithography simulation runtime significantly at the cost of some loss in EPW accuracy as some EPW critical geometries are not captured due to sampling/clustering. (i.e., the representative snippets do not have all the features of the critical geometries.)

Table 2.4: Lithography runtime for representative layouts

Benchmark Circuit	Total cells	Critical cells	Lithography Runtime (Hours)		
			Full Design	Representative Layout	Post-clustering
MIPS	11577	382	198	101	93

[8] The runtime values are the CPU TIME as reported by Mentor Calibre [cal08].
[9] Clustering runtime is not included, but depending on implementation it can be expensive.

2.7 EPW Including SRAM

To evaluate the EPW of digital circuits, we need to consider the PW for random logic as well as memory cells. Since the original benchmark circuits does not have memory cells, we draw the layout of the SRAM according to the geometrical dimensions in [BAB05]. After that we optimize the bit cell for Nangate devices by sizing up the pull down transistors from 80nm to 120nm. This improves the static noise margin from 163mV to 213mV. The area of the bit cell is $2.9\mu m^2$ ($0.785\mu m \times 0.370\mu m$).

In our experiment, we duplicate the layout of a 6T SRAM cell to form a memory array for lithography simulation. During PW analysis, we evaluate the bit cell in the middle of the array, which is not affected by empty patterns around layout boundaries.

2.7.1 GPW vs. EPW

Figure 2.11 shows that SNM-EPW is much larger compared to the GPW because of the following reasons:

- SNM is affected by the relative "drive strength" of transistors instead of absolute critical dimension deviation. E.g. when channel length of all transistors increase due to lithography variation, the impact of I_{on} reduction in pull down transistor is compensated by I_{on} reduction of access transistor. As a result, the SNM of a SRAM cell may still within desired specification even though the printed contour violates geometrical tolerance.

- There is an averaging effect across transistor channel.

To perform a full EPW analysis on benchmark circuits, we define C-EPW as the intersection of delay, power and SNM-EPW. We use ±10% CD tolerance for SRAM and ±10% CD tolerance for random logic in our experiments.

Figure 2.12 shows that both GPW and C-EPW do not change after intersecting the digital logic and SRAM PWs. This implies that the SRAM bit cell is not a

limiting factor for PW. Also, we notice that A-GPW shows some feasible area which is not covered by SNM-EPW or P-EPW. This happens when actual channel length deviation is larger than 10% but none of the EPE exceeds 10%. In other words, these process points are considered valid if we use the definition of AGPW but the actual SNM and leakage power violate pre-defined specifications. The results in Table 2.5 show that C-EPW is about 8× larger than GPW on average for digital logic and SRAM circuits.

Table 2.5: GPW and EPW area with SRAM.

	A-GPW	C-EPW (delay,power,SNM)	Feasible area
c432	300	1086	2760
c499	117	1103	2760
c880	196	890	2565
c1355	95	1052	2760
c1908	139	900	2565
MIPS	0	248	1590
average	109	839	2448

2.7.2 Impact of SRAM on Approximation Methods

We also study the impact of including SNM-EPW to the approximation methods mentioned in Section 5 and 6. Figure 2.13 shows that the C-EPWs (including SRAM C-EPW) of approximations methods are greater than the PW of GPW. Including SNM-EPW in the C-EPW does not change the result of approximation methods (Section 5) because the SNM-EPW is not the limiting PW in this case. Similarly, Figure 2.14 shows that including SNM-EPW does not change the result of our representative layout approaches.

2.8 Conclusions

In this work, we have proposed electrical process window which is a better measure of process window than the conventional geometric process window. The area of EPW is found to be 1.5$\sim$ 8 $\times$ larger than the GPW for our benchmark circuits because it removes the inherent pessimism of GPW by averaging the impact of geometric variation on electrical parameters. We have also analyzed various layout transparent methods to enlarge EPW. Based on our experiment results, we found that gate length biasing and V_{th} push can improve EPW by about 10%. Calculation of delay centric EPW requires information of critical cells in design which is often not available to foundries. Hence, two approximations to EPW, one based on EPE histogram and another based on transistor shape analysis have been proposed. Our results show that the EPW estimated using transistor shape covers more than 70% of the area of reference EPW on average. We also proposed a method to extract representative layouts which can be used to reduce simulation runtime for process window extraction. The method was able to reduce process window evaluation runtime by 49% with limited impact on accuracy. Though we demonstrate the process window analysis under defocus and exposure variations, other lithography imperfections such as mask error can be included in lithography simulation.

In this work, we measure the EPW at a process point and a supply voltage. Though averaging across The reported EPW will be smaller if we consider v_{th} and V_{dd} fluctuation. To account for additional process variation and supply voltage fluctuation, we can evaluate EPW at worst case corners, which gives a more pessimistic estimation. If probability distribution of a process parameters are available, we can reduce the pessimism by simulating the circuit with the statistical information.

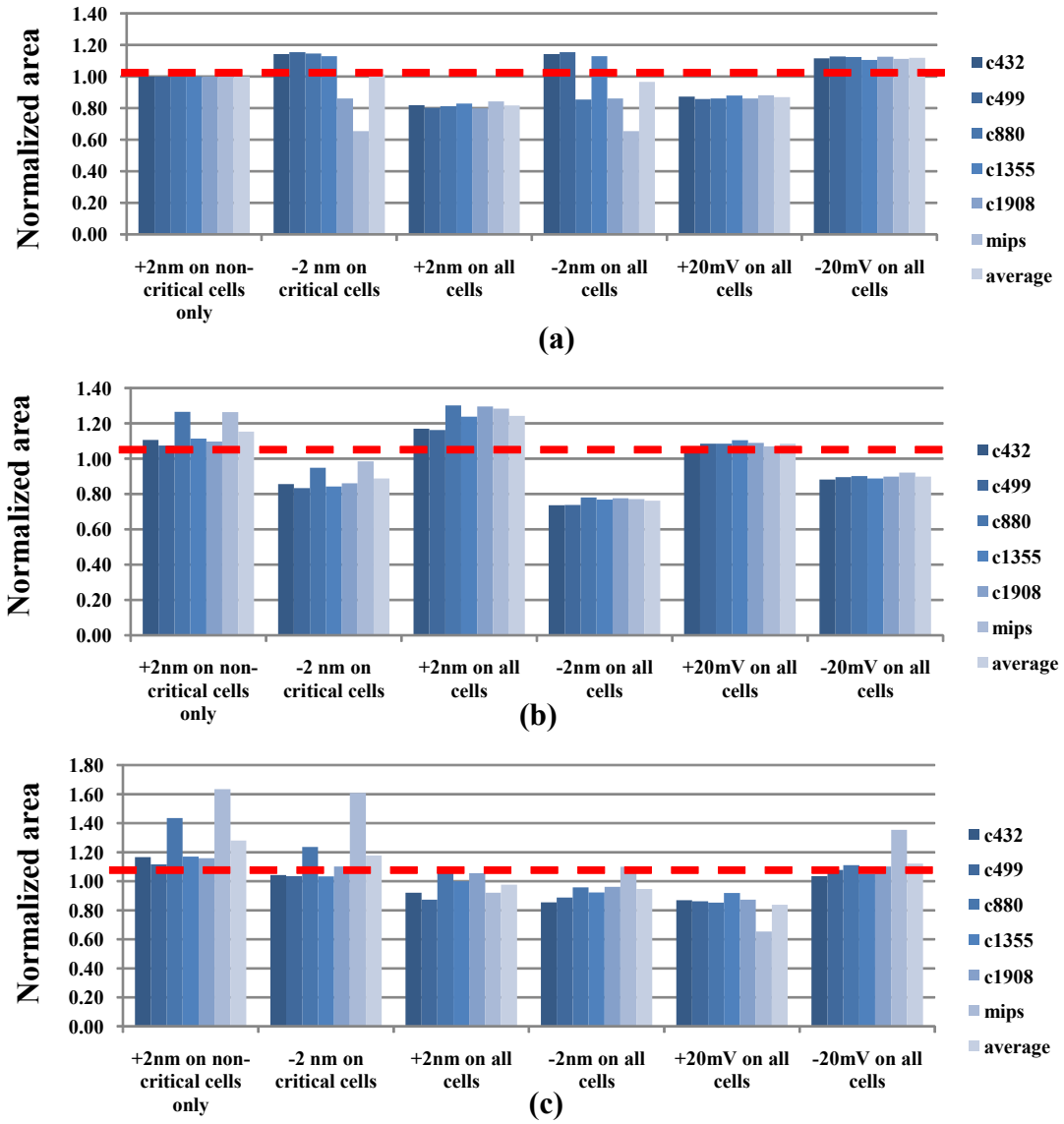


Figure 2.5: Optimized EPW area normalized to unoptimized EPW area for a)D-EPW b)P-EPW c)C-EPW. Tolerances for delay and leakage power are 21% and 311% respectively.

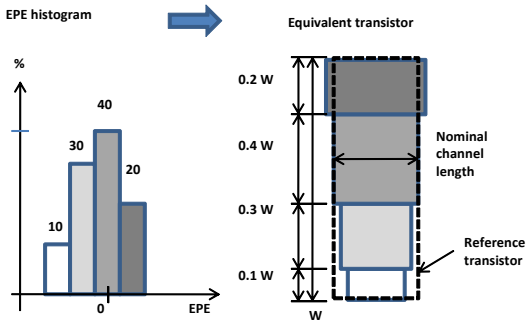


Figure 2.6: Extracting equivalent transistor from EPE histogram.

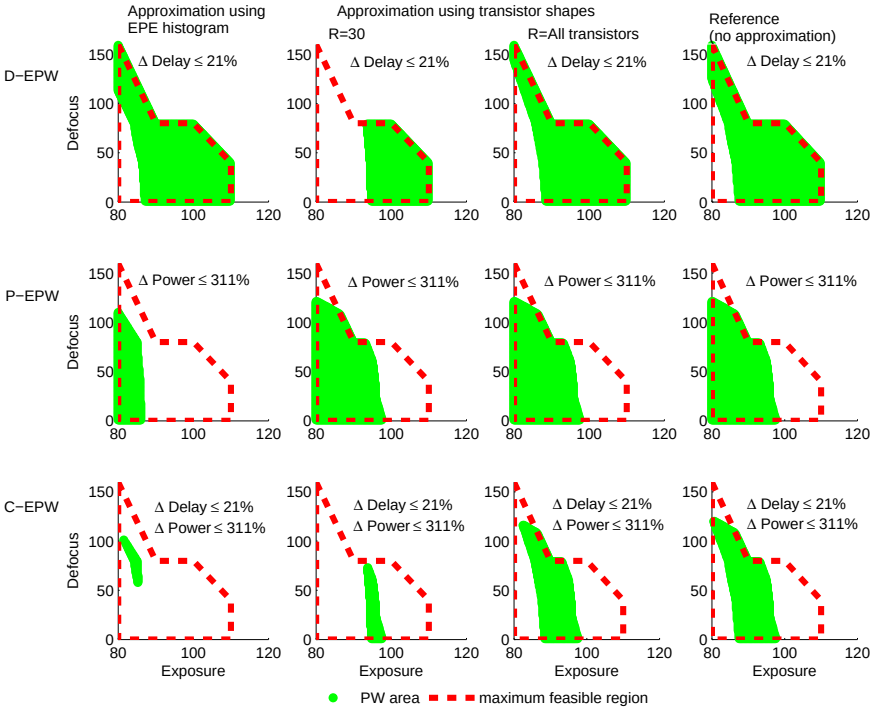


Figure 2.7: Comparison between EPW and its approximations for benchmark circuit c1908.

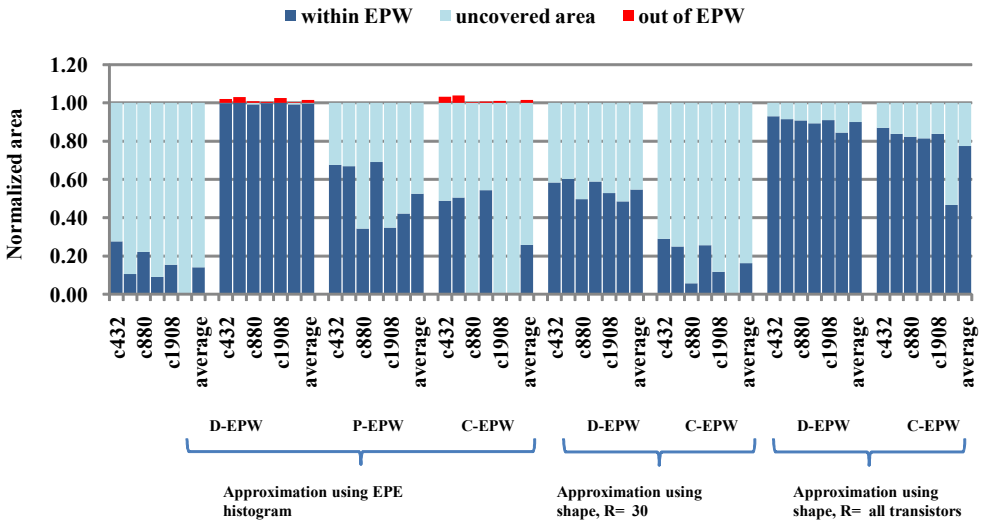


Figure 2.8: Accuracy analysis for A-GPW and approximated EPWs of benchmark circuits. EPE tolerance=10%, delay tolerance = 21% and leakage power tolerance = 311%.

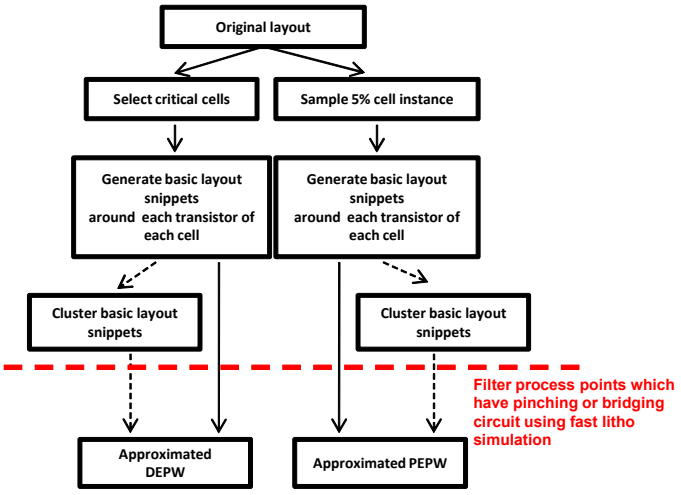


Figure 2.9: Clustering flow.

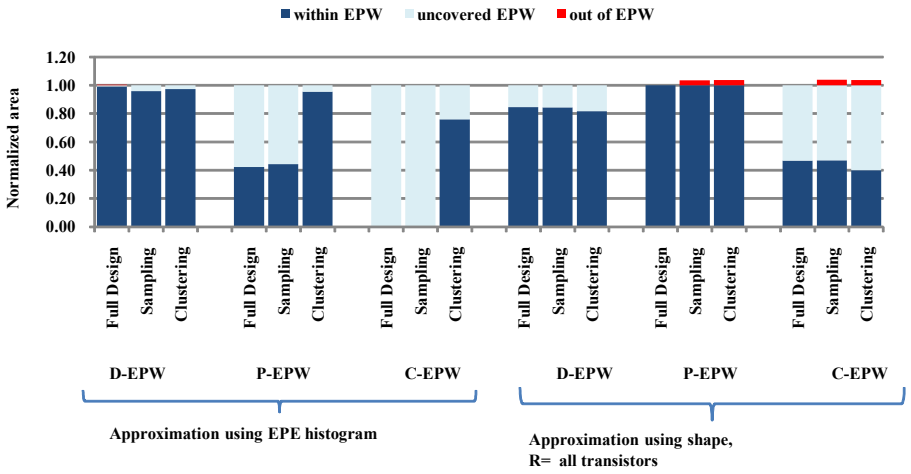


Figure 2.10: Accuracy of clustering approach for benchmark design $MIPS$.

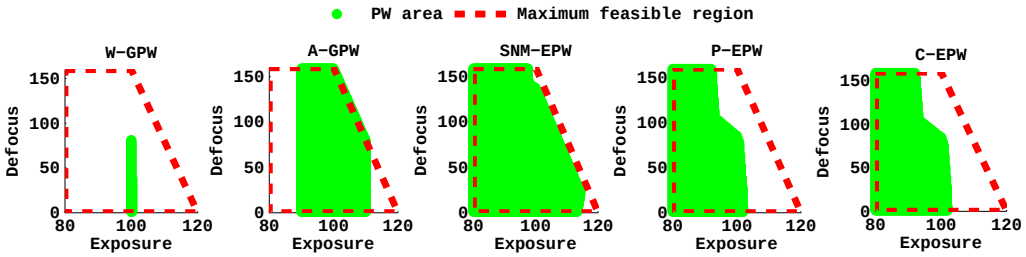


Figure 2.11: SRAM GPW vs EPW.

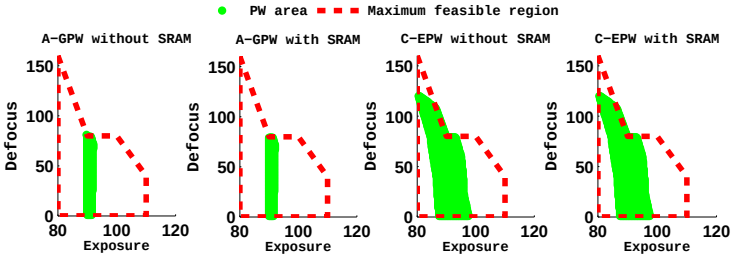


Figure 2.12: GPW vs EPW for benchmark circuit c1908.

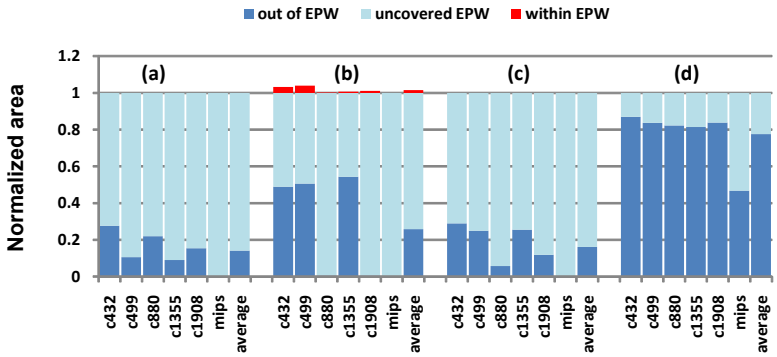


Figure 2.13: Accuracy of a)A-GPW b) C-EPW using histogram approximation c) C-EPW using shape approximation with R=30 and d) C-EPW using shape approximation with R=total.

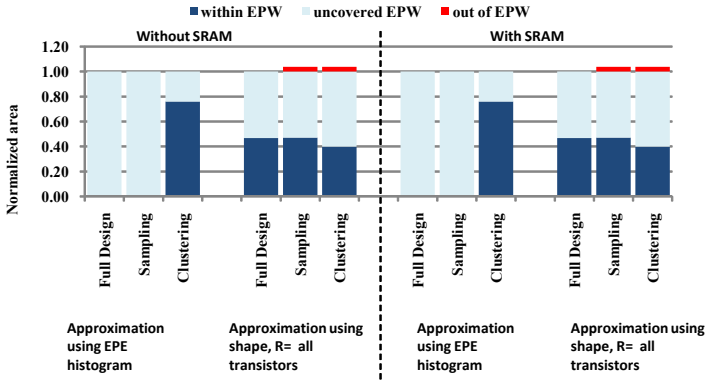


Figure 2.14: Accuracy of clustering approach including SNM-EPW for benchmark design $MIPS$.

CHAPTER 3

Design-Aware Mask Inspection

3.1 Introduction

Reticle inspection is a significant contributor to the mask cost. In fact, mask inspection is more challenging (and expensive) than mask writing itself [HK08]. High resolution reticle inspection tools are required to detect every potential printable defect in order to prevent yield loss, but inspection tools can report a large number of defects that do not affect yield. As a result post-inspection review of the yield impact of defects has become very time consuming. This slow, and often manual inspection flow has a considerable impact on mask cost and turnaround time (TAT). Hence there is a strong need to improve the inspection flow.

In this chapter we develop a methodology to assign criticality to different mask features based on their design impact. Mask inspection tools can use this information to adapt their resolution locally for different regions without missing any critical defects, thereby saving inspection time. We now present a brief introduction of current mask inspection methodology, followed by a survey of some related work and a summary of our contributions.

3.1.1 Mask Inspection Primer

A comprehensive inspection of the reticle must be done by the mask shop before sending it to the fabs. The basic steps of inspection are shown in Figure 3.1.

Initially the reticle is passed through an inspection tool such as KLA-Tencor's Terascan [DMI08] or NEC's LM series [MBM08] which takes an image of die on

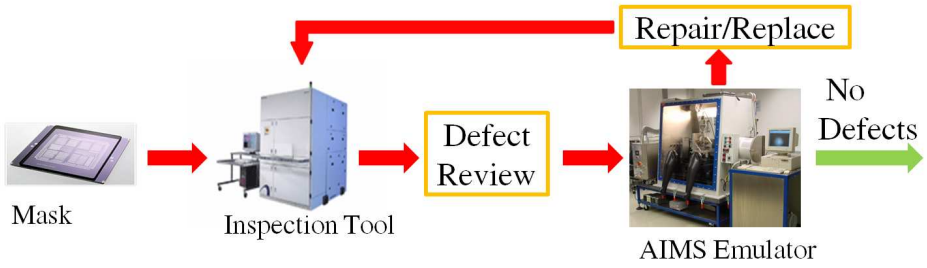


Figure 3.1: Key steps of reticle inspection

the mask and compares it to a reference database or another die (die-database or die-die modes). The difference between the two image intensities is found and if the difference exceeds a predefined threshold, the difference pattern is labeled a defect. The inverse of this threshold is referred to as sensitivity. These tools can have a pixel size as low as $55nm$ and can detect critical dimention (CD) defects as small as $20nm$ on the mask at maximum sensitivity (minimum threshold) [DMI08].

Inspection tools can generate a very large number of defects (100+) most of which do not impact the final design. Defects can be classified as shown in Figure 3.2. A *false defect* is an incorrect detection reported by the inspection tool due to vibration, misalignment, optical distortion, error in database rendering (die-database mode), etc. Real defects are caused either due to misalignment or vibration of the mask writer (CD defects) or contamination of the mask (contamination defects). Inspection tools typically have different algorithms to detect these two categories of defects and hence have different sensitivities for these defects. Many real defects do not print on the wafer. Among printable defects, some lie on non-critical regions of the design such as dummy fill or redundant vias. Only a small fraction of the defects reported by the inspection tool really matter. All the non-printable and non-critical defects are also called *nuisance defects*. Reducing the number of false+nuisance defects reported by the inspection tool is essential to reduce inspection cost. Reducing nuisance defects is particularly important to maskshops as it impacts *first pass yield*, which is the fraction of total masks manufactured that can be shipped without repair or detailed review.

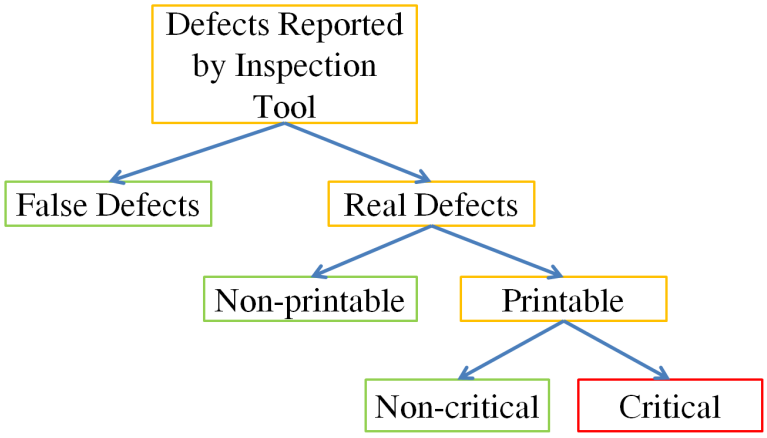


Figure 3.2: Various categories of defects reported by an inspection tool

The next step in mask inspection is defect review where each defect reported by the inspection tool is checked to find out if it really matters. False, non-printable and non-critical defects are filtered out during this step. Images of defects reported by the inspection tool are analyzed using software tools [PLC03, KAC01] or manually. Often defect images need to be recaptured at a better resolution. For this the inspection tool could be reused (online review) or an e-beam inspection is employed [KLZ08]. After pruning out a significant fraction of false/non-printable/non-critical defects the mask is passed through an aerial imaging tool. Aerial image measurement system (AIMS) [DZB06] is essentially a hardware emulator of the wafer stepper that operates at optical settings similar to the stepper and gives a very accurate estimate of the printability of defects. Although extremely accurate, AIMS is slow and cumbersome. Hence, minimizing the number of defects that have to pass through AIMS tools is important in order to ensure reasonable turnaround time. Defects which are found to be printable by the AIMS tool are then either repaired or if they are unrepairable the reticle must be replaced. The repaired or replaced reticle must again go through this inspection cycle. Because of the manual steps and use of AIMS tool, defect review is typically the slowest part of reticle inspection.

3.1.2 Related Work

There has been considerable work to improve mask manufacturing by using design intent though most of it has focused on OPC. For instance, Banerjee et. al. [BEL08a] use estimates of on/off current of transistors, based on simulated resist contours, to reduce OPC runtime and mask complexity. Zhang et al. [ZA07b] modeled the impact of corner rounding in printed transistors on saturation current and integrated their model into an OPC framework. Similarly, [THT08, KNO07] used device performance estimates to tune the aggressiveness of optical correction achieving up to 93% reduction in mask complexity. Gupta et. al. [GKS07b] use electrical and design metrics to reduce OPC runtime and mask write cost. In chapter 2, we used estimates of design metrics like delay and power to improve the evaluation of process window. These approaches indicate that considerable benefit can be derived by using design intent to reduce the inherent pessimism in various mask manufacturing steps, including inspection.

The traditional approach to mask inspection discussed in the previous section does not use any design information to assess the criticality of defects. Defect disposition is done only on the basis of printability which is determined using software tools like Virtual Stepper [PLC03, KAC01] along with AIMS emulation [DZB06]. It assumes that all printable defects larger than a threshold size (say 10% of mask CD) are critical. If design information is available to maskshops, they may be able to avoid the expensive process of repair/replacement of the mask due to printable but non-critical defects. Design information can also be used to reduce false and nuisance defects reported by the inspection tool.

Communicating design intent to the inspection tool in the form of additional control layers has been suggested before [VHR03, HLE08, TTN08]. Mask shops can use design information to lower the inspection sensitivity of non-critical regions in order to reduce the number of false+nuisance defects. Hedges et al. [HLE08] have shown that up to 100× reduction in nuisance defect count is possible just by using variable sensitivity during reticle inspection. Current inspection tools allow the user

to define inspection sensitivity on a per pixel basis. But memory requirements to store this sensitivity information are impractical since a reticle can have up to 10^{12} pixels. These approaches assume that maskshops know the design criticality of the layout which is rarely the case. Driessen et al [DGS08] analyze a post-OPC layout to extract some non-critical features in the absence of any design data. Stoler et al [SRM07] extract some criticality information as part of Manufacturing rule check(MRC). Both these approaches focus on extracting assist features from the layout which are a major source of nuisance defects.

3.1.3 Our Work

The key contributions of the work described in this chapter is as follows:

- We develop a graph based algorithm to locate non-functional features (redundant and dummy features) in a post-OPC layout (flat and 10× more complex than pre-OPC layout) in the absence of any design information.
- We assign minimum defect size that impacts the design to each feature of the reticle for both CD and contamination defects. This is inferred using the timing slack of critical paths and the location of non-functional features found using the method mentioned above. This analyis is done for polysilicon, active, contact and all the backend layers.
- Using the minimum defect size of each feature of a reticle, we partition the layout using a recursive algorithm, where each partition is assigned a different pixel size and sensitivity to minimize false + nuisance defects. First order models for false and nuisance defects are developed to do this partitioning.
- We develop a model to estimate first pass yield of masks and show the improvement achieved by our design-aware mask inspection.

3.2 Non-Functional Feature Finding

In this section, the following problem is explored: *Given a post-OPC layout identify non-functional features of the layout.* We focus on locating redundant vias and dummy fill geometrically. Other non-functional features such as non-tree routes and assist features can also be found using our graph based methodology, but are not explored in this work. The layout is assumed to have only rectilinear shapes, and that floating dummy fill in different metal layers are not connected through a via[1]. This is consistent with most commercial fill synthesis tools.

In order to identify non-functional features, we first fracture the layout into rectangles. A scan-line based algorithm is used to construct a neighborhood graph for these rectangles. The neighborhood graph is then simplified using some edge contraction operations. This reduced neighborhood graph (RNG) can then help identify dummy fill and redundant vias. The various steps of our approach are detailed below.

3.2.1 Algorithm Steps

- *Fracturing Polygons:* The rectilinear polygons are fractured into rectangles using a simple horizontal slicing method [GG83]. The rectangles are then stored in different sets based on their layer. For example, a rectangle corresponding to a Metal 2 shape is stored in two sets, M_2V_1 and M_2V_2. A set M_iV_j corresponds to all rectangles belonging to same/adjacent metal or via layers, where a via layer V_j connects metal layer M_j and M_{j+1}[2].

- *Neighborhood Graph Construction:* The new layout with fractured polygons is used to construct an undirected Neighborhood Graph, $G(V, E)$ in which every rectangle of the fractured layout corresponds to a vertex and edge $(u, v) \in E$ if the two corresponding rectangles are physically in contact with each other

[1] This constraint can be relaxed, if needed.

[2] Although storing each rectangle twice leads to redundant computation, we actually found this method be be faster than storing all the rectangles in a single set due to smaller interval+segment tree size during scan-line.

in the layout.

A scan-line based one-pass, optimal algorithm is used to solve the rectangle intersection problem as described in [SW80]. The problem is reduced to two subproblems, an interval query and a point query. Interval tree and range tree are two "semi-dynamic" tree data-structures that are used to solve this problem [CLR01]. We shall refer to these two sets of trees as scan-line trees. A separate scan-line is used for each set M_iV_j but there is a single graph for the entire layout. Both these trees can perform INTERSECTSEARCH[3], INSERT and DELETE operations in $O(log(m))$, where m is the number of nodes in the tree [CLR01].

- *Edge Contraction:* All neighboring vertices of the Neighborhood Graph that correspond to rectangles of the same layer are merged. At the end of this operation each vertex has an edge only to vertices belonging to an adjacent layer. Hence, a vertex corresponding to Metal 2 in RNG will have edges only to vertices of Via 1 or Via 2 and so on.

- *Graph Analysis:* Floating fill is identified by looking for isolated vertices. Cycles in the RNG correspond to redundant vias which can be identified using depth first search (DFS). Double and even multi-cut vias can be identified by scanning the reported cycles and identifying the set of vias connected to the same pair of metal layer vertices in RNG.

3.2.2 Runtime Improvement Techniques

- *Routing-Aware Scan-Line:* The routing direction of each set of rectangles, M_iV_j can be found by taking the larger of the average length and width of all rectangles in the set. If the routing direction is X (Y) we define y (x) coordinates of the rectangles as scan-line events so that the average duration

[3]INTERSECTSEARCH returns all rectangles stored in the scan-line tree that intersect the input rectangle and constructs edges in the neighborhood graph between the input and all returned rectangles

for which a rectangle needs to be stored in the tree reduces, thus improving INTERSECTSEARCH time.

- *Shape Simplification:* The complexity of layout features increases tremendously after OPC. This results in a large number of rectangles after fracturing and slows down the algorithm presented in Section 3.2.1. Before fracturing the polygons into rectangles, we perform shape approximation on the post-OPC polygons to reduce the number of rectangles created after fracturing. We create two sets of buckets for the coordinates of each polygon. Each point is included in two buckets, one in x-direction and another in y such that x (or y) coordinate of each point in a bucket is within a certain threshold distance of others. All the x (or y) coordinates of a bucket are then changed to the average x (or y) coordinate of the corresponding bucket. This approach reduces small deviations along a straight line as shown in Figure 3.3 and hence reduces rectangle count, while preserving connectivity.

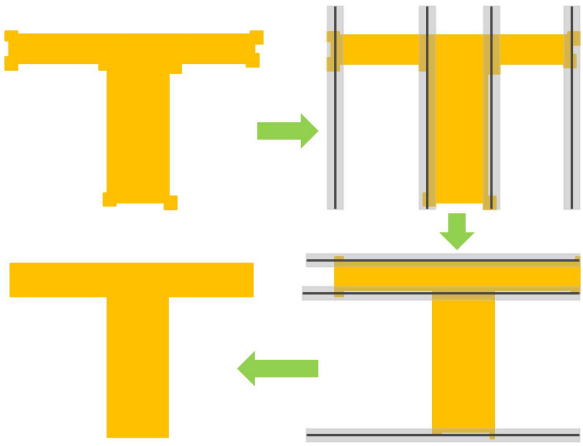


Figure 3.3: Shape simplification for a distored T-shape

Algorithm 1 summarizes the entire algorithm. Figure 3.4 illustrates the complete algorithm for a sample double via.

We can now analyze the runtime complexity of our approach. If a layout has N rectangles, then the neighborhood graph construction can be done in $O(Nlog(N) + E)$ time, where E is the number of intersecting pairs of rectangles [SW80]. The neigh-

borhood graph has N vertices and E edges. We can sort the vertices in $O(Nlog(N))$, and perform edge contraction in $O(E)$. RNG will then have N' vertices ($N' < N$) and E' edges ($E' < E$). N' and E' depend on the particular design layout and the aggresiveness of OPC. We can then perform DFS on the RNG to identify redundant vias and dummy fill in $O(N' + E')$ time. Hence, the overall complexity of our approach is $O(Nlog(N) + E)$.

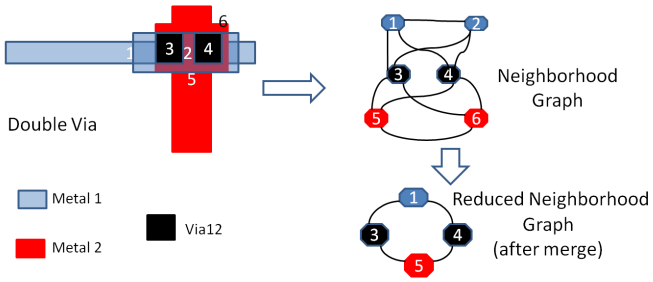


Figure 3.4: Illustration of various steps of non-functional feature finding

3.3 Criticality Assignment

This section focuses on the following problem: *Given the timing of critical paths and non-functional features identified in the previous section , find the minimum size reticle defect at each location in the layout which can cause failure.*

On the basis of geometry, reticle defects are classified as pindots, pinholes, intrusion and extrusion. Intrusion and extrusion defects are considered CD defects. Pindots and pinholes are usually classified as contamination defects. These two categories of defects are detected using different approaches and hence we treat them separately during criticality assignment. Apart from the size, type and location of defect, CD impact of a defect on the wafer also depends on the type of reticle (bright-field or dark-field), type of resist (positive or negative) and mask error enhancement factor (MEEF) at the defect location.

Reticle and resist type depends on the mask layer under consideration. MEEF, on the other hand, changes within a mask itself. It is a function of neighborhood mask features and the optical parameters of the lithography system. There are three

Algorithm 1 Non-functional feature finding

Require: Shapes of all metal and via layers, S.

1: **for all** Shape $s \in S$ **do**

2: SHAPE-SIMPLIFICATION(s)

3: Set of rectangles, B_s = FRACTURE(s)

4: Store B_s in set M_iV_j corresponding to shape layer

5: **end for**

//EVENT DEFINITION

6: Find routing direction R of each rectangle set, M_iV_j

7: **if** Routing direction R is X(Y) **then**

8: Store $bottom(left)$ and $top(right)$ of each rectangle in set as separate events in E_{ij}.

9: **end if**

//SCAN-LINE

10: **for all** Events $e \in E_{ij}$ for each set E_{ij} **do**

11: **if** e is $bottom(left)$ **then**

12: INTERSECTSEARCH(Scan-line Tree, $e.rect$)

13: INSERT(Scan-line Tree, $e.rect$)

14: **else**

15: DELETE(Scan-line Tree, $e.rect$)

16: **end if**

17: **end for**

//EDGE CONTRACTION

18: Edge Contract $G(V, E)$ to obtain RNG $G(V', E')$

//GRAPH ANALYSIS

19: Mark all isolated vertices as dummy fill

20: Find cycles in $G(V', E')$ using DFS to detect redundant vias

potential methods of accounting for MEEF in criticality assignment, which we shall explore further in Section 6.5:

- Rely on modern inspection tools that support adaptive thresholding, i.e. the threshold value is dynamically changed by the tool depending on online MEEF estimation [DMI08]. In this case, we can choose MEEF=1 since the inspection tool can adjust for it.
- Find the worst case MEEF (across process window) for all fragments for each mask shape through lithographic simulation and assign a MEEF value to each mask shape.
- Find the worst case MEEF for the entire mask for each layer type and use that value for assigning criticality of every shape of that reticle.

Note that our criticality analysis is focused on binary defects only. Phase defects are not considered since defect data from a commercial maskshop suggests they are rare[4]. A square approximation is used to model defect shape (similar to most critical area analysis methods).

Details of criticality assignment for different reticle layers is detailed in the the following sub-sections. For the polysilicon layer, we use the timing slack of various paths to assign the minimum size defect for each polysilicon shape corresponding to a transistor pair (PMOS+NMOS) on the critical path. Since defects are very small compared to layout shapes, we assume that their impact on parasitic or coupling capacitance is negligible for all layers. Minor change in dimensions of back-end layer shapes do not affect circuit metrics like delay or power, as shown in [CGG10b]. Hence prevention of opens or shorts is the only concern for assigning criticality to back-end layer reticles. As a result we only utilize location of redundant vias and dummy fill to assign minimum size defect for via and metal layers, respectively. For all our analysis, we assume that assigning a tolerable defect size of 20% the minimum width/space design rule is sufficient to prevent shorts or opens[5]. **We also assume**

[4]In data for over 700 reticles, we did not see any phase defects

[5]For simplicity and pessimism we use minimum DR values instead of using exact design values.

that a single layout shape is not affected by more than one defect since mask defect density is typically very low (order of few tens of defects for a full reticle). Table 3.1 lists the notation used in this section.

Table 3.1: Glossary of terminology used in this section

Term	Definition
a	Size of square defect
a_{min}	Minimum detectable defect size of the inspection tool
W_{min}	Width design rule of given layer
S_{min}	Spacing design rule of given layer
Df_{min}^{CD}	Minimum tolerable CD defect
Df_{min}^{Con}	Minimum tolerable contamination defect

3.3.1 Polysilicon Layer

Polysilicon layer printing typically uses bright-field masks with positive photoresist. The impact of different reticle defect types is illustrated in Figure 3.5 and summarized in Table 3.2.

Table 3.2: Design impact of different defect types in polysilicon layer

Type	Gate Length	Design Impact
Intrusion	Decrease	Open/Delay Decrease
Extrusion	Increase	Short/Delay Increase
Pinhole	Decrease	Open
Pindot	No change	None

Since extrusion defects can cause timing failure, we must estimate the minimum size of an extrusion defect that can cause timing failure. Consider an extrusion defect as shown in Figure 3.5. In order to estimate the delay change caused by this defect, the transistor is sliced into three parts (similar to [CGG10b]) to estimate the effective $\frac{W}{L}$ as shown in Equation 3.1. Using first order transistor models, we

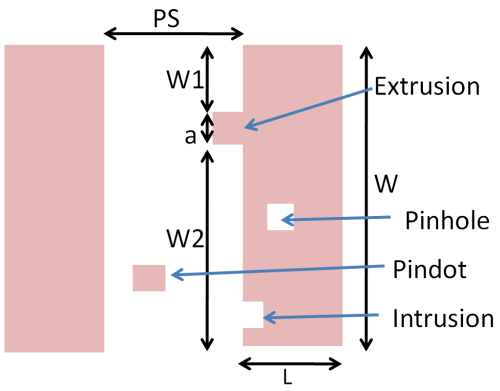


Figure 3.5: Illustration of various defect types on polysilicon layer

can then estimate the change in drive current caused by this defect, which is then used as a pessimistic approximation of change in cell delay as shown in Equation 3.2. Assuming that at most K defects lie on a critical path[6], we can evaluate the minimum defect size that changes the delay of each affected transistor by less than T_{slack}/K and hence ensure timing correctness of the path. Here, the timing slack must be obtained from a timing report that designers need to pass on to mask shops. The maximum tolerable defect size, $a_{critical}$ can then be estimated as shown in Equation 3.3, with an additional guardband of α_{cycle} to allow for other sources of variation.

$$\left(\frac{W}{L}\right)_{new} = \frac{W1}{L} + \frac{W2}{L} + \frac{a}{L-a} \tag{3.1}$$

$$\frac{\triangle Delay}{Delay_{nom}} = -\frac{\triangle \frac{W}{L}}{\frac{W}{L}} = \frac{a^2}{WL} \tag{3.2}$$

$$\begin{aligned} a_{critical} &= a_{min} \quad if \quad T_{slack} < \alpha_{cycle} \\ &= \sqrt{\frac{(T_{slack} - \alpha_{cycle})/K}{Delay_{nom}} WL} \quad otherwise \end{aligned} \tag{3.3}$$

To guardband against process variations downstream, we set the minimum defect size as 20% the width (opens) and spacing (shorts) dimensions. We assume that pinholes do not have any parametric impact and can only cause an open if they are

[6]A critical path typically consists of only $20-50$ transistors and hence the area occupied by a critical path is very small compared to the area of the chip. We take $K = 10$ as a pessimistic value.

bigger than the gate length. Hence, we can assign the minimum size of CD defects and contamination defects for any polysilicon feature as shown in Equation 3.4 and Equation 3.5, respectively.

$$Df_{min}^{CD} = \frac{min(0.2W_{min}, 0.2S_{min}, a_{critical})}{MEEF} \tag{3.4}$$

$$Df_{min}^{Con} = \frac{0.2W_{min}}{MEEF} \tag{3.5}$$

3.3.2 Active Layer

The potential impact of any active defect is determined by the location of the defect relative to an overlapping polysilicon or contact shape as shown in Figure 3.6.

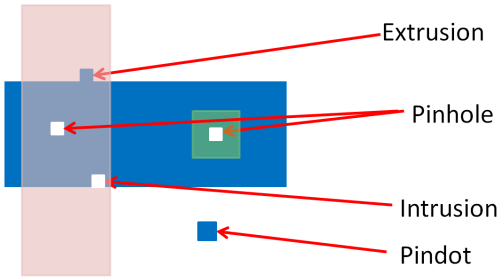


Figure 3.6: Illustration of various defect types on active layer

Active layer is usually patterned using brightfield masks with positive photoresist. Hence, we can summarize the design impact of any defect on an active layer reticle as shown as Table 3.3. Note that an intrusion defect on the active layer reticle can cause change in delay of a transistor if it lies on the overlap area with poly. Although the exact analysis would require TCAD simulations, we make the pessimistic assumption that an intrusion defect of size a reduces the transistor width by the same amount. Hence, we can calculate the maximum defect size that does not cause timing failure using a similar analysis as that of the polysilicon layer, as shown in Equation 3.6,

3.7 and 3.8.

$$\left(\frac{W}{L}\right)_{new} = \frac{W-a}{L} \tag{3.6}$$

$$\frac{\triangle Delay}{Delay_{nom}} = W(\frac{1}{W-a} - \frac{1}{W}) \tag{3.7}$$

$$\begin{aligned} a_{critical} &= a_{min} \quad if \quad T_{slack} < \alpha_{cycle} \\ &= W\left(\frac{T_{slack} - \alpha_{cycle}/K}{Delay_{nom}}\right) \quad otherwise \end{aligned} \tag{3.8}$$

Table 3.3: Design impact of different defect types in active layer

Type	Design Impact
Intrusion	Delay Increase
Extrusion	Active short
Pinhole	Contact/Polysilicon Open
Pindot	No impact

Pindot defects have no impact but pinhole defects can cause an open contact or a malfunctioning transistor[7]. The maximum tolerable pinhole defect size is therefore, determined by polysilicon/contact design rules since it can cause an open contact or transistor. Extrusion defects do have not have a significant impact unless they cause a short with another active shape.

Based on the above analysis, we can assign maximum acceptable defect size for CD and contamination defects as shown in Equation 3.9 and Equation 3.10, respectively.

$$Df_{min}^{CD} = \frac{min(0.2W_{min}, 0.2S_{min}, a_{critical})}{MEEF} \tag{3.9}$$

$$Df_{min}^{Con} = \frac{min(0.2W_{min}^{poly}, 0.2W_{min}^{contact})}{MEEF} \tag{3.10}$$

[7]Modeling the delay change on a transistor due to a pinhole defect on the transistor also requires more elaborate TCAD based simulation study and is not dealt with in this work.

3.3.3 Metal Layer

Dark-field masks with positive resist are typically used to make trenches for depositing copper (dual damascene process). The impact of various types of defects is shown in Table 3.4.

Table 3.4: Design impact of different defect types in metal layer

Type	Wire Width	Design Impact
Intrusion	Increase	Short
Extrusion	Decrease	Open
Pinhole	No change	None
Pindot	Decrease	Resistance Change

Small changes in back-end layers are known to have little impact on timing [CGG10b]. Hence, we focus only on opens and shorts for assigning criticality to metal layer shapes. Dummy fill do not have any design impact and can be assigned a relaxed defect size tolerance for both CD and contamination defects. Hence for a non-dummy metal layer feature, minimum defect size for CD defects and contamination defects can be assigned as shown in Equation 3.11 and Equation 3.12.

$$Df_{min}^{CD} = \frac{0.2min(W_{min}, S_{min})}{MEEF} \tag{3.11}$$

$$Df_{min}^{Con} = \frac{0.2S_{min}}{MEEF} \tag{3.12}$$

3.3.4 Contact and Via Layer

Dark-field masks with positive resist are typically used to print via layer. Impact of various defect types on via layer is summarised in Table 3.5 and shown in Figure 3.7.

Similar to metal layers, we assume that the impact of mask defects on electrical metrics is negligible and we only consider opens and shorts while assigning criticality. Note that regions where the non-fill shapes on adjacent metal layers overlap must be assigned minimum detectable defect size of the inspection tool for contamination

Table 3.5: Design impact of different defect types in via/contact layer

Type	Via Width	Design Impact
Intrusion	Increase	Short
Extrusion	Decrease	Open/Resistance Increase
Pinhole	None	Metal Short
Pindot	Decrease	Resistance Increase

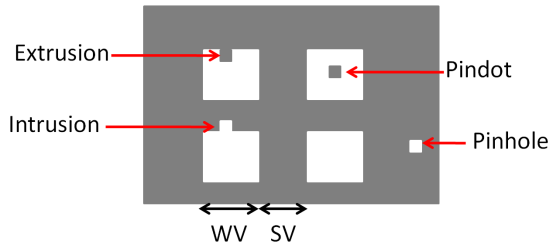


Figure 3.7: Illustration of different defect types on via layer

defects since even the smallest pinhole defect could cause a short. Similar to the other layers, 20% change in via area is taken as the constraint to assign defect size for CD and contamination (pindot) defects. Redundant vias will have a larger tolerance for defects. We can write the minimum size defects for a set of mXn redundant vias ($m = 1, n = 1$ for single via) as shown in Equation 3.13 and Equation 3.14.

$$Df_{min}^{CD} = \frac{0.2max(m,n)min(W_{min}, S_{min})}{MEEF} \tag{3.13}$$

$$\begin{aligned} Df_{min}^{Cont} &= \frac{0.2max(m,n)W_{min}}{MEEF} \\ &= a_{min} \quad for \quad metal \quad intersect \quad regions \end{aligned} \tag{3.14}$$

3.4 Modeling the Inspection Process

In this section, we develop a model for two key inspection tool properties; resolution and defect count. Defect count of the inspection tool is sub-divided into false defects and nuisance defects. All these properties are modeled in terms of pixel size and sensitivity, which are the two key tunable parameters for mask inspection. In addi-

tion to this, we develop a model to estimate first pass yield, which is a key metric that determines mask cost.

3.4.1 Resolution

The resolution of any digital imaging system scales linearly with pixel size. Also, increasing the sensitivity helps in detecting smaller features. Hence, for an inspection with pixel size, pix and sensitivity S, we shall model resolution as shown in Equation 3.15 for both CD and contamination defects. Current inspection tools are capable of inspecting a $20nm$ defect (on the mask) which corresponds to $5nm$ on the wafer (MEEF=1) at a pixel size of $55nm$ and sensitivity of 100 [DMI08]. Hence, we take $K_c \approx 9$ for our experiments[8].

$$R_{min} = K_c \frac{pix}{S} \tag{3.15}$$

3.4.2 Defect Models

3.4.2.1 False Defects

Due to the presence of random temporal noise[9], the intensity falling on each pixel of the inspection tool sensor during image capture can be modeled as a Gaussian random variable as shown in Equation 3.16, where $p(I)$ is the probability of the intensity value being equal to I, I_m is the average intensity at the pixel under consideration and σ is the temporal noise [Nak05].

$$p(I) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(I-I_m)^2}{2\sigma^2}} \tag{3.16}$$

Now, suppose die-to-database inspection is done with optimum biasing settings such that intensity at each pixel of the reference database is equal to mean intensity of the corresponding mask. Let us assign the threshold for intensity as T, i.e. any pixel is labeled as a defect if $|I - I_m| > T$. Hence the probability of a particular pixel

[8] K_c is slightly different for CD and contamination types of defects but we assume a constant value for simplicity.

[9] We assume that fixed point noise sources can be compensated for by post-capture image processing

being labeled as defective due to the Gaussian noise is given by Equation 3.17.

$$P(defect) = 1 - \int_{I_m - T}^{I_m + T} p(I)dI = 2erfc\left(\frac{T}{\sqrt{2}\sigma}\right) \tag{3.17}$$

The various components of temporal noise in a typical CCD sensor are reset noise, shot noise and read noise. Reset noise is typically compensated by correlated double sampling. The most critical component of noise is shot noise, comprising dark current and photon shot noise [Nak05]. We assume that the inspection system is photon noise limited. Photon noise is caused due to the randomness in the number of photons exposed to each pixel. The number of photons falling on a pixel follows a Poisson distribution [Nak05]. Hence, we can model the noise σ as shown in Equation 3.18, where N_{sig} is the number of photons falling on a pixel, K, K_n are constants and pix is the pixel size of the sensor used in the inspection tool.

$$\sigma = Noise = K\sqrt{N_{sig}} = K_n pix \tag{3.18}$$

Apart from changing the pixel size for inspection, mask engineers can also adjust sensitivity which is related to the threshold for detection of defects. Increasing sensitivity corresponds to reduction of threshold and greater false defect count. For simplicity, we assume that sensitivity is inversely proportional to threshold. The value of threshold also depends on the background intensity that falls on each pixel, which is proportional to the pixel area. Hence, $T = K_t pix^2/S$, where S is the sensitivity used for inspection. Using this, we can estimate number of false defects as shown in Equation 3.19.

$$FalseDefects = K_a \frac{A}{pix^2} erfc\left(\frac{K_t pix^2/S}{\sqrt{2}K_n pix}\right) = K_a \frac{A}{pix^2} erfc\left(K_m \frac{pix}{S}\right) \tag{3.19}$$

Now since CD and contamination defects are flagged using different algorithms whose sensitivity can be set independently [DMI08], we calculate false defects reported by the two methods separately to get the overall false defect count of the inspection tool as shown in Equation 3.20, where K^{CD}, K^{Con}, K_m^{CD}, and K_m^{Con} are constants that depend on the inspection tool.

$$FalseDefects = \frac{A}{pix^2}(K^{CD} erfc\left(K_m^{CD} \frac{pix}{S^{CD}}\right) + K^{Con} erfc\left(K_m^{Con} \frac{pix}{S^{Con}}\right)) \tag{3.20}$$

We used a commercial maskshop's inspection data from over 800 reticles with inspection area ranging from $8000 - 15000mm^2$, pixel size ranging from $72 - 250nm$, and sensitivities ranging from $75 - 100$ to fit these parameters to get $K^{CD} = 30.33$, $K^{Con} = 34.21$, $K_m^{CD} = 0.071$ and $K_m^{Con} = 12.99$ if the inspected area is taken in mm^2 and the pixel size in nm.

3.4.2.2 Nuisance Defects

The number of nuisance defects depends on the design and the total number of real defects, which are the non-nuisance defects. Assuming that the defect distribution for a reticle follows the same negative binomial distribution as wafer defects[10], we can derive a model for the total number of real defects for a reticle of area A, inspected with pixel size p and sensitivities S^{CD} and S^{Con} using Equation 3.21.

$$\begin{aligned} RealDefects &= A \times \sum_{DefectTypes} \int_{R_{min}}^{\infty} \frac{K_2}{D^{\beta}} dD = A \times \sum_{DefectTypes} \frac{K_2}{\beta - 1} \left(K_c \frac{p}{S} \right)^{\beta - 1} \\ &= A \times T^{CD} \left(\frac{p}{S^{CD}} \right)^{\beta^{CD} - 1} + A \times T^{Con} \left(\frac{p}{S^{Con}} \right)^{\beta^{Con} - 1} \end{aligned} \tag{3.21}$$

The constants were fitted using the same maskshop data used to fit false defects to obtain $T^{CD} = 0.0002555$, $\beta^{CD} = 1.3$, $T^{Con} = 0.00008208$ and $\beta^{Con} = 0.88$. Note that this measure of real defects considers both critical and nuisance defects.

3.4.3 First Pass Yield

First-pass yield of masks is defined as the number of masks that can be passed without any repair/review. This is the most important metric for mask shops as it strongly dictates the manufacturing cost of masks. In this section, we develop a simple critical area based methodology to estimate first pass yield.

Let us assume that the defect distribution on the mask is $P(r)$, which corresponds to the probability of a defect to be of size r. Let the maximum defect size on the mask be x_M. Let us also assume that the spatial distribution of defects on the mask

[10]Though there is no published study of reticle defect distribution (to the best of our knowledge), the similarity of mask writing process to wafer patterning suggests a similar defect distrbution.

is uniform. If a mask area A_M was inspected at a single resolution R, then the probability that a given defect will be detected by the inspection tool is given by P_{dd} in Equation 3.22. Assuming $P(r)$ to be inversely proportional to r^3 (similar to wafer defect distribution) [LD97] and $x_M \rightarrow \infty$, we can calculate P_{dd} as shown in Equation 3.23 and Equation 3.24. We can then calculate the expected number of defects, N_d and consequently the expected number of detected defects, N_{dd} as shown in Equation 3.25 and Equation 3.26, respectively, where D_{avg} is the average number of defects on the reticle. Assuming that the number of potential defect sites are very large, we can treat the number of detected defects as a Poisson distribution with expected value N_{dd}. Hence, the probability of detecting no defects is given by Equation 3.27.

$$P_{dd} = \int_R^{x_M} P(r).dr \tag{3.22}$$

$$P(r) = \frac{K}{r^3} \tag{3.23}$$

$$P_{dd} = \frac{K}{2}\frac{1}{R^2} \tag{3.24}$$

$$N_d = D_{avg} A_M \tag{3.25}$$

$$N_{dd} = P_{dd} N_d \tag{3.26}$$

$$P_Y^{A_M} = \exp^{-N_{dd}} \tag{3.27}$$

If the total mask area is partitioned into N regions of area $A_{M1}, A_{M2}, ...A_{Mk}$, each of which is inspected at a different pixel size and sensitivity (different resolution), then the first pass yield of the full mask can be expressed as a product of the probabilities of no defect detection from any partition as shown in Equation 3.28.

$$First \quad Pass \quad Yield = \prod_{k=1}^{k=N} P_Y^{A_{Mk}} \tag{3.28}$$

3.5 Partitioning

In this section, we present a method to partition the reticle where each partition is assigned a pixel size and sensitivity such that the number of false+nuisance defects reported by the inspection tool are minimized without missing any critical defects.

We wish to perform inspection of different regions of the reticle at different pixel size and sensitivities. When we perform inspection for a particular pixel size, partitions marked for inspection at a different pixel size must be labelled as DNIRs (Do Not Inspect Regions) during the current pixel size inspection. DNIR rules specify that a DNIR can be as small as one pixel but there is a forty pixel band in each direction that is not inspected. For our partitioning problem this essentially means that a partition must have dimensions of at least 80 pixels (recall that multiple pixel sizes are implemented as multiple scans with DNIRs). For simplicity we assume the same partition for both pixel size and sensitivity and use the largest pixel size in our experiments to define minimum dimension of a partition. Hence, the design-aware reticle partitioning problem can be formally stated as:

Given a reticle with minimum size defect for each feature, create a partitioning such that a partition j of width W_j and height H_j is assigned a pixel size p_j, and sensitivities S_j^{CD}, S_j^{Con} such that the following function is minimized:

$$F = FalseDefects + \gamma_1 RealDefects \tag{3.29}$$

and the following constraints are obeyed:

- *Minimum dimension constraint:*

$$\min(W_j, H_j) > L_{min} \tag{3.30}$$

- *For any feature with minimum size defect D_{CD} and D_{Con} lying in the j^{th} block of the partition:*

$$D^{CD} > K_c \frac{p_j}{S_j^{CD}}, \quad D^{Con} > K_c \frac{p_j}{S_j^{Con}} \tag{3.31}$$

where γ_1 and γ_2 are weighting factors of the cost function and L_{min} is the minimum dimension constraint.

We use a recursive partitioning heuristic to reduce the defect count metric. At any iteration, if we have k partitions, we find an optimal vertical or horizontal split line that minimizes the defect count for each of the k partitions. Computing the

false/nuisance defect cost for any partition requires scanning the entire partition to find the feature with the minimum defect size assigned. This value of tolerance dictates the resolution for inspection. We then pick the pixel size, sensivitity option that minimizes the cost while keeping the inspection resolution equal to the minimum defect size. Note that splitting a partition can never decrease the total defect count. If both new partitions after splitting have the same value of minimum tolerance then the cost remains the same. If one of them has a higher tolerance then it can be inspected at a lower resolution which would reduce false/nuisance defect count. If a partition has reached the minimum dimension constraint (L_{min}) or no split line reduces the cost then we mark that partition as "optimized" and do not analyze it in any future iterations. This step helps to improve runtime. To locate the optimal split line for any partition, we exhaustively search all the potential horizontal and vertical lines at increments of L_{min} and pick the line which minimizes the cost. The algorithm terminates when all partitions have been labelled as "optimized" or a fixed maximum number of iterations have been reached. The overall algorithm has been summarized in Algorithm 2.

3.6 Experimental Results

3.6.1 Non-Functional Feature Finding

We implement our neighborhood graph based algorithm to identify redundant vias and dummy fill in C++ using OpenAccess (OA) API [oa]. Layouts of some benchmark circuits implemented in $45nm$ Nangate OpenCell library along with the insertion of double vias and dummy fill was done in Cadence Encounter [enc08]. OPC was performed on the generated GDSII files using Mentor Calibre [cal08]. All the implementation was done on a 2.0 GHz Intel Xeon machine with 4 GB memory.

The size of the post-OPC benchmark layouts that we considered along with improvement in the rectangle count due to shape simplification is shown in Table 3.6. The threshold for bucketing was taken as $20nm$, which is less than the minimum

Algorithm 2 Design-aware reticle partitioning for inspection

Require: Criticality value of each shape of reticle, minimum dimension constraint L_{min}.

Define partition array P.
Initialize P with one partition, the full reticle.
while $iter < MAX - ITER$ AND $num_{opt} < size(P)$ **do**
    for all $p_i \in P$ **do**
        if p_i NOT "optimized" **then**
            Find minimum cost split line SL for pi which reduces cost by $\Delta Cost$.
            if $\Delta Cost > 0$ **then**
                Partition p_i using SL to get new partitions p_{iA} and p_{iB}.
                Insert p_{iA} and p_{iB} into P.
            else
                p_i is "optimized".
            end if
        end if
    end for
    Count number of "optimized" partitions in P, num_{opt}.
    $iter++$.
end while

metal width for $45nm$ FreePDK [fre09]. Around 50% reduction in number of shapes is observed for the three benchmark circuit layouts considered.

Table 3.7 summarizes the results of redundancy finding for all the layers. Notice that the benchmark design aesCipher takes a long time despite the small number of rectangles. This is because the design is heavily congested and has a very large number of edges in the neighborhood graph. Runtime can be easily improved by partitioning the layout into smaller blocks and using a separate graph for each region. The algorithm can also be parallelized easily by running the critical graph construction step for each set M_iV_j in parallel. These techniques are left for future work.

The number of redundant vias and dummy fill reported by our approach are verified with the number obtained from DEF file of the corresponding design. Double vias are reported with 100% accuracy by our approach and there is less than 1% error in dummy fill due to some outliers.

Table 3.6: Rectangle count before and after shape simplification for all layers

Design Name	# Gates	Area (um^2)	# Rectangles (before)	# Rectangles (after)
Mips	19983	15947	2582260	1591124
AesCipher	11395	19678	2893906	1850315
Nova	62800	169628	20621302	13626203

Table 3.7: Non-functional feature finding results

Design Name	# Double Vias	# Dummy Fill	Runtime (min.)	Memory Usage (MB)
Mips	23562	20040	3	1212
AesCipher	24267	5308	683	1143
Nova	156774	144727	135	5888

Table 3.8 shows the percentage non-critical regions for the benchmarks which indicates the potential benefits that can be derived from design-aware inspection of metal and via layers. For contact and via layers, we mention the total number of

contact/vias along with the number of redundant ones. Contact layer has redundancy only at the standard cell level. Since very few Nangate cells have redundant contacts, the number of redundant contacts is very low. For metal layers, dummy area is reported as a percentage of the total die area. Higher metal layers typically have less congestion after routing and hence have a greater percentage of dummy area. Note that we have not considered active or polysilicon fill and hence the criticality assignment of polysilicon and active layers is based on the slack of timing critical paths only.

Table 3.8: Layer by layer non-critical regions

Design	Via Layer	# Vias	# Redundant	Metal Layer	% Dummy Area
Mips	Contact	171272	6	Metal1	3.47
	Via1	30737	3140	Metal2	26.54
	Via2	40715	29592	Metal3	26.68
	Via3	15731	9874	Metal4	42.29
	Via4	7666	3056	Metal5	53.37
	Via5	2811	1468	Metal6	81.69
AesCipher	Contact	190230	0	Metal1	0.00003
	Via1	46857	8987	Metal2	4.0
	Via2	52950	27461	Metal3	4.9
	Via3	27872	10641	Metal4	11.8
	Via4	15854	2022	Metal5	14.3
	Via5	10287	1650	Metal6	31.9
Nova	Contact	1399817	62	Metal1	0.0009
	Via1	237337	22878	Metal2	19.97
	Via2	300009	206249	Metal3	19.69
	Via3	104190	58682	Metal4	43.14
	Via4	39005	17708	Metal5	58.54
	Via5	14897	8054	Metal6	83.63

3.6.2 Criticality Assignment & Reticle Partitioning

For assigning minimum size defect to each layout feature, we use $45nm$ design rules from FreePDK [fre09]. Location of dummy fill and redundant via was used for metal and via layers, respectively. For polysilicon and active layers, we need slack values which was obtained from the timing analysis of the post-routed design using Cadence Encounter [enc08]. The criticality assignment method assumes MEEF=1 unless otherwise stated. The tolerance guardband, α_{cycle}, used for criticality assignment of polysilicon and active layers is taken as 1% of the design cycle time

Using the criticality assignment, reticle partitioning was implemented in C++ using OpenAccess API [oa]. From the fitting results of false and nuisance defects, we found that the false defect count is typically at least 10× the number of nuisance defects. But nuisance defects are more important to maskshops as they help improve first pass yield. Hence, we took $\gamma_1 = 10$ in the cost function for these experiments. Only two pixel sizes, $72nm$ and $90nm$, were used in our experiments. The minimum dimension constraint was taken as $2\mu m$, which is slightly larger than dimension of 80 pixels at $90nm$ pixel size.

We tested our partitioning algorithm for poly, active, contact and all the back-end layer reticles for the same three designs for which the non-functional features have been reported. Since the designs we consider are very small compared to real reticle sizes, we find out the number of copies of these benchmark designs which can fit on an industrial reticle ($104mm \times 132mm$). Since defect count is proportional to inspection area, the false and real defect count of one design layout are scaled by the number of copies of the design on a full field reticle in order to demonstrate the potential benefits of our approach for a full field reticle. Table 3.9 shows these results. The false and real defect count after partitioning is compared to the case where inspection is done at a single value of pixel size and CD and contamination sensitivities for the entire reticle.

The results of Table 3.9 demonstrate the reduction of false and real defects with our design-aware inspection methodology. *Note that the improvement in the real*

defect count is due to the reduction in nuisance defects only as the partitioning method is constrained to not miss any critical defects. Highly congested layers with very few non-critical features such as the lower metal/via layers show little benefit of design-aware inspection since most areas of the reticle are very critical. Polysilicon and active layers show significant improvement due to different timing criticality of various features. Higher via and metal layers, which have a significant amount of redundancy as shown in Table 3.8, show up to 4× improvement in false and real defect count. *Note that the metal/via layer processing does not require any explicit timing information while the polysilicon layer leverages it heavily.*

Inspection tools typically allow inspection of different mask regions at different sensitivities in a single scan of the mask. But inspection at different pixel sizes implies that the reticle needs to be scanned multiple times. Hence, it is important to evaluate the additional benefit achieved by varying the pixel size over the reticle area. Table 3.10 compares the false and nuisance defect count after partitioning using both pixel size and sensitivity as parameters versus using sensitivity as the only tunable parameter (pixel size taken as $72nm$). The results show that pixel size is a significant knob and using it might be worthwhile despite the need for multiple scans of the reticle. Note that we retain the minimum partition size constraint for sensitivity-only case even though it is not a strict requirement as too many partitions are impractical to store in the inspection tool.

3.6.3 First-Pass Yield

Using the formulation for first pass yield described in Section 3.6.3, we can estimate the first pass yield if the entire reticle is inspected at a single resolution and compare it to the design-aware approach of inspecting different regions of the mask at different resolution. A reticle is assumed to yield only when all copies of the design on the full field reticle work. Table 3.11 shows the result of this computation for the three benchmark designs we partitioned. Notice that the improvement in first pass yield

Table 3.9: Improvement in defect count after partitioning

Design Name	Layer	Before		After		Runtime
		# False	# Real	# False	# Real	(sec.)
Mips	Polysilicon	70.59	5.52	39.95	3.48	24
	Active	70.59	5.52	38.16	2.37	17
	Contact	66.60	3.36	60.44	3.05	157
	Via1	66.60	3.36	56.78	2.95	64
	Via2	65.94	3.17	57.84	2.83	82
	Via3	65.94	3.17	47.90	2.46	68
	Via4	56.83	2.07	30.86	1.25	30
	Via5	56.83	2.07	14.02	0.65	30
	Metal1	42.62	2.80	42.62	2.80	18
	Metal2	42.20	2.57	39.37	2.40	30
	Metal3	42.20	2.57	40.40	2.46	15
	Metal4	36.37	1.14	32.38	1.02	8
	Metal5	36.37	1.14	32.56	1.02	8
	Metal6	36.37	1.14	22.55	0.72	8
AesCipher	Polysilicon	74.95	5.83	52.60	4.44	42
	Active	74.95	5.83	60.72	4.44	22
	Contact	70.71	3.54	70.71	3.54	129
	Via1	45.24	2.98	44.06	2.91	26
	Via2	44.76	2.71	44.48	2.71	36
	Via3	44.76	2.71	42.54	2.57	20
	Via4	60.30	2.22	33.31	1.04	10
	Via5	60.30	2.22	28.73	0.90	8
	Metal1	45.24	2.98	45.24	2.98	67
	Metal2	44.76	2.71	44.76	2.71	33
	Metal3	44.76	2.71	44.76	2.71	41
	Metal4	38.58	1.18	38.24	1.18	22
	Metal5	38.58	1.18	38.10	1.18	15
	Metal6	38.58	1.18	35.39	1.11	15
Nova	Polysilicon	73.86	5.78	44.77	3.90	2132
	Active	73.86	5.78	45.74	2.98	801
	Contact	69.69	3.51	67.61	3.40	7684
	Via1	69.69	3.51	42.02	2.76	1217
	Via2	44.16	2.68	42.08	2.56	1000
	Via3	68.99	3.32	35.19	2.15	542
	Via4	59.46	2.17	19.05	0.65	272
	Via5	59.46	2.17	8.89	0.36	548
	Metal1	44.60	2.93	44.60	2.93	2444
	Metal2	44.16	2.68	43.06	2.62	981
	Metal3	44.16	2.68	43.38	2.63	509
	Metal4	38.05	1.19	35.78	1.13	236
	Metal5	38.05	1.19	33.61	1.06	141
	Metal6	38.05	1.19	22.03	0.70	195

Table 3.10: Comparison of pixel size and sensitivity (P+S) partitioning versus sensitivity only (S) partitioning

Design Name	Layer	P+S partitioning		S partitioning	
		# False	# Real	# False	# Real
Mips	Polysilicon	39.95	3.48	62.34	3.48
	Active	38.16	2.37	58.82	2.37
	Contact	60.44	3.05	60.44	3.05
	Via1	56.78	2.95	57.75	2.95
	Via2	57.84	2.83	59.47	2.83
	Via3	47.90	2.46	49.98	2.46
	Via4	30.86	1.25	33.97	1.25
	Via5	14.02	0.65	17.05	0.65
	Metal1	42.62	2.80	66.60	2.80
	Metal2	39.37	2.40	61.52	2.40
	Metal3	40.40	2.46	63.13	2.46
	Metal4	32.38	1.02	50.59	1.02
	Metal5	32.56	1.02	50.87	1.02
	Metal6	22.55	0.72	35.23	0.72

correlates well with the reduction in real defect count in Table 3.9. For example, contact and lower metal/via layers show little improvement in first pass yield whereas poly, active and higher via layers show an increase of 30% in first pass yield in some cases. The only exception to this similarity is the higher metal layers, which have a high first pass yield even with the conventional approach and hence only a small improvement in first pass yield. The reason for this is the relaxed design rules of the higher metal layers.

3.6.4 Accouting for non-unity MEEF

Our previous results have assumed that the inspection tool can report MEEF-adjusted defect dimensions [DMI08], hence we used $MEEF = 1$. In this section, we do not rely on this feature and instead use a single worst-case value of MEEF for each reticle layer during criticality assignment. Computing the MEEF value separately for each feature would be more accurate but due to the minimum partition size limitation of inspection tools, it is unlikely to yield much benefit. Hence, we chose the simplistic approach of applying a single pessimistic MEEF correction for each layer.

In order to compute MEEF we used Mentor Calibre [cal08] which can compute MEEF value for each layout fragment separately. Since typical MEEF values are a function of technology node and OPC recipe, we computed the MEEF for different layers of a $10\mu m \times 10\mu m$ snippet of post-OPC Mips layout and used the worst-case value across all fragments of this snippet layout. MEEF was computed using this approach at two defocus values, $0nm$ and $50nm$ and the worst case value is chosen. The MEEF values for different layers are shown in Table 3.12. These values are larger than previously reported data from the industry [KSJ06, XCP08, CJH06]. This is due to the lack of a well optimized optical correction recipe, that should include SRAF insertion and retargeting. Despite this limitation, these MEEF values can be used to validate the feasibility and potential benefit of our approach in the presence of high MEEF mask features.

Table 3.11: Improvement in first pass yield (FPY) with design-aware mask inspection.

Design Name	Layer	FPY before (%)	FPY after (%)
Mips	Polysilicon	12.74	33.72
	Active	12.74	51.71
	Contact	29.17	32.69
	Via1	29.17	34.00
	Via2	31.21	36.08
	Via3	31.21	40.94
	Via4	42.92	61.81
	Via5	42.92	79.14
	Metal1	49.32	49.32
	Metal2	54.36	56.63
	Metal3	54.36	55.79
	Metal4	85.87	87.31
	Metal5	85.87	87.25
	Metal6	85.87	90.98
AesCipher	Polysilicon	11.22	23.50
	Active	11.22	21.38
	Contact	27.04	27.04
	Via1	47.22	48.19
	Via2	52.36	52.60
	Via3	52.36	54.08
	Via4	40.74	86.65
	Via5	40.74	87.80
	Metal1	47.22	47.22
	Metal2	52.36	52.37
	Metal3	52.36	52.38
	Metal4	85.07	85.19
	Metal5	85.07	85.25
	Metal6	85.07	86.30
Nova	Polysilicon	11.58	29.55
	Active	11.58	41.14
	Contact	27.55	28.62
	Via1	27.55	49.81
	Via2	52.85	54.46
	Via3	29.57	60.01
	Via4	41.27	90.00
	Via5	41.27	93.17
	Metal1	47.73	47.73
	Metal2	52.85	53.69
	Metal3	52.85	53.44
	Metal4	85.26	86.10
	Metal5	85.26	86.87
	Metal6	85.26	91.18

Table 3.12: Improvement in defect count and first pass yield after partitioning when MEEF correction applied

Design Name	Layer	Worst-case MEEF	Before			After		
			# False	# Real	FPY	# False	# Real	FPY
Mips	Polysilicon	2	70.59	5.87	8.73	64.15	5.34	10.91
	Active	2	70.59	5.87	8.73	63.47	4.71	15.39
	Contact	6.5	70.59	5.87	8.73	64.06	5.33	10.94
	Via1	5.5	70.59	5.87	8.73	61.21	5.15	11.70
	Via2	20	70.59	5.87	8.73	63.67	5.31	11.01
	Via3	2.5	70.59	5.87	8.73	53.50	4.56	14.85
	Via4	2	65.94	3.17	31.21	35.80	1.95	50.03
	Via5	1.5	63.30	2.65	36.91	15.62	0.85	74.91
	Metal1	9	70.59	5.87	8.73	70.59	5.87	8.73
	Metal2	6.5	70.59	5.87	8.73	65.85	5.48	10.28
	Metal3	6	70.59	5.87	8.73	67.58	5.63	9.69
	Metal4	9	70.59	5.87	8.73	62.84	5.23	11.41
	Metal5	6.5	70.59	5.87	8.73	63.18	5.26	11.28
	Metal6	4.5	70.59	5.87	8.73	43.76	3.65	22.06

After correcting for the MEEF values of different layers during criticality assignement, results obtained from partitioning are shown in Table 3.12 for a Mips design. Note that the defect count and first pass yield before partitioning are worse compared to the case when $MEEF = 1$. This is expected since the minimum tolerable defect size across the entire reticle worsens due to larger MEEF values. Despite the pessimistic and high MEEF values, the results demonstrate the benefits of our partitioning based design-aware inspection.

3.7 Conclusions

In this chapter we developed a comprehensive design-aware mask inspection flow. A summary of the key contributions of this chapter is as follows:

- We proposed a graph based algorithm that finds non-functional features (dummy fill and redundant vias) in a post-OPC layout with almost 100% accuracy.
- We formulated a method to assign a minimum size defect to each feature of a reticle for poly, active, contact and all the back end layers.
- We developed a recursive partitioning algorithm to inspect different regions of the layout with different pixel size and sensitivity and up to 4× reduction in nuisance and false defects was observed along with up to 4× improvement in first pass yield coming from reduction in nuisance defects.

We also demonstrated the importance of pixel size as a paramater in achieving the full benefit of design-aware inspection. Despite the overhead of additional scans of the reticle for each pixel size, the significantly lower defect count suggests that it is a parameter that needs to be exploited in any design-aware inspection flow.

The design-aware methodology that we proposed can be applied easily by captive mask shops since they have access to the design database. Merchant mask shops would need additional information from their customers in the form of either timing report for front-end layer reticles or the database of all the back-end layers so that redundant and dummy features can be identified. In case mask shops cannot get access to design database and are limited to die-die inspection mode, the criticality partitioning can be done at the design end.

In the future, we plan to test our approach in an actual commercial mask shop and explore the implications of our methodology if all mask layers are not available. Only amplitude defects are dealt with in this work. We plan to extend this methodolody to model the design impact of phase defects as well.

CHAPTER 4

Benchmarking of Mask Fracturing Heuristics

4.1 Introduction

Photomasks are one of the most significant contributors to semiconductor manufacturing cost. The use of aggressive resolution enhancement techniques (RETs) has made mask manufacturing extremely expensive and challenging. Moreover, the number of critical masks required for a particular design has increased due to the use of multiple patterning. As a result, controlling the cost of mask manufacturing is urgently needed to sustain benefits derived from Moore's-Law scaling of patterning technologies.

Masks are fabricated using variable-shaped electron beam (VSB) writing tools. These tools directly expose *shots*, i.e., axis-parallel rectangles of different sizes. Mask fracturing is used to obtain a set of shots from the mask pattern, which can then be input to a VSB tool. Since the total shot count strongly affects mask fabrication time, the key objective of mask fracturing tools is to minimize the number of shots.

Traditionally, mask fracturing has been formulated as rectilinear polygon partitioning, which is a very well-studied problem. Imai and Asano propose an $O(n^{1.5} \log(n))$ algorithm to optimally partition a polygon into the smallest number of rectangles [**?**]. Since such theoretical approaches are unable to handle additional manufacturing constraints such as minimization of slivers, Kahng et al. [**?**] propose an ILP based fracturing method. A faster heuristic based on selection of rays from concave corners is also proposed by the same authors [**?**]. Jiang and Zakhor propose a recursive algorithm to minimize a weighted sum of shot count and slivers [**?**]. Ma et al. propose a rectangle combination technique to minimize sliver length along with

shot count [**?**].

Due to aggressive RET techniques such as ILT, mask shapes are now often curved and non-rectilinear [**?**] [**?**]. Fracturing these polygons using traditional methods with acceptable fidelity can dramatically increase the shot count [**?**]. To manage the shot count of such complex patterns, Chua et al. propose model-based fracturing [Chu11], which is also often referred to as model-based mask data preparation (MB-MDP). Two key features of model-based fracturing distinguish it from traditional mask fracturing:

(i) shots are allowed to overlap, which allows greater flexibility in determining shot locations and hence lower shot count; and

(ii) e-beam proximity effects in VSB mask writers are simulated during the mask fracturing itself to ensure that the final pattern on the mask accurately matches the intended target.

A consequence of allowing overlapping shots is that model-based mask fracturing becomes similar to the rectilinear covering problem, which is known to be NP-hard [CR88]. In fact, there is no known constant-factor approximation algorithm for rectilinear covering [Aro03] [**?**]. For polygons which are convex in vertical or horizontal direction, Franzblau and Kleitman propose a quadratic-time algorithm to solve the covering problem optimally [FK84]. But, ILT shapes are rarely convex in vertical or horizontal direction. For hole-free polygons that do not obey this convexity constraint, Wu and Sahni propose several heuristics that guarantee covers that do not have more than twice the optical number of rectangles [**?**]. Franzblau propose an approximation algorithm with strong bounds for any rectilinear polygons [Fra89]. The need to correct for proximity effects means that these methods cannot be used for mask fracturing.

In addition to overlapping shots and proximity effect correction, Galler et al. propose to adjust the dose of each shot independently [**?**]. . The use of L-shaped shots for reducing shot count has been suggested by Yu et al. [YGP13]. The use

of circular shots [FKK10] [**?**] or shots with 45^o edges [**?**] has also been proposed. Elayat et al. [**?**] analyze the benefits and disadvantages of different mask fracturing strategies. They conclude that, among the alternatives studied, model-based mask fracturing with fixed dose is the most viable candidate for improvement of shot count without significant changes in mask writing tools. Hence, in this work we focus on the fixed-dose problem.

Jiang and Zakhor propose an algorithm based on matching pursuit to solve fracturing problem [JZ11]. The same authors also propose a greedy approximate covering algorithm that grows rectangles from convex vertices of a target polygon [JZ14]. Although both these heuristics allow the shot dosage to be adjusted, the method can be extended to solve the fixed-dose problem. Lin et al. present a comparison of a few different heuristics to solve the model-based fracturing problem [**?**]. Although these recent works on model-based mask fracturing have demonstrated improvements in shot count over traditional partitioning-based approaches, the gap between existing methods and optimal solutions remains unclear.

Benchmarking of heuristics used to solve NP-hard EDA problems such as placement [CRX03][**?**], gate sizing [**?**] and partitioning [**?**] enables the development of better methods for solving these problems. The goal of our present work is to enable the benchmarking of model-based fracturing as a foundation for further research towards more effective heuristics. To the best of our knowledge, this is the first work that attempts to benchmark model-based mask fracturing. The key contributions of this work are the following:

- We propose an ILP formulation to optimally solve the model-based mask fracturing problem. We then develop a branch and price method that, in practice, generates strong upper and lower bounds for benchmarking.
- To deal with the slow running time of ILP-based benchmarking, we propose a systematic method to generate benchmarks with known optimal shot count.
- To make the benchmark generation more realistic, we propose an automated benchmark generation method that takes a real ILT shape as input and creates

a benchmark with known optimal shot count that looks similar to the input shape.

- Using the above methods of benchmarking, we evaluate the suboptimality of three mask fracturing heuristics: greedy set cover, matching pursuit and a state-of-the-art prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition.

The rest of this paper is organized as follows. Section 4.2 defines the mask fracturing problem. Section 4.3 describes heuristics for solving the mask fracturing problem that we shall benchmark in this work. Section 4.4 proposes an ILP-based method to obtain tight upper and lower bounds on the optimal shot count. Section 4.5 introduces our method for benchmark generation with known minimum shot count. Section 4.6 presents a method to automatically generate benchmarks with known minimum shot count which are similar to an input mask shape. Section 5.6 concludes the paper. We summarize the key notation used in this work Table 4.1.

4.2 Mask Fracturing Problem

The goal of mask fracturing is to find the minimum number of *rectangular* shots required to construct a mask *target shape*. Although each shot is rectangular, the *e-beam proximity effect* blurs its boundary [Chu11]. As a result, the developed mask pattern is different from the union of rectangular shots. Note that the blurring due to the e-beam proximity effect is significantly smaller than the spacing between different shapes. Hence, each shape in the mask can be fractured independently. *Moreover, to better understand which target shapes are more challenging, the suboptimality of mask fracturing heuristics should be evaluated for individual mask target shapes rather than for the entire mask.*

We define S as the set of all possible candidate shots that could be used to reconstruct the target shape t_{ori}, i.e., the dictionary of candidate shots. S consists of all the different shot sizes ranging from W_{min} to W_{max} with shot size granularity

Table 4.1: Glossary of Terminology

Term	Meaning
S	Set of all possible candidate shots with different sizes and locations to fracture target shape
(x, y)	Coordinates of particular point on the mask
$p(x, y)$	Pixel at coordinate (x, y) for sampled mask shape
σ	Parameter characterizing the spreading of the e-beam
s	Particular candidate shot under consideration
$I(x, y, s)$	Intensity at pixel $p(x, y)$ due to shot s
$W(s), H(s)$	Width and height of shot s
ρ	Threshold value for e-beam resist
t_{ori}	Target mask shape to fracture
γ	CD tolerance limit for fracturing
P_d	Set of pixels lying within distance γ of boundary of t_{ori}
$P_1(P_0)$	Set of pixels within (outside) the closed boundary t_{ori} not belonging to P_d
$S_{min}(t_{ori})$	Minimal set of shots used to fracture t_{ori} (Depends on γ, σ, ρ as well, but not shown for brevity)
$W_{min}(W_{max})$	Minimum (Maximum) shot size allowed
Δ	Shot size granularity
Δ_{MP}	Shot size granularity used in matching pursuit heuristic
$I(x, y)$	Total intensity at $p(x, y)$ due to all shots in current fracturing solution
z_s	$0-1$ indicator whether candidate shot s is part of fracturing solution
$W(t_{ori}), H(t_{ori})$	Width and height of bounding box of t_{ori}
$B_v(B_h)$	Vertical (horizontal) boundary segments of t_{ori}
b_i	i^{th} boundary segment
$t_{ori,i}$	i^{th} split-shape of t_{ori}
λ_p^*	Value of dual variable at pixel $p(x, y)$
P_{neg}	Set of pixels for which $\lambda_p^* \leq 0$
$(x_{bl}(s), y_{bl}(s))$	Bottom left coordinate of candidate shot s
$(x_{tr}(s), y_{tr}(s))$	Top right coordinate of candidate shot s
N_C	Maximum number of candidate shots inserted in one pricing round
α	Maximum distance outside target boundary where candidate shot corner can lie without exposing any pixel in P_0
β	Maximum distance outside shot at which the intensity is nonzero
B^n	Set of all boundary segments that require at least n shots to construct

of Δ and shifted copies of each shot size.[1] E-beam proximity effect is modeled using a low pass filter, typically a Gaussian or sum of Gaussians [Pav86]. In this work, we model the proximity effect by a single 2D Gaussian low-pass filter, described by Equation (4.1). However, our proposed methods for benchmarking can be easily extended to handle other proximity effect models.

$$K(x,y) = \begin{cases} \frac{1}{F}\exp^{-\frac{x^2+y^2}{\sigma^2}} & \text{if } -3\sigma \leq \sqrt{x^2+y^2} \leq 3\sigma \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Here, $K(x,y)$ is the kernel function of the Gaussian filter, F is a normalization factor (i.e., the sum of $K(x,y)$ across all values of x and y) and σ is a parameter which characterizes the spreading of the e-beam. For any rectangular shot s, the intensity at a pixel can be computed by convolving the *ideal rectangular function* ($\psi(\hat{x},\hat{y})$)[Bra65] with the kernel function. That is,

$$\begin{aligned} I(x,y,s) &= K(x,y) \otimes \psi(\frac{(x-x_{c,s})}{W(s)}, \frac{(y-y_{c,s})}{H(s)}) \\ \psi(\hat{x},\hat{y}) &= \begin{cases} 1 & \text{if } |\hat{x}| < 0.5 \text{ and } |\hat{y}| < 0.5 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.2}$$

where $x_{c,s}$ and $y_{c,s}$ are the x and y coordinates of the center of the shot. In this paper, all dimensions are in wafer scale.[2]

We model the e-beam resist using a constant-threshold model with threshold value of ρ. Any pixel ($p(x,y)$) on the mask will be exposed if and only if the total intensity at that pixel resulting from all shots exceeds the resist threshold ρ.[3]

As shown in Figure 4.1, we divide the set of pixels on the mask into three disjoint sets: P_1, P_0 and P_d. The pixels in P_1 must have intensity greater than or equal to ρ. Similarly, we define P_0 as the set of the pixels outside the target shape which do not belong to P_d. The pixels in P_0 must have intensity less than ρ.

[1] The step size of shifting is Δ.
[2] Typically, mask scale is 4× wafer scale.
[3] The exposed pixels will form the mask shape.

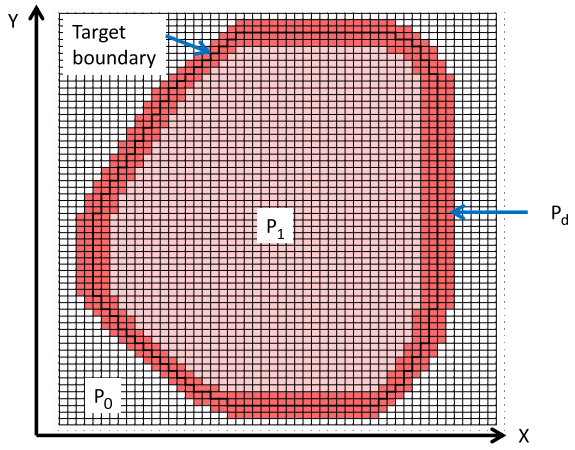


Figure 4.1: Each grid in the figure is a pixel $p(x, y)$. The thick black line is the target boundary. In this figure, the CD tolerance is $\gamma = 2nm$ and the grid size is $1nm \times 1nm$. $p(x, y) \in P_d$ if $p(x, y)$ is within $2nm$ of the target boundary.

The mask fracturing problem is formally defined as follows.

Goal: Minimize the total number of mask shots $N = |S_{min}(t_{ori})|$.

Inputs: Mask target shape, set of all candidate shots S, ρ, σ, γ.

Outputs: Set of rectangular shots, $S_{min}(t_{ori})$.

Constraints:

$$\begin{aligned} \sum_{s \in S_{min}} I(x, y, s) &\geq \rho \text{ if } p(x, y) \in P_1 \\ \sum_{s \in S_{min}} I(x, y, s) &< \rho \text{ if } p(x, y) \in P_0 \end{aligned} \tag{4.3}$$

CD control of the target pattern is a key concern for mask manufacturing. To minimize CD variation, any critical vertical or horizontal segment of the target shape boundary should not be constructed with more than one shot [**?**]. This issue is illustrated in Figure 4.2. To check if a particular candidate shot satisfies the CD control constraint, we first identify critical vertical or horizontal regions of a given target shape (i.e., long and narrow part of a target shape). Then, any candidate shot that overlaps with these horizontal (vertical) critical regions must be such that both it's vertical (horizontal) edges touch the target boundary. Only candidate shots that satisfy this criteria of CD control are included in the set S.

The mask writing process may also require additional constraints to avoid resist

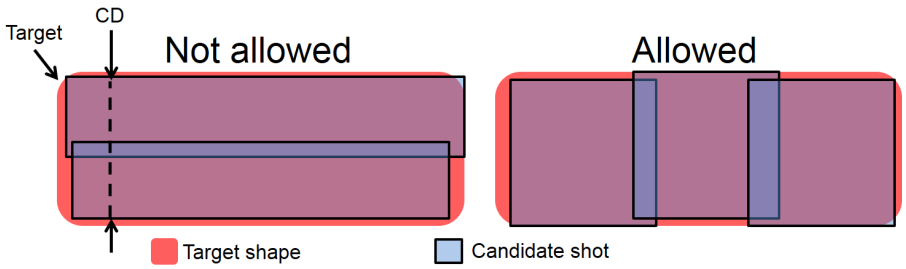


Figure 4.2: Illustration of CD control constraint for candidate shots for a horizontal critical region.

over-heating. In this work, we do not consider the imposition of maximum intensity constraints to model resist over-heating, since the over-heating is an effect at length scales on the order of microns [**?**][**?**].

4.3 Fracturing Heuristics

To evaluate the suboptimality of different mask fracturing heuristics, we have developed two simple methods to fracture mask shapes, based on prior work, that we describe in this section. The fracturing solutions created by both these heuristics tend to have CD violations, i.e. pixels that violate Constraint 4.3. Hence, we use an additonal step, *shot refinement*, to fix the CD violations. In addition to the two simple heuristics, we evaluate the suboptimality of a prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition (PROTO-EDA).

Greedy set cover (GSC) heuristic is inspired by the well-known greedy approximation algorithm for the NP-complete set cover problem [Chv79]. We first construct an Hanan grid by constructing X- and Y-axis parallel lines from each vertex of the target polygon. Then every grid element that lies inside the polygon and contains at least one pixel $p(x, y) \in P_1$ is considered an 'element' of the set cover problem. We then find all the maximal rectangles lying inside the polygon [4]. Each maximal rectangle is treated as a 'set' that covers some of the grid elements. The fracturing problem then reduces to the set cover problem to which we apply the greedy approximation algorithm. Note that the e-beam proximity model is not considered

[4] A rectangle is maximal if all four edges touch the boundary of the target polygon.

in constructing this fracturing solution but is handled during the subsequent shot refinement.

Matching pursuit (MP) is a well-known technique to represent a signal sparsely for an over-complete basis set [**?**]. Jiang and Zakhor [JZ11] propose a technique to apply this method to the mask fracturing problem. A dictionary of different shot sizes is first constructed. To keep the dictionary size tractable, some step size $\Delta_{MP} \geq \Delta$ is used to discretize the width/height range between W_{min} and W_{max}. The proximity model is applied to each shot in the dictionary. Then we iterate over the dictionary and over all potential positions of the candidate shot to pick the shot that reduces the *residual error* the most. This procedure is repeated until no shot is found that can reduce the residual error. We define residual error as the sum of $|I(x, y) - \rho|$ for all the pixels which violate the CD constraint.

Shot refinement works by moving the edges of the shots greedily to minimize the residual error. Once this greedy procedure stops reducing the residual error, we bias all the shots of the current solution by a small value. If the number of pixels in set P_0 that violate the CD constraint (over-cover) are more than the number of pixels in P_1 that violate the CD constraint (under-cover), we shrink all shots otherwise we expand all shots during this bias step. After biasing, we continue with the greedy shot edge adjustment. If this iterative procedure fails to reduce the residual error for several iterations, we add or remove one shot depending on if more pixels are under-covered or over-covered, respectively. We terminate when the residual error is zero, i.e. there are no CD violations. Although this iterative procedure is not guaranteed to find a feasible (CD error-free) fracturing solution, in practice the method works well for most test-cases.

4.4 ILP-Based Benchmarking

To evaluate the suboptimality of fracturing heuristics on any given mask shape, we apply an optimal ILP formulation. The straightforward ILP formulation requires a large number of binary variables, even for small target shapes. As a result, even

commercial ILP solvers can run out of memory on high-performance computers. To circumvent this, we propose three strategies, described in this section: (1) pruning the set of candidate shots, (2) splitting large target shapes, and (3) solving the ILP using branch and price. With these strategies, we can obtain strong upper and lower bounds on the optimal solution within feasible runtime. *Note that although the proposed ILP can be used to inspire effective mask fracturing heuristics, the goal of this work is benchmarking. Hence, runtime is important only to the extent of making the method tractable.*

4.4.1 Optimal ILP Formulation

Inspired by the ILP formulation of Heinrich-Litan et al. [HL07], we propose a simple ILP formulation for the model-based mask fracturing problem. We define a binary selection variable z_s for each candidate shot $s \in S$, where $z_s = 1$ if shot s is used and $z_s = 0$ otherwise. Then, based on the problem description in Section 4.2, we may formulate an optimal ILP to solve the fracturing problem as:

$$\begin{aligned} \text{Minimize} \quad & \sum_s z_s \\ \text{subject to} \quad & \sum_s \{z_s \cdot I(x,y,s)\} \geq \rho, p(x,y) \in P_1 \\ & \sum_s \{z_s \cdot I(x,y,s)\} < \rho, p(x,y) \in P_0 \end{aligned} \tag{4.4}$$

The problem with this ILP formulation is that $|S|$ can be very large even for small target shapes. For a target shape t_{ori} with a bounding box of $W(t_{ori}) \times H(t_{ori})$, if the shot size granularity is Δ and no shots are disallowed due to the CD control constraint, then the size of the set of candidate shots would be $\left(\frac{(W_{max}-W_{min})}{\Delta}\right)^2 \cdot (W(t_{ori}) - \frac{W_{max}+W_{min}}{2}) \cdot (H(t_{ori}) - \frac{W_{max}+W_{min}}{2})$, where W_{min} and W_{max} are the minimum and maximum allowed shot sizes. Even for a small post-ILT contact shape as shown in Figure 4.3a ($W_{min} = 13,\ W_{max} = 55, W(t_{ori}) = H(t_{ori}) = 60,\ \Delta = 1$), the number of candidate shots is $1.19M$.

Attempting to solve such a large ILP, even with commercial solvers, is very challenging due to long runtime and large memory usage. In fact, the

CPLEX v12.5 solver[**?**] runs out of memory when we attempt to solve the instance of Figure 4.3a on an Intel Xeon L5420 server with 128GB RAM.

4.4.2 Pruning Candidate Shot Dictionary

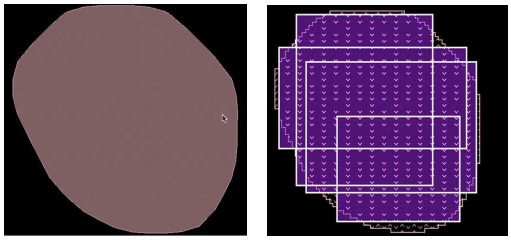
Reducing $|S|$ can significantly help in making the above ILP tractable for benchmarking. Here we highlight two simple rules that can be used to reduce $|S|$:

(i) For any candidate shot s, if there exists a pixel $p(x, y) \in P_0$ such that $I(x, y, s) \geq \rho$, then s can be removed from the set S. This pruning criterion obviously does not affect optimality because any candidate shot that satisfies this condition cannot be a part of a feasible solution of the ILP. Depending on the specific target shape, this pruning strategy can significantly reduce $|S|$.

(ii) If a candidate shot s is inside the target shape and none of its four edges are close to the target boundary, then we remove s from set S. If s is a part of the optimal solution, then we can replace s with a larger shot that covers s and has at least one boundary close to the edge of the target shape, without affecting the optimality of the solution.

An interesting side-effect of pruning candidate shots is that the LP relaxation of the ILP becomes a stronger lower bound for the optimal shot count. After applying these pruning rules to the contact shape of Figure 4.3a we can reduce $|S|$ to $591K$ for a Gaussian proximity effect model ($\sigma = 1nm$). Using *CPLEX v12.5* solver, the final optimal shot count is just four (shown in Figure 4.3b).

Note that the reduction in $|S|$ due to these pruning rules depends strongly on the specific target shape and the e-beam proximity effect model. For many target shapes, the number of variables even after pruning could be more than 10^6, making it difficult to solve the problem efficiently with commercial ILP solvers.



(a) (b)

Figure 4.3: (a) ILT mask target shape and (b) optimal mask fracturing solution obtained from ILP.

4.4.3 Splitting Target Shapes

The size of the mask fracturing ILP depends on the size of the bounding box of the target shape because it affects both the number of variables (candidate shots) and the number of CD constraints (pixels). Hence, the ILP can become prohibitively big for large target shapes. In this subsection, we propose a simple strategy that can be used to split large target shapes. This allows us to solve a separate smaller ILP for each smaller split-shape. The fracturing solution of each smaller ILP can then be aggregated to obtain a solution of the full target shape. Using this solution as the initial solution, the larger ILP corresponding to the full target shape can then be solved.

For splitting any target shape, the key step is determining locations where the shape should be split. We find horizontal and vertical line segments which serve as split locations to obtain smaller target shapes. The procedure we use to determine split locations is illustrated in Figure 4.4. Algorithm 3 describes the procedure we use to obtain horizontal split locations. An analogous procedure can be used to find vertical split locations.

In Algorithm 3, we first identify vertical *boundary segments*[5] of t_{ori} which are longer than L_{th}. Each such vertical boundary segment b_i is characterized by three features: x-coordinate of the orthogonal segment ($b_i.val$), along with y-coordinates of the two end points of the segment ($b_i.low$ and $b_i.high$). Using these, we then

[5]A boundary segment is a contiguous part of the boundary of a target shape.

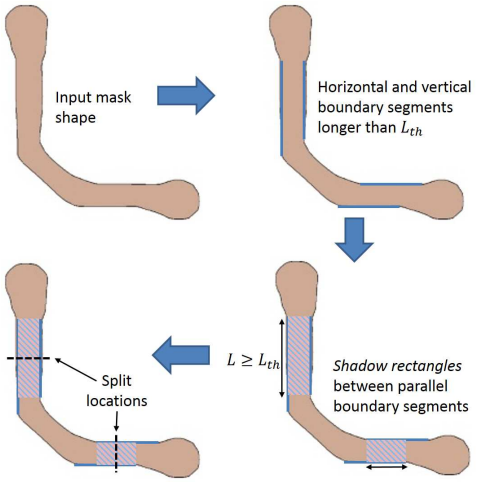


Figure 4.4: Steps involved in splitting a target polygon into smaller polygons.

Algorithm 3 Determine horizontal locations to split target shape

Require: Target shape t_{ori} and length threshold L_{th}

Ensure: Locations where t_{ori} is split

1: $B_v \leftarrow$ all vertical boundary segments longer than L_{th}, sorted by x-coordinate
2: **for all** $b_i \in B_v$ **do**
3: **for all** $b_j \in B_v$ **do**
4: $low \leftarrow max(b_i.low, b_j.low)$
5: $high \leftarrow min(b_i.high, b_j.high)$
6: **if** $b_i \neq b_j \&\& (high - low) > L_{th}$ **then**
7: $(x_{bl}(rect), y_{bl}(rect)) \leftarrow (b_i.val, low)$
8: $(x_{tr}(rect), y_{tr}(rect)) \leftarrow (b_j.val, high)$
9: **if** $rect$ lies inside t_{ori} **then**
10: Split location $\leftarrow$ Line segment from $(b_i.val, \frac{low+high}{2})$ to $(b_j.val, \frac{low+high}{2})$
11: **end if**
12: **end if**
13: **end for**
14: **end for**

find pairs of parallel vertical segments which satisfy the following two conditions; (i) Length of shadow rectangle between the two parallel segments is longer than L_{th} (Lines 5-7). (ii) Any line segment perpendicular to the two parallel segments connecting two point, one on each segment, lies inside the target polygon (Lines 8-10). Then we can split t_{ori} at the center of the shadow rectangle.

Suppose we split a target shape into two target shapes, $t_{ori,i}$ and $t_{ori,j}$ with a horizontal split location, using the steps described above. Then we can bound the minimum shot count of t_{ori} as a function of the minimum shot count of $t_{ori,i}$ and $t_{ori,j}$ using the two lemmas we describe next.

Lemma 4.4.1. $|S_{min}(t_{ori})| \leq |S_{min}(t_{ori,i})| + |S_{min}(t_{ori,j})|$

Proof. If we consider all the shots of an optimal fracturing solution of the two split-shapes $t_{ori,i}$ and $t_{ori,j}$, we extend all the shots that touch the split location vertically, so that these shots from $t_{ori,i}$ and $t_{ori,j}$ overlap at the split location. This modified set of shots is a feasible solution that satisfies all the CD constraints. This feasible solution is clearly an upper bound on the optimal solution of the full shape. Note that this step of combining solutions of the smaller split-shapes is valid if and only if the distance between the horizontal split line from the top and bottom of shadow rectangle is large enough to ensure that any small change to a shot at the split location does not add intensity to any pixel above or below the shadow rectangle. To ensure this, we set the length threshold $L_{th} = 2\beta$. □

Lemma 4.4.2. $|S_{min}(t_{ori})| \geq max(|S_{min}(t_{ori,i})|, |S_{min}(t_{ori,j})|)$

Proof. Without loss of generality, let us assume that $|S_{min}(t_{ori,i})| \geq |S_{min}(t_{ori,j})|$, and $t_{ori,i}$ lies below a horizontal split location. We can prove this lemma using contradiction. Suppose there exists an optimal fracturing solution $S^*_{min}(t_{ori})$ for target shape t_{ori}, such that $|S^*_{min}(t_{ori})| \leq |S_{min}(t_{ori,i})|$. From this solution, we can obtain a feasible fracturing solution for $t_{ori,i}$ by taking all the shots that lie below the split location and cutting all shots that overlap with the split location. Clearly, the number of shots of this fracturing solution $|S^*_{min}(t_{ori,i})| \leq |S^*_{min}(t_{ori})|$ since only

a subset of the shots comprising the fracturing solution of t_{ori} are used. This implies that $|S^*_{min}(t_{ori,i})| \leq |S_{min}(t_{ori,i})|$. This is clearly not possible. Consequently our original assumption must be incorrect, and $|S_{min}(t_{ori})| \geq |S_{min}(t_{ori,i})|$. □

After splitting t_{ori} using the method described above, we solve a separate ILP for each split-shape. We then combine the fracturing solution of the separate ILPs to obtain a feasible fracturing solution for t_{ori}, which we use as a starting solution for the larger ILP for t_{ori}. Note that this splitting technique is effective only if the target shape boundary contains long vertical or horizontal boundary segments.

4.4.4 Branch and Price Method

Branch and price (B&P) is a well-known method for solving large ILPs [**?**]. The key feature that distinguishes B&P from typical ILP solvers is that the LP relaxation at each node of the branch and bound tree is solved using column generation. To solve the LP relaxation, which contains too many variables to handle efficiently, a reduced master problem (RMP) containing only a small subset of the variables is solved first. To confirm the optimality of this RMP, a separate pricing subproblem is solved to find any new variables that must be inserted back into the RMP. If no variable is found by the pricing subproblem, then the LP relaxation is optimal and branching can be done to obtain the integral solution to the original ILP.

The selective insertion of variables based on the pricing subproblem in B&P means that most variables are never inserted into the LP relaxation. As a result, the LP relaxation solver does not consume too much memory. This is the main reason why we choose to apply this technique to solve the ILP described in Equation (4.4). The runtime of B&P is known to be limited by the pricing subproblem for most problems [**?**]. Hence, we propose a novel pricing mechanism comprising a fast, approximate pricer and a slower, optimal pricer.

The goal of the pricing subproblem is to identify additional variables that should be inserted into the RMP. For the mask fracturing problem, let λ^*_p be the optimal value of the dual variable corresponding to the CD constraint (Equation 4.4) at pixel

$p(x,y) \in P_1 \cup P_0$, obtained after an iteration of the RMP. The pricing subproblem (derived from the dual of the RMP) reduces to finding a new candidate shot s such that $\sum_p I(x,y,s) \cdot \lambda_p^* \leq -1$. This candidate shot must also satisfy the pruning rules discussed above. Moreover, additional constraints imposed by the branching rules of the branch and bound tree must be met. The *reduced cost* of any candidate shot s is given by $R_s = 1 + \sum_p \{I(x,y,s) \cdot \lambda_p^*\}$. For the sake of brevity, we shall refer to any candidate shot that has $R_s \leq 0$ and satisfies all the pruning and branching constraints as an *insertable candidate shot* (ICS).

To ensure that the LP relaxation is solved optimally, the pricing subproblem must guarantee that no ICS exists. If there are several ICSs, the pricing subproblem only needs to find a subset of all the ICSs in an iteration. The maximum number of candidate shots that are inserted in each pricing iteration can be tuned to improve the convergence of B&P. In this work, we limit the maximum number of variables that can be inserted in each iteration to $N_C = 500$.

One simple strategy to solve the pricing problem is to iterate over all possible sizes and locations of candidate shots and insert any shot that has a negative reduced cost and satisfies pruning and branching rules. To improve the efficiency of this naive pricing strategy, we carefully analyze the dual variables of the RMP. Based on the well-known Karush-Kuhn-Tucker (KKT) conditions, we note the following key features of the dual variables:

(i) Due to complementary slackness, $\lambda_p^* \neq 0$ if and only if $\sum_s \{z_s \cdot I(x,y,s)\} = \rho$. Since this is likely to occur only close to the boundary of the target shape, λ_p^* is nonzero only for a small number of pixels that lie very close to the target boundary. We shall refer to the set of pixels with nonzero dual values as *dual points*.

(ii) To ensure dual feasibility, $\lambda_p^* \geq 0$ for $p(x,y) \in P_0$ and $\lambda_p^* \leq 0$ for $p(x,y) \in P_1$. This implies that all negative dual points (P_{neg}) are located inside the target shape.

That negative dual points are sparse and are located close to the target shape bound-

ary is illustrated in Figure 4.5 for a particular pricing iteration of a target shape. Based on this insight, we propose two pricing strategies to effectively find ICSs, as we now describe.

4.4.4.1 Fast Pricer

The basic idea behind the fast pricer is to look for ICSs in the vicinity of $p(x, y) \in P_{neg}$ because if any candidate shot s has negative reduced cost, then s must be located such that it covers or is close to at least one negative dual point. The steps involved in finding ICSs are summarized in Algorithm 4.

Algorithm 4 Fast Pricer Heuristic

Require: Target shape t, and list of pixels with negative dual values P_{neg}

Ensure: Set of candidate shots inserted into the RMP

1: **for all** $p(x, y) \in P_{neg}$ **do**
2: Draw vertical/horizontal line from (x, y) to find y_{low}, y_{high}, x_{low} and x_{high} (iillustrated in Figure 4.5)
3: Find all candidate shots in vicinity of (x, y) that satisfy Equation (4.5) below
4: Insert (up to $\frac{N_C}{|P_{neg}|}$) candidate shots that satisfy reduced cost, pruning and branching constraints to RMP
5: **end for**

$$\begin{aligned} x_{low} - \alpha \leq x_{bl}(s) \leq x + \beta \quad , \quad x - \beta \leq x_{tr}(s) \leq x_{high} + \alpha \\ y_{low} - \alpha \leq y_{bl}(s) \leq y + \beta \quad , \quad y - \beta \leq y_{tr}(s) \leq y_{high} + \alpha \end{aligned} \tag{4.5}$$

The intuition behind constraining $x_{bl}(s)$, $y_{bl}(s)$, $x_{tr}(s)$ and $y_{tr}(s)$ as shown in Equation (4.5) is that such candidate shots will have nonzero intensity at the negative dual point under consideration and are likely to obey the first pruning rule (not exposing any pixel in P_1).

4.4.4.2 Optimal Pricer

Although the pricing heuristic we described above is effective in identifying most ICSs which can be inserted into the RMP, it does not guarantee that if no ICS is found, then there does not exist any ICS. Hence, if the heuristic fails to find any ICS, we call the optimal pricer that is guaranteed to find a candidate shot with negative reduced cost, if it exists. The optimal pricer iterates over all candidate shots in

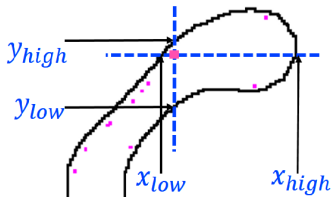


Figure 4.5: Illustration of negative dual points (pink dots) for part of a target shape. Coordinates x_{low}, x_{high}, y_{low} and y_{high} (points of intersection of blue dashed lines with target shape boundary) for a particular negative dual point (point of intersection of the two dashed lines) are also shown.

the vicinity of the negative dual points. The method is described in Algorithm 5. The optimal pricer first constructs constructs square boxes of size $2 \times \beta$ centered at each negative dual point. Any candidate shot which could have negative reduced cost must overlap with at least one of these boxes. Hence, we could iterate over all such candidate shots to find ICSs. But several dual points may lie close to each other which may cause candidate shots to be generated twice. To avoid this, we first merge the boxes using polygon Boolean OR operation. Then we find the bounding box of each resulting polygon. All candidate shots that overlap with these bounding boxes are then checked for insertion into the RMP.

Algorithm 5 Optimal Pricer

Require: Target shape t, and list of pixels with negative dual values P_{neg}.

Ensure: Set of candidate shots inserted into the RMP

1: $mergedBoxes \leftarrow$ new list of polygons
2: **for all** $p(x, y) \in P_{neg}$ **do**
3: $rect \leftarrow$square box of size $2 \times \beta$ with $p(x, y)$ as center
4:
5: $mergedBoxes \leftarrow mergedBoxes \bigvee rect$ (Polygon Boolean OR operation)
6: **end for**
7: **for all** $polygon \in mergedBoxes$ **do**
8: Find bounding box of $polygon$
9: Find all candidate shots that overlap with the bounding box of $polygon$
10: Insert (up to $\frac{N_C}{|P_{neg}|}$) candidate shots that satisfy reduced cost, pruning, CD control and branching constraints to RMP
11: **end for**

4.4.5 Initialization and Overall Summary

In addition to solving the pricing subproblem efficiently, B&P can benefit significantly from a good initial feasible solution. Although B&P is capable of discovering feasible solutions using Farkas pricing [Ach09], it can take many iterations of pricing to do so. In this work, we use the results of the GSC heuristic as the initial solution for B&P.

Although B&P allows us to circumvent the problem of excessive memory usage, it can take a lot of time to converge to the optimal solution. Since our objective is to evaluate suboptimality, we choose to run B&P with a fixed time limit and report the best upper and lower bounds on the optimal shot count. The overall method we use to solve LP relaxations within B&P is summarized in Figure 4.6.

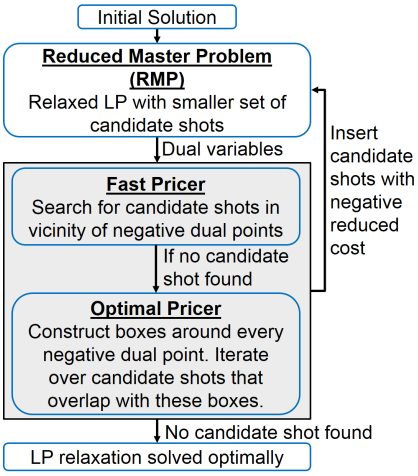


Figure 4.6: Steps involved in solving LP relaxation at any node of the branch and bound tree in B&P.

For each split-shape $t_{ori,i}$, we first obtain an initial solution using GSC and we then solve the ILP using B&P. Based on the upper and lower bounds of each smaller ILP, we can obtain lower and upper bounds for t_{ori} using Lemma 4.4.1 and Lemma 4.4.2, respectively. To further improve these bounds, we combine the solutions from each split-shape by merging shots lying at the split locations. If this does not give a feasible solution for t_{ori}, we apply shot refinement to fix the constraint violations and then use this as the initial solution for solving the larger ILP corresponding to

the t_{ori} using B&P.

To improve the running time of the pricing method, we parallelize both the fast and optimal pricing method. For the fast pricer, Lines 4-5 in Algorithm 4 can easily be parallelized since candidate shot can be checked for insertion to the RMP independently. Similarly for the optimal pricer, Lines 8-9 can be parallelized easily.

4.4.6 Experimental Results

Our B&P based suboptimality evaluation method has implemented in C++. We use the OpenAccess API to parse layouts [oa], Boost Polygon Library to perform polygon operations [**?**] and Eigen Library to perform matrix operations [Gue10]. To implement B&P, we use the SCIP framework [Ach09], along with *CPLEX v12.5* as the LP solver [**?**]. We parallelize the pricing methods using OpenMP.

In this work, we set resist threshold $\rho = 0.5$. We use a Gaussian e-beam proximity effect model with two values of σ, $6.25nm$ and $4nm$ [6]. For CD tolerance γ, we consider values of $2nm$ and $1nm$. The shot dimension constraints are $W_{min} = 13nm$, $W_{max} = 1000nm$ and $\Delta = 1nm$. The pixel size in all our experiments is $1nm$.

We apply ILT to benchmark pre-RET layouts from the 2013 ICCAD contest [**?**], using a commercial EDA tool. From the ILT solutions, we select ten representative mask shapes for evaluation. These benchmarks are illustrated in Figure 4.7.

For each of the 10 target shapes, we run B&P on an eight-core machine with a time limit of 12 hours. Half the time limit is devoted to solving the ILP corresponding to the split-shapes, with the time limit of each split-shape $t_{ori,i}$ proportional to the size of its bounding box. The remaining time limit is spent in solving the larger ILP corresponding to t_{ori}.

In any branch and bound based search method for integer programs, the upper bound corresponds to the best integral solution that has been discovered so far. The lower bound corresponds to the LP relaxation at a particular level of the branch and

[6] $\sigma = 6.25nm$ is consistent with recent work on mask fracturing [JZ11] [JZ14]. Results are shown for $\sigma = 4nm$ to highlight the impact of Gaussian blur on shot count.

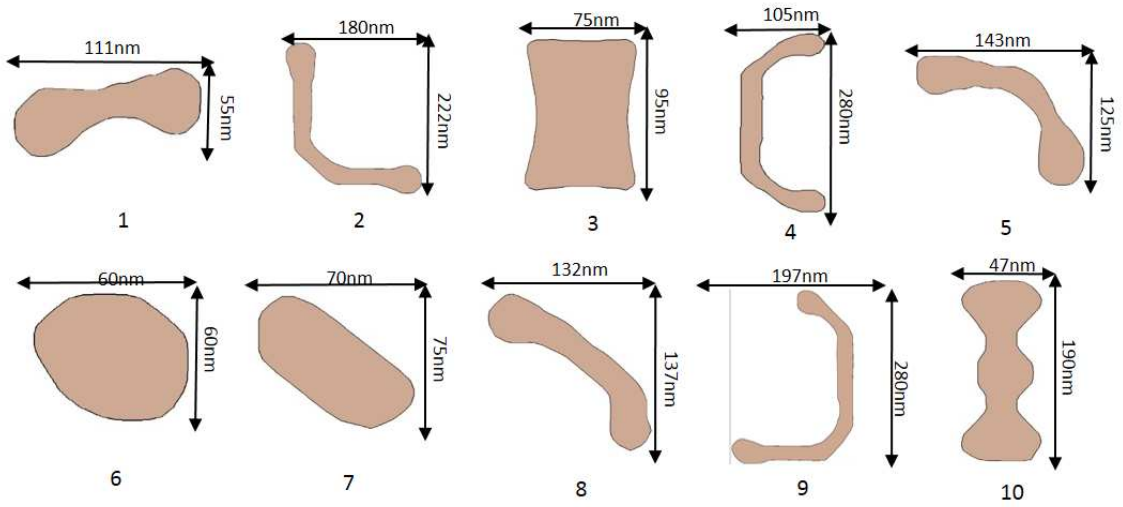


Figure 4.7: ILT mask shapes obtained after applying inverse lithography to layouts from the ICCAD-2013 contest [**?**] (wafer scale).

bound tree. We report the upper and lower bounds reached by B&P within the set time limit.

The shot count and running time of the MP heuristic depends strongly on Δ_{MP} and the value by which each shot size is shifted for finding new shots. Selecting a large value for these parameters reduces the running time, but typically increases shot count. Moreover, it can cause many mask shapes to have CD violations even after shot refinement. For this work, we set $\Delta_{MP} = 15nm$ and shots are shifted by $(\Delta_{MP}/2)$.

We show the results of our ILP-based suboptimality analysis method for actual ILT mask shapes in Table 4.2. For all ten benchmark shapes and three scenarios, our method is able to report a lower bound based on LP relaxation.[7] Although this seems trivial, typical LP methods (simplex and barrier methods) run out of memory while trying to solve the LP relaxation of the ILP in Equation (4.4) for these benchmark shapes. Hence, our B&P based method appears to be enabling to the computation of this lower bound. Moreover, for the baseline case with $\sigma = 6.25nm, \gamma = 2nm$, our method is able to discover a fracturing solution (upper bound) better than any of the heuristics for five of the ten shapes. For two of them, the optimal solution is

[7]The fractional LP relaxation value is rounded up to the next integer to obtain the lower bound.

Table 4.2: Comparison of shot count for ILT mask shapes shown in Figure 4.7 for three different heuristics (GSC, MP and PROTO-EDA) along with lower bound (LB) and upper bound (UB) obtained from Branch and Price (B&P). Shot count is shown for three scenarios with different values of e-beam proximity model Gaussian variance (σ) and CD tolerance (γ).

Clip-ID												
	$\sigma = 6.25nm$, $\gamma = 2nm$				$\sigma = 4nm$, $\gamma = 2nm$				$\sigma = 4nm$, $\gamma = 1$			
	GSC	MP	PROTO-EDA	**B&P LB/UB**	GSC	MP	PROTO-EDA	**B&P LB/UB**	GSC	MP	PROTO-EDA	**B&P LB/U**
1	14	14	7	3/4	14	7	12	4/5	22	20	12	5/10
2	18	13	21	5/9	23	27	31	6/13	47	28	32	8/23
3	5	4	7	3/3	3	9	27	3/3	13	78	27	3/8
4	31	14	21	6/17	40	27	36	7/14	60	52	36	10/3
5	23	25	12	5/13	27	19	22	5/16	40	25	22	8/23
6	9	5	6	3/3	9	5	11	3/4	17	27	11	4/7
7	10	7	8	3/4	15	5	11	3/4	22	33	14	4/10
8	26	9	12	5/17	26	21	21	5/9	44	29	23	8/26
9	39	14	26	7/20	39	22	53	4/19	66	46	54	6/36
10	14	7	11	4/8	15	8	19	4/6	30	16	21	6/16

found.

If we assume that the lower bound reported by the ILP is indeed the optimal shot count, the suboptimality of GSC, MP and PROTO-EDA heuristics ranges from 1.7× to 5.6×, 1.7× to 5.0× and 2.3× to 4.2×, respectively for $\sigma = 6.25nm$ and $\gamma = 2nm$. Although MP is able to achieve lower shot count than the other two heuristics for most shapes, it is significantly slower than GSC. For the above ten shapes, the running time of GSC ranges from $1s$ to $38s$. In contrast, the running time of MP ranges from $11s$ to $787s$.

Since the gap between the optimal solution of an ILP and the LP relaxation can be very large, suboptimality analysis based on the lower bound may be too pessimistic. If we make the optimistic assumption that the integer solutions obtained by the ILP are in fact optimal, i.e. the upper bound is equal to the optimal shot count, then the suboptimality of the GSC, MP and PROTO-EDA heuristics could be as large as 3.5×, 1.7× and 2.3×, respectively. These results suggest that there is significant room for improving the quality of mask fracturing solutions.

Improvements in e-beam mask writing tools are likely to reduce the blurring of shot intensity caused by forward scattering since it helps improve the resolution of the tool. Consequently, the Gaussian model σ is likely to reduce. The shot count with different heuristics, along with the lower/upper bounds are also shown in Table 4.2, if the σ of the Gaussian proximity model is reduced to $4nm$ instead of $6.25nm$. The impact of change in σ on the shot count can vary for different shapes. For most shapes, the change in shot count of GSC heuristic, and the lower upper bounds of B&P is not very large. However, MP heuristic is very sensitive to the value of σ and the shot count increases for most mask shapes shown here.

In addition to σ, another important factor that can affect the shot count is CD tolerance γ. Tighter CD tolerance will increase the number of constraints that a fracturing solution must follow, leading to higher shot count. More constraints also slows down B&P and increases the gap between the reported upper and lower bounds. This is illustrated in Table 4.2, if we compare the shot count of the scenario

with $\sigma = 4nm, \gamma = 2nm$ to the one with $\sigma = 4nm, \gamma = 1nm$.

4.5 Generation of Benchmarks with Known Optimum

The results of Section 4.4.6 show that ILP-based benchmarking requires considerable computational resources just to find lower and upper bounds on the optimal shot count. In this section, we propose a more scalable method to evaluate the suboptimality of mask fracturing heuristics by constructing target shapes for which the minimum shot count is known. This benchmark generation method is based on the key observation that there is a set of boundary segments each of which requires at least two shots in *any* fracturing solution. Here we use B^n to denote the set of all boundary segments such that each boundary segment $b^n \in B^n$ requires at least n shots. For example, the union of green and red lines in Figure 4.10 is a boundary segment b^2.

In our benchmark generation method, we first use exactly two shots to generate a target shape which contains a boundary segment b^2. By the definition of b^2, we need at least two shots to produce the boundary segment. Since we use exactly two shots to generate the target shape, our solution is optimal and the minimum shot count is two. To extend the target shape, we then add a new shot adjacent to one of the existing shots. We select the location of the new shot such that there is a new b^2 in the extended target shape. Note that we only increase the total shot count by one (and reuse an existing shot) to produce the new b^2 which requires two shots. Because the extended target shape cannot be produced by stretching or shifting the shots in the previous solutions (i.e., at least one more shot is required), the solution corresponding to the extended target shape remains optimal with respect to shot count.

In the remainder of this section, we describe the details of our benchmark generation method and prove that a generated target shape has known minimum shot count.

4.5.1 Boundary Segment Analysis

To determine the set of boundary segments which require at least two shots, we analyze the relationship between straight/concave boundary segments and the image produced by a shot. We do not analyze the case of convex boundary segments because it is not used in our benchmark generation.

Straight boundary segment. Since a mask shot must be isothetic, a single mask shot cannot produce a long straight boundary at an angle (θ) which is not a multiple of 90°.[8] Figure 4.8 shows a straight boundary segment b_{seg} (black solid line) at an angle θ. The dashed lines parallel to b_{seg} are the inner and outer boundaries. The inner (resp. outer) boundary is obtained by shrinking (resp. expanding) the target boundary towards the inside (resp. outside) of the target shape by the value of γ. To produce the straight boundary using a single shot, we must place a corner of the shot close to b_{seg}. The longest straight boundary covered by the single shot is the length ($L^{\theta}_{lin}(W, H)$) between the crossing points (blue cross marks in Figure 4.8) of the inner target boundary and the image boundary.[9] To maximize the coverage of a single shot, we must shift the shot and therefore the image boundary to touch the outer boundary as shown in Figure 4.8. The shot must not be shifted beyond the outer boundary because $I(x, y, s)$ must be less than ρ for all pixels in P_0.

Concave boundary segment. Figure 4.9(a) shows a concave boundary b_{seg} and its inner (b_{seg_in}) and outer (b_{seg_out}) boundaries. For a concave target boundary, the maximum boundary length covered by a single shot is defined by the straight line between the points of intersection between b_{seg_in} and the shot image boundary (i.e., the blue cross marks in Figures 4.9(a) and 4.9(b)). From the straight line between the points of intersection, we define a "virtual" straight line (b_{vir}) and its inner (b_{vir_in}) and outer (b_{vir_out}) boundaries. Note that because of the concavity of b_{seg}, any point along b_{vir_in} is always closer than b_{seg_in} to the point that touches the target boundary (i.e., p_c in Figure 4.9(a)). Thus, b_{vir_out} is always in P_0, outside

[8] ILT masks can have non-orthogonal target shapes, especially if methods such as *level set* are used for ILT [**?**].

[9] W and H correspond to the width $W(s)$ and height $H(s)$ of the shot s under consideration.

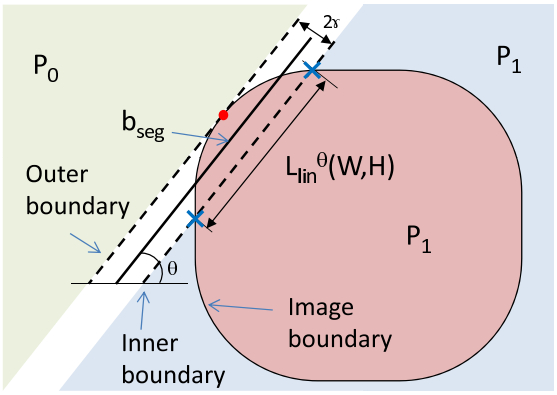


Figure 4.8: Definition of the length $L^{\theta}_{lin}(W, H)$ of a straight-line target boundary covered by a single shot.

the boundary of the shot image. This means that the shot and its corresponding image can be shifted until the shot image boundary touches b_{vir_out} as shown in Figure 4.9(b). As a result, *the length of the virtual straight line, which is the same as* $L^{\theta}_{lin}(W, H)$ *at the same* θ, is always larger than the length $L^{\theta}_{con}(W, H)$ of the concave target boundary.

Maximum length covered by a shot. As mentioned above, the rounded corner of a single shot image determines the maximum length covered by the shot. As the shot size increases, the corner rounding due to the e-beam proximity effect saturates. As a result, the $L^{\theta}_{lin}(W, H)$ does not change further with respect to the shot size. For example, Therefore, we can calculate the L^{θ}_{max} by increasing W and H iteratively, and stopping when L^{θ}_{max} does not increase.

$$L^{\theta}_{max} = \max_{s \in S}\{L^{\theta}(W(s), H(s))\} \tag{4.6}$$

Since $L^{\theta}_{con}(W, H) < L^{\theta}_{lin}(W, H)$ for any shot s, the maximum value of $L^{\theta}_{lin}(W, H)$ is an upper bound on $\max_{s \in S}\{L^{\theta}_{con}(W(s), H(s))\}$.

Lemma 4.5.1. *For a mask fracturing problem with finite* γ *and* σ*, if a target boundary segment is a straight line or concave shape with length* L_t *(defined in Figure 4.10) larger than* L^{θ}_{max}*, more than one mask shot is required to pattern the target boundary segment.*

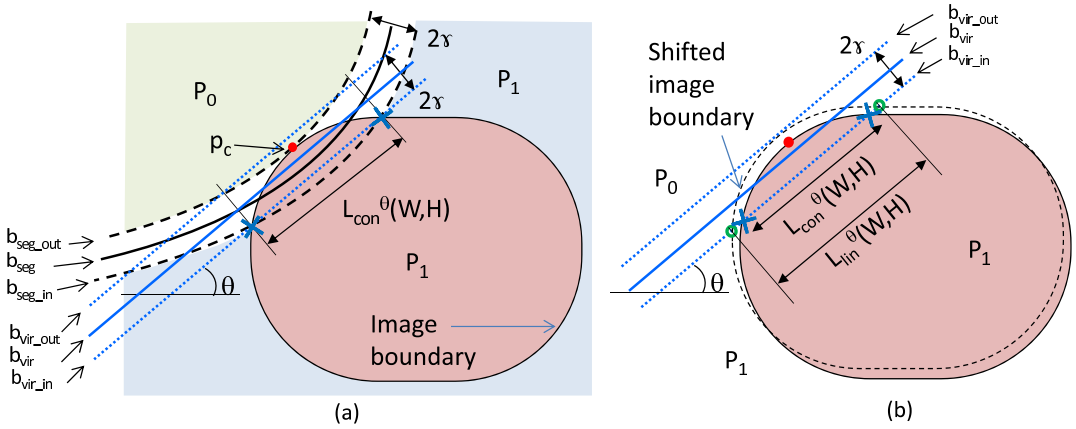


Figure 4.9: (a) Definition of the length $L^{\theta}_{con}(W, H)$ of a concave target boundary covered by a single shot. (b) Comparison of the lengths covered by a single shot for concave vs. straight-line target boundaries.

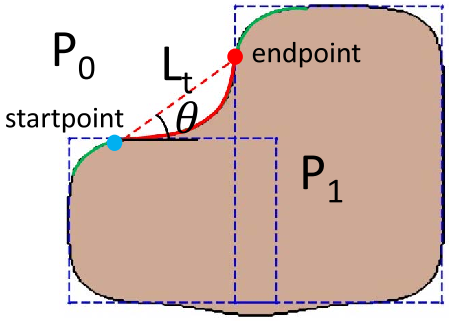


Figure 4.10: L_t is the Euclidean distance between the startpoint and the endpoint on the target boundary, provided that the target boundary from the startpoint to the endpoint is concave or a straight line.

Proof. As mentioned above, the corner of the image produced by a shot does not change beyond a certain shot size, and there exists an L^{θ}_{max} for straight boundary which is also the upper bound for concave target boundary. By definition, L^{θ}_{max} is the maximum length on the target boundary which can be covered by a single shot. Therefore, when a target boundary segment has length $L_t > L^{\theta}_{max}$, we require more than one shot to produce the boundary segment. Note that to check whether a boundary segment has length $> L^{\theta}_{max}$, it suffices to calculate the L_t for all combinations of startpoint and endpoint along the boundary. □

4.5.2 Construction of a Target Shape

We now describe a systematic method to construct a target shape with known minimum shot count. We first construct a b_{seg} using two shots by placing the second shot to the top right of the first shot as shown in Figure 4.11. We define the top left boundary (e.g., the union of green and red lines in Figure 4.11) as the *main boundary* (b_{main}).[10] By placing the second shot far enough from the first shot, we create a *critical boundary segment* $b_{cri} \in B^2$, which is part of the b_{main}. The b_{cri} is a straight line or a concave segment with length L^θ larger than L^θ_{max} (Lemma 4.5.1). Note that although there can be many boundary segments $\in B^2$, only those overlapping with b_{main} are considered as the critical boundary segments. For example, the yellow boundary segment in Figure 4.11, while an element of B^2, is not considered to be a b_{cri} because it does not overlap with b_{main}.

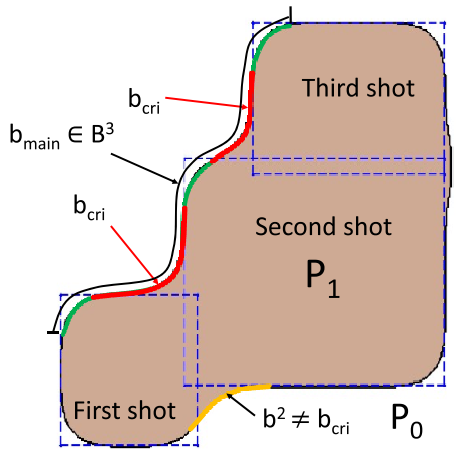


Figure 4.11: Example of benchmark generation with three shots. b_{main} is the union of green and red lines and contains two b_{cri}.

Lemma 4.5.2. *Given a boundary segment b^n of a target with $n-1$ critical boundary segments, and its corresponding shots, we can add a shot to obtain b^{n+1} with an optimal $(n+1)$-shot solution if the addition satisfies the following conditions:*

(i) Adding a shot does not affect the critical boundary segments of b^n.

[10] b_{main} is at the top left boundary because we place the next shot to the top right of previous shots.

(ii) b_{main} *of the new target shape is continuous.*

(iii) There is a b^2 *in the* b_{main} *of the new target shape which cannot be made by extending the shots which produce* b^n *without altering the critical boundary segments of* b^n*.*

Proof. Since there are $n-1$ critical boundary segments in b^n and the newly added shot does not affect the critical boundary segments in b^n, the new target shape still requires n shots for the $n-1$ critical boundary segments. To create a b^2 in the new boundary which cannot be made by extending shots which produce b^n, we need exactly one more shot. Thus, the new target shape has a boundary segment b^{n+1}. □

Based on Lemma 4.5.2, we then add a shot at the top right of the existing target shape. This ensures that we have a continuous b_{main}. Moreover, the top-left coordinate of the newly added shot is selected such that there is a b^2 in the new b_{main}. Since the new b^2 is always at the top of the target shape, it cannot be made by extending previous shots unless the existing critical boundary segments are altered. Also, placing the shot at the top right does not affect the existing critical boundary segments. By adding $n-2$ shots to the target shape generated by two shots, we can obtain a target shape $\in B^n$ based on Lemma 4.5.2.

An important property of our method is that the critical boundary segments are defined only by the top-left coordinates of the shots. Therefore, we may freely place the bottom-right coordinates of the shots to create different target shapes as long as they do not affect the critical boundary segments.

4.5.3 Merging Target Shapes

Lemma 4.5.3. *Given two target shapes with critical boundary segments* $b_a \in B^{n_a}$ *and* $b_b \in B^{n_b}$*, which have, respectively,* $n_a - 1$ *and* $n_b - 1$ *critical boundary segments, we can merge* b_a *and* b_b *by stretching a shot to create* $b_c \in B^{n_a+n_b-1}$ *if the following conditions are satisfied:*

[(i)]

(i) The stretched shot must not alter the critical boundary segments in b_a or b_b.

(ii) The stretched shot must merge a shot from b_a with a shot from b_b.

(iii) The non-stretched shots in b_a must be far apart from or misaligned from the non-stretched shots in b_b so that any two non-stretched shots cannot be merged to reduce the number of shots.

Proof. Since stretching the shot does not alter the critical boundary segments in b_a and b_b, we need at least n_a shots for target shape b_a and n_b shots for the target shapes b_b. Since the merged b_c contains both b_a and b_b, which share one shot, b_c requires $n_a + n_b - 1$ shots. Therefore, b_c belongs to $B^{n_a+n_b-1}$. □

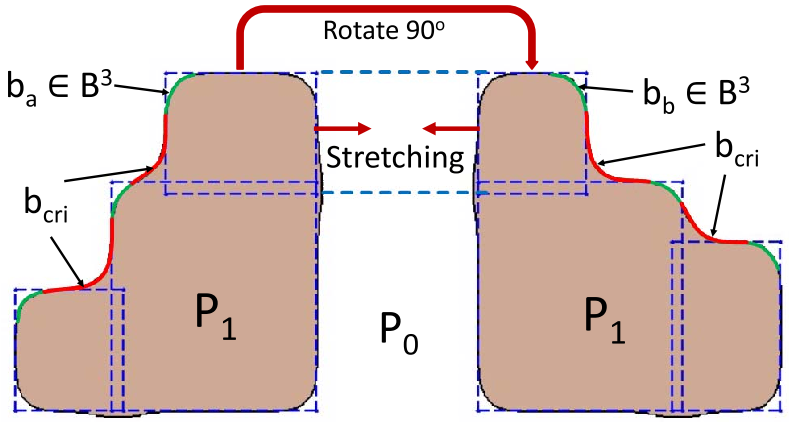
The first condition in Lemma 4.5.3 imposes a tight constraint on merging the target shapes generated by the method described in Section 4.5.2. This is because we can only stretch a shot by moving the lower right corner of the shot in either the rightward and/or downward direction, such that the critical boundary segments are not affected. However, stretching a shot of a target shape to the right and/or down directions will affect the critical boundary segments on the other target shape. This problem can be solved by rotating the target shapes before merging them.

Lemma 4.5.4. *A b^n rotated by $90°$ is still an element of B^n.*

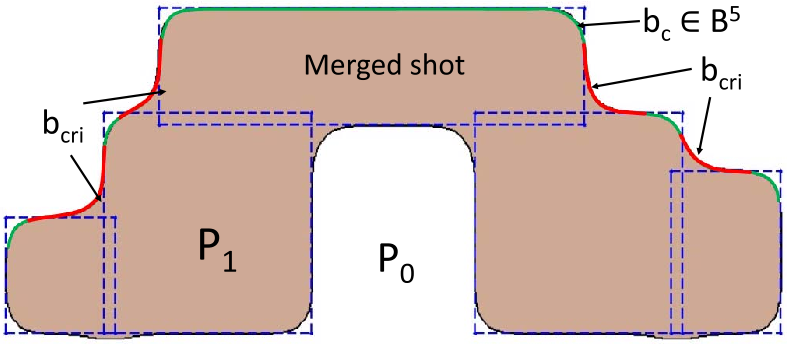
Proof. After applying Gaussian blur, the intensity of a shot is symmetric about the x- and y-axes. Therefore, rotating a target shape by a multiple of 90^o does not affect the number of shots. As a result, if any boundary segment b is in B^n, the boundary segment $b' = b$ rotated by 90^o (or any multiple of $90°$) is also in B^n. □

Figure 4.12 shows an example in which we use Lemmas 4.5.3 and 4.5.4 to merge a target shape and its rotated copy into a larger and more complex target shape.

Using the incremental target boundary extension (Lemma 4.5.2) and merging/rotation of optimal target shapes, we can generate a variety of different benchmarks with arbitrary values of optimal shot count.



(a)



(b)

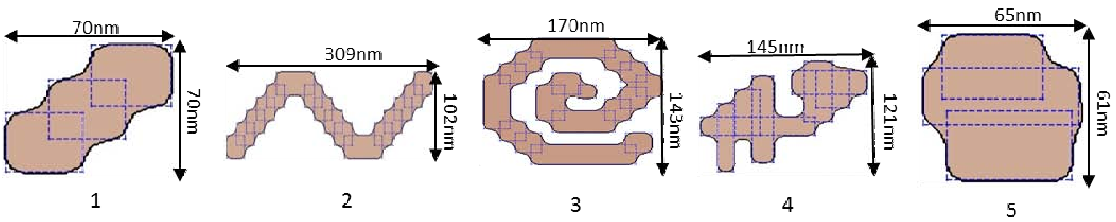
Figure 4.12: Example of rotating a target shape for merging.

4.5.4 Experimental Results

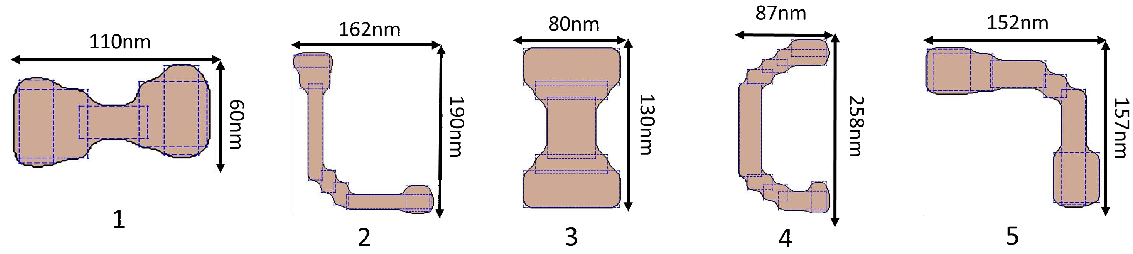
Using the same experimental setup as described in Section 4.4.6, with $\sigma = 6.25nm, \gamma = 2nm$, we generate the following two types of target shapes:

(i) **Arbitrary generated benchmarks (AGB):** We generate five shapes with known optimal shot count using the method described in Section 4.5.2. These benchmarks are shown in Figure 4.13a.

(ii) **Realistic generated benchmarks (RGB):** Since generated benchmarks can often be unrealistic compared to actual ILT mask shapes, we also generate five mask shapes that look similar to actual ILT shapes with known optimal shot count, again using the method described in Section 4.5.2. We manually select shot locations so that the generated benchmarks are similar to actual ILT mask



(a) Arbitrary generated benchmarks



(b) Realistic generated benchmarks

Figure 4.13: Illustration of generated benchmarks with the optimal mask fracturing solution shown in dashed lines (wafer scale).

shapes.

These benchmarks are illustrated in Figure 4.13b.

We compare the optimal shot count of our generated benchmarks with the shot counts of the comparison heuristics in Table 4.3. For the ten target shapes that we analyze, the suboptimality ranges from 2.4× to 5×, 1.3× to 5.6× and 1.6× to 2.9× for the GSC, MP and PROTO-EDA heuristics, respectively.

For comparison, we also report the lower and upper bounds obtained from B&P for our generated benchmarks in Table 4.3. The results show that for testcases AGB-{1,5} and RGB-{3,5}, the B&P method can find the optimal solution, i.e. the upper bound is equal to the optimal shot count. However, for some shapes, the upper bound reported by B&P within the set time limit may be very far from the optimal shot count (testcases AGB-2, AGB-4 and RGB-4).

Note that the generated benchmarks are more wavy (i.e., have high-frequency components in the boundary of the target shape) compared to actual ILT shapes. This could make the suboptimality estimation more pessimistic. However, highlighting scenarios where mask fracturing heuristics perform poorly is important for

Table 4.3: Comparison of shot count for generated benchmarks with known optimal solution.

Clip-ID		Shot count				
		Opt	GSC	MP	PROTO-EDA	Branch and Price
						LB/UB
AGB	1	3	8	4	7	3/3
	2	16	64	26	30	10/47
	3	17	52	35	40	11/19
	4	7	26	9	20	5/18
	5	3	13	6	8	3/3
RGB	1	5	12	19	8	3/6
	2	7	15	31	14	5/8
	3	5	13	28	12	4/5
	4	9	45	19	17	6/14
	5	6	21	16	14	4/6

developing better heuristics.

4.6 Automated Benchmark Generation

In Section 4.5, we constructed benchmark shapes by placing shots manually. This can be extremely tedious, especially for generating benchmarks similar to real ILT shapes. In this section, we propose an automated benchmark generation (AutoBG) method to generate a benchmark ILT mask shape (t_{gen}), which resembles a given actual shape (t_{ori}), with known optimal shot count. To guarantee that the optimal fracturing solution of t_{gen} is known, AutoBG places shots such that they obey the constraints specified in Section 4.5.

In AutoBG, we first split t_{ori} using the method described in Section 4.4.3.[11] For each split-shape ($t_{ori,i}$), we generate a separate benchmark shape ($t_{gen,i}$) with known minimum shot count that resembles $t_{ori,i}$. Then we apply Lemma 4.5.3 to obtain the benchmark shape t_{gen}, which resembles t_{ori}.

[11]We set the threshold (L_{th}) to split t_{ori} as 2β, then to get better similarity, we push the L_{th} lower for certain shapes in AutoBG but we check the intensity map to guarantee optimality

To obtain $t_{gen,i}$ from $t_{ori,i}$, we first generate several candidate sets of linear segment such that each set approximates part of the boundary of $t_{ori,i}$. The candidate set of linear segments that is eventually picked would become the main boundary segment b_{main} of $t_{gen,i}$. Next, we determine the locations of corner points of shots to construct b_{main}. Lastly, for each corner point of a shot used to generate b_{main}, we find the diagonally opposite corner point to minimize the XOR difference between the input and generated shape ($d(t_{ori,i}, t_{gen,i})$). In the remainder of this section, we describe the details of these steps.

4.6.1 Finding Candidate Main Boundary Segments

Based on the Lemma 4.5.2, b_{main} is a continuous boundary segment of the generated mask shape which determines the minimum number of shots required to construct $t_{gen,i}$. Moreover, all the shot corner points used to generate b_{main} must be the same type (i.e., bottom-left, bottom-right, top-left, top-right). Given the input split-shape $t_{ori,i}$, to construct the benchmark shape with known optimal shot count $t_{gen,i}$, we need to find a boundary segment that it can be used as the main boundary segment to place the optimal shot corners. To simplify this, we use a set of connected straight line segments, which approximates part of the boundary of $t_{ori,i}$. We enumerate several such candidate main boundary segments, construct the optimal benchmark shape for each such main boundary segment, and then pick the generated shape that is the most similar to the input shape $t_{ori,i}$ as $t_{gen,i}$. Note that the similarity between any two shapes is measured by the area of the region obtained after polygon Boolean XOR operation between the two given shapes.

For finding a candidate main boundary segment, we shall select an ordered subset of the vertices of $t_{ori,i}$ (V_{main}) such that the line segments obtained after connecting the vertices approximates some boundary segment of $t_{ori,i}$ within the CD tolerance γ. We first define a cost function *slopeCDcost* for selecting the two vertices $v_k(t)$ and $v_l(t)$ as a part of V_{main} in Algorithm 6. The cost is equal to the area of the error pixels which are outside the CD tolerance region of the line segment, as illustrated in

Figure 4.14 (Line 3). To ensure that the set of line segments obtained from V_{main} can be constructed using only one type of shot corner points, we need to guarantee that all the chosen linear segments have angle with x-axis in the same quadrant. We add an additional constraint to the cost function that assigns the cost value to infinity if the angle of the linear segment with the x-axis is outside the specified lower and upper bound (θ_{LB} and θ_{UB}). The values of θ_{LB}/θ_{UB} could be 0/90, 90/180, 180/270 or 270/360 corresponding to upper-left, upper-right, lower-right and lower-left shot corners respectively.

Algorithm 6 Cost function for selecting a pair of vertices as part of b'_{main}

Procedure: *slopeCDcost*(*Target shape t, two vertices* $v_k(t)$ and $v_l(t)$, θ_{LB}, θ_{UB})

Output: *Cost of the segment formed by vertices* $v_k(t)$ *and* $v_l(t)$

1: $\theta_{(v_k(t),v_l(t))} \leftarrow$ the angle of the segment formed by vertices $v_k(t)$ and $v_l(t)$
2: **if** $(\theta_{(v_k(t),v_l(t))} \leq \theta_{UB})$ && $(\theta_{(v_k(t),v_l(t))} \geq \theta_{LB})$ **then**
3: *cost* $\leftarrow$ CD violating area of line segment between $v_k(t)$ and $v_l(t)$ (Figure 4.14)
4: **else**
5: *cost* $\leftarrow +\infty$
6: **end if**
7: return cost;

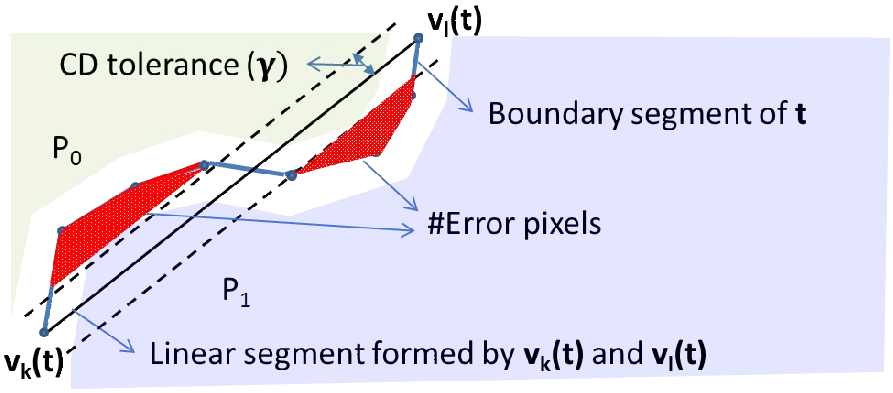


Figure 4.14: Number of error pixels along the segment $v_k(t)$ - $v_l(t)$.

Inspired by Sato's dynamic programming method to approximate any given curve by a set of linear segments, we propose a similar technique to find a candidate main boundary segment as shown in Algorithm 7. For each vertex of the input shape, we iterate over all the previous vertices in the list and find the cost (*slopeCDcost* of

Algorithm 6) for using the linear segment connecting the two vertices as a part of the main boundary segment (Lines 6-8). For each vertex, we then pick the previous vertex with the minimum cost and store it (Lines 9-10). To find only those line segments which have zero cost, we also store the last vertex which has zero cost (Lines 11-13). Then we backtrack from this last vertex to the first vertex and get all the linear segments with zero cost that approximate part of the boundary of the input shape $t_{ori,i}$ (Lines 15-20) to obtain a candidate V_{main}.

Algorithm 7 Dynamic programming algorithm to get a candidate main boundary segment for target shape t

Input : *Vertices* $V(t)$, *target shape* t, CD tolerance γ, θ_{LB}, θ_{UB})
Output: *Ordered subset of* $V(t)$, V_{main}
1: $k = 1$
2: $last\_index = 1$
3: $min\_cost(k) = 0$
4: **for** $k = 1$ to $|V(t)|$ **do**
5: **for** $l = 1$ to k **do**
6: $cost(l) = slopeCDcost(v_l(t), v_k(t), t, \theta_{LB}, \theta_{UB})$;
7: **end for**
8: $sol\_index(k)$ = the index l which has the minimum cost
9: $min\_cost(k) = cost(sol\_index(k))$
10:
11: **if** $(min\_cost(k) == 0)$ **then**
12: $last\_index = k$
13: **end if**
14: **end for**
15: $j = last\_index$
16: Insert $v_j(t)$ to V_{main}
17: **while** $j \neq 1$ **do**
18: Insert $v_{sol(j)}(t)$ to V_{main}
19: $j = sol(j)$
20: **end while**
21: return V_{main}

The list of vertices of any polygon, $V(t)$, is cyclical, i.e. any vertex can be used as the starting point. Moreover, the vertices can be ordered in clockwise or anti-clockwise direction. However, the choice of the starting vertex and the direction affect the candidate boundary segment that we obtain from Algorithm 7. A poor choice of the starting vertex or direction for getting V_{main} could construct $t_{gen,i}$

which does not look similar to $t_{ori,i}$. To avoid this limitation, we pick different starting vertices. Then for each starting vertex, we consider both the clockwise and anti-clockwise direction to obtain different candidate main boundary segments. We obtain a benchmark shape for each candidate segment and then pick the one which looks most similar to $t_{ori,i}$. We select the starting vertices for generating candidate V_{main} using the following criteria:

- If the target shape t_{ori} has no split location, we select the four vertices that are the top, bottom, left and right vertices of t_{ori} as the starting vertices.
- When the target shape is a split-shape, we can find the starting point from the split-line. Split-line is the horizontal (vertical) line splits t_{ori}. We define the shot which is split by the split-line as split-shot. Since we want to maximize the coverage of split-shot, we stretch the shot as much as possible from the split-line. When the split-shot is across the boundary of $t_{ori,i}$, we set the crossing vertices as the starting vertices.

4.6.2 Determine Corner Points

For each candidate main boundary segment that we get from Algorithm 7, we need to place shot corner points to construct the boundary segment. The set of shot corner points that construct the straight line segments obtained from V_{main}, C_{main}, must be placed such that Lemma 4.5.1 and Lemma 4.5.2 are obeyed so that the fracturing solution of the generated benchmark split-shape $t_{gen,i}$ is actually optimal. Algorithm 8 outlines the steps we use to find C_{main}.

In Algorithm 8, we first order the points in V_{main} such that they are sorted by x-coordinate of the points in Line 1. Next we shift all the line segments of V_{main} to get V^{sft}_{main}, such that all the shot corner points must lie on the line segments obtained by connecting consecutive points of V^{sft}_{main} (Line 2). This shift compensates for the difference between the rectangular mask shot and its rounded corner due to the e-beam proximity effect, as illustrated in Figure 4.15.

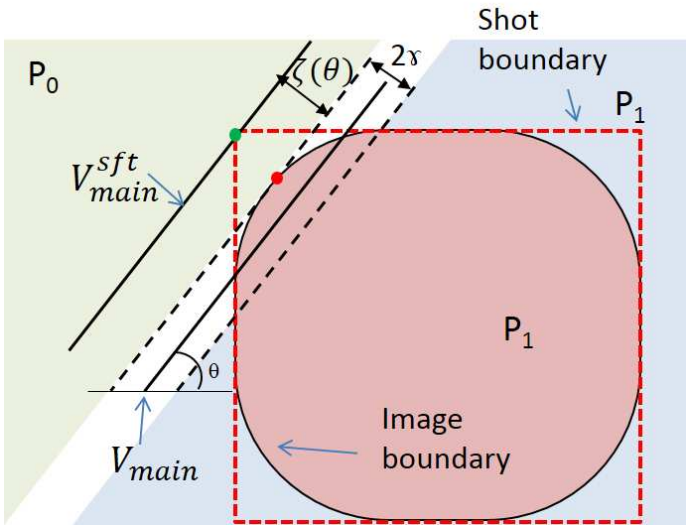


Figure 4.15: Illustration of the gap ($\zeta(\theta)$) between a shot corner point and the line segment with slope θ that is part of boundary segment.

Once we obtain the shifted set of points V^{sft}_{main} such that all shot corner points lie on the line segments connecting consecutive points from V^{sft}_{main}, we can then find a set of shot corner points using the function $getCnrPtsFrmSrt$ in Algorithm 8. We first sample the line segments connecting consecutive points from V^{sft}_{main} and get all points with integral coordinates that lie on these line segments, V_{samp} (Line 1). Next, we include the first point of V_{samp} as a shot corner point in Line 2. We then iterate over the set of sampled points and if the distance between the previously added shot corner point c_{prev} and the sampled point $v_{samp,i}$ is greater than L^{θ}_{th}, we add the sampled point to the set of shot corner points (Lines 4-9). Note that θ is the angle of the line segment connecting c_{prev} and $v_{samp,i}$ with the x-axis. This distance criteria ensures the optimality of the fracturing solution of $t_{gen,i}$ by adhering to Lemma 4.5.1 and Lemma 4.5.2.

If we obtain shot corner points from the shifted set of points V^{sft}_{main} using the method described in $getCnrPtsFrmSrt()$ of Algorithm 8, it could lead to large error between the input mask shape and generated shape near the location of the last point of V^{sft}_{main}. This is illustrated in Figure 4.16(a) that shows V^{sft}_{main}, V_{main} and $t_{ori,i}$. If the right-most point of V^{sft}_{main} is the first point, then the shot corner

points we get from $getCnrPtsFrmSrt()$ are $c^l_{main,1}$, $c^l_{main,2}$ and $c^l_{main,3}$. Due to the minimum distance constraint between shot corner points imposed by Lemma 4.5.1 and Lemma 4.5.2, no additional shot corner points can be chosen after $c^l_{main,3}$. As a result, there is significant pixel error after the last shot corner point $c^l_{main,3}$.

To reduce the pixel error after the last shot corner point, we first get two sets of potential shot corner points: C^l_{main} with the ordered set of points V^{sft}_{main} as input, and C^r_{main} with reverse ordered set of the same points $V^{sft,rvr}_{main}$, as input to $getCnrPtsFrmSrt()$ (Lines 2-4). Then we reverse the order of C^r_{main} and check if the first corner point of C^l_{main} and C^r_{main} are close to each other ($\leq \gamma$) in Line 5. If this is true, then all the points of C^l_{main} and C^r_{main} will be close to each other, and hence we can just use C^l_{main} as the set of shot corner points C_{main} that can construct the line segments formed by V_{main} (Line 7). However, if the potential shot corner points of C^l_{main} and C^r_{main} are not close to each other, we take the average of the x- and y- coordinate of the corresponding points in C^l_{main} and C^r_{main}. We also include the lowest x-coordinate point, i.e. the first point of C^l_{main}, and the highest x-coordinate point, i.e. the last point of C^r_{main}. Figure 4.16(b) shows the result with this choice of corner points. Although this creates error pixels all along the main boundary, it does not cause any large pixel error after the last corner point. Moreover, choosing the lowest and highest x-coordinate shot corner point in C_{main} makes it easier to merge the $t_{gen,i}$ shapes to obtain t_{gen}.

4.6.3 Determine opposite corner points

We now describe the method to determine the locations of diagonally opposite corner points (C^{opp}_{main}) of C_{main}. Since a shot is determined by C^{opp}_{main} and C_{main}, C^{opp}_{main} must be placed such that the shot do not affect b'_{main}, the shot size constraints are obeyed, and the generated shape $t_{gen,i}$ is similar to $t_{ori,i}$.

Algorithm 9 summarizes our method for finding the opposite shot corner points. Given a fixed corner point $c_{main} \in C_{main}$, we first enumerate all points which could become the opposite corner point $c^{opp}_{main} \in C^{opp}_{main}$ in Line 3. If c_{main} is a top-left shot

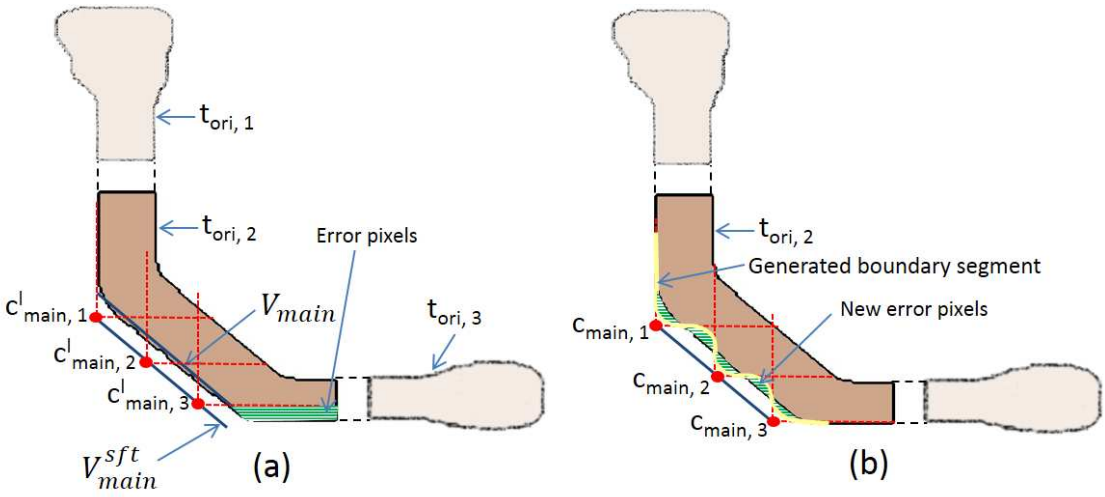


Figure 4.16: (a) Example of C^{l}_{main}. (b) Example of C_{main}.

corner, we can find candidate opposite points by considering all the points within distance γ of the boundary of $t_{ori,i}$ that also satisfy the following two conditions:

(i) lie below and to the right of c_{main}, and

(ii) distance from c_{main} is such that the corresponding shot will satisfy shot size constraints.candidates for c^{opp}_{main}.

Candidate opposite points for c_{main} if it is a bottom-left, top-left or top-right shot corner can be obtained using a similar criteria.

After finding the candidate set of opposite corner points in Algorithm 9, we iterate over this set and find the opposite point for which the corresponding shot best covers the input shape $t_{ori,i}$ in Lines 4-10. This opposite point is then inserted to the list of opposite shot corner points in Line 11. Once the set of shot corner points that construct b'_{main} (C_{main}) are known along with the corresponding opposite shot corner points (C^{opp}_{main}), we can obtain the shape $t_{gen,i}$ by adding the intensity of all the corresponding shots and applying the resist threshold.

4.6.4 Experimental Results

We first generate shapes by using the manually generated benchmark shapes from Figure 4.13 as input to our implementation of AutoBG in C++. Table 4.6.4 shows

the shot count, runtime and the similarity (area of XOR of input shape and generated shape divided by the area of the input shape). The shapes generated by AutoBG are also shown in Figure 4.17.

Table 4.4: Validation of AutoBG method.

Clip-ID		Manual	AutoBG		
		Shot count	**Shot count**	**Similarity (%)**	**Runtime (s)**
AGB	1	3	3	87	<1
	2	16	10	87	4
	3	17	16	85	8
	4	7	8	70	2
	5	3	3	87	2
RGB	1	5	3	82	2
	2	7	7	85	4
	3	5	3	90	4
	4	9	8	79	4
	5	6	6	85	4

From Table 4.6.4, it is clear that AutoBG can generate shapes that are more than 80% similar to input mask shapes for most cases. The similarity is somewhat less for a few complex shapes such as AGB-4. In addition to similarity, the optimal shot count of the input shapes and the optimal shot count of the AutoBG generated shapes are fairly close.[12] In fact, they are the same for four cases. This suggests that the optimal shot count of the AutoBG generated shapes for real ILT mask shapes will be close to the unknown optimal shot count of the ILT shapes. Lastly, the runtime to generate benchmark shapes is less than eight seconds.

The Tables 4.5 and 4.6 show the shot count and similarity of AGB1 according to different γ and σ. As increasing γ and σ, shot count decreases from five to two shots and the similarity becomes worse from 92% to 73%. This is because the larger γ and σ increase the L_{th}^{θ} and the increased L_{th}^{θ} results in reducing the shot and worsening the similarity. For the AGB3 shape, the same trend is shown in Tables 4.7 and 4.8.

[12]Note that this difference in optimal shot count is between the input shape and AutoBG generated shape. The shot count of the AutoBG generated shapes is still optimal since the generation process obeys the constraints of Section 4.5.

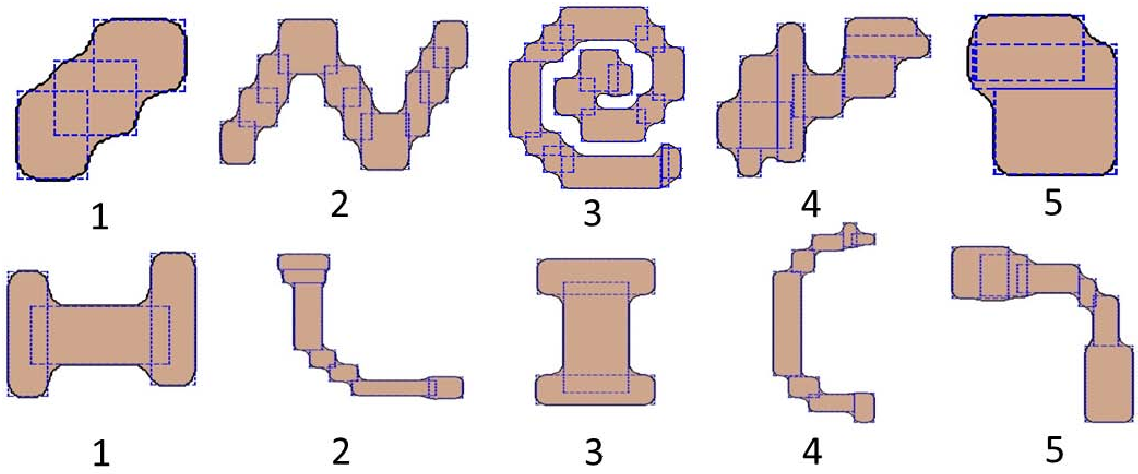


Figure 4.17: Illustration of generated benchmark shapes obtained from AutoBG with shapes in Figure 4.13 as inputs.

		CD tolerance (nm)		
		1	2	3
Proximity model (nm)	3.25	5	5	4
	6.25	4	3	3
	9.25	3	3	2

Table 4.5: #Shots of AGB1

		CD tolerance (nm)		
		1	2	3
Proximity model (nm)	3.25	92	90	82
	6.25	86	88	84
	9.25	78	73	73

Table 4.6: Similarity of AGB1

		CD tolerance (nm)		
		1	2	3
Proximity model (nm)	3.25	28	21	19
	6.25	17	17	14
	9.25	13	12	13

Table 4.7: #Shots of AGB3

		CD tolerance (nm)		
		1	2	3
Proximity model (nm)	3.25	90	90	86
	6.25	87	86	83
	9.25	78	78	78

Table 4.8: Similarity of AGB3

Table 4.9: Comparison of optimal shot count for AutoBG-generated benchmark shapes that are similar to the shapes shown in Figure 4.7 to the shot count of three fracturing heuristics. Benchmark shapes are generated for three scenarios with different values of e-beam proximity model Gaussian variance (σ) and CD tolerance (γ).

Clip-ID												
	$\sigma = 6.25nm$, $\gamma = 2nm$				$\sigma = 4nm$, $\gamma = 2nm$				$\sigma =$			
	Optimal	**GSC**	**MP**	**PROTO-EDA**	**Optimal**	**GSC**	**MP**	**PROTO-EDA**	**Optimal**	**GSC**	**MP**	**P**
1	3	9	5	7	4	7	8	13	6	13	10	
2	7	20	9	16	7	10	13	35	10	32	42	
3	1	1	9	6	1	1	1	8	3	24	24	
4	9	26	20	15	11	21	15	27	13	46	35	
5	6	18	12	11	7	16	11	19	10	38	27	
6	3	4	5	6	3	4	4	13	3	16	10	
7	4	11	4	8	4	7	6	13	6	20	11	
8	7	24	9	9	9	19	15	19	11	33	24	
9	9	29	16	21	10	20	47	29	14	44	59	
10	4	11	7	9	4	9	10	17	7	23	19	

4.6.4.1 Scalability

Next we generate benchmark shapes using all the ten ILT mask shapes shown in Figure 4.7 as inputs with different values of σ and γ. We then compare the sub-optimality of the fracturing heuristics for all these generated benchmark shapes, with the results summarized in Table 4.9. The suboptimality ratio of GSC, MP and PROTO-EDA across the ten shapes for the baseline case ($\sigma = 6.25nm, \gamma = 2nm$) ranges from 1× to 3×, 1× to 9× and 1.2× to 6×, respectively. Interestingly, if σ is reduced to $4nm$, GSC and MP heuristics perform better and the suboptimality ratio reduced. However, the suboptimality range of PROTO-EDA worsens and becomes 2.1× to 8×. On the other hand, if γ is reduced, the performance of PROTO-EDA improves and that of GSC and MP worsens.

4.7 Conclusions

The use of aggressive RET techniques such as ILT, the need for e-beam proximity effect correction, and the use of overlapping shots have transformed mask fracturing into a very challenging computational problem. Although several heuristics have been proposed in the last few years, there has been no systematic study to analyze the quality of solutions. In this work, we propose two methods to evaluate the suboptimality of mask fracturing heuristics. First, we formulate the mask fracturing problem as an integer linear problem and develop a practical branch and price method to generate tight upper and lower bounds on the optimal shot count. Second, we introduce a systematic method to generate a set of benchmarks with known, provably optimal solutions. Third, we describe an automated benchmark generation method to construct shapes which look similar to real ILT shapes.

Using our benchmarking method, we evaluate the suboptimality of three mask fracturing heuristics: greedy set cover, matching pursuit and a state-of-the-art prototype [version of] capability within a commercial EDA tool for e-beam mask shot decomposition (PROTO-EDA). Our experimental results show that PROTO-EDA has up to 6× more shots compared to the optimal solution for generated benchmarks, and has up to 2.3× more shots for ILT mask shapes with unknown optimal solution. These results suggest that there remains considerable opportunity to improve mask fracturing heuristics.

Our future works include (i) to develop better mask fracturing heuristics based on insights from boundary analysis and ILP solutions, and (ii) extend our suboptimality evaluation methodology for the fracturing problem which allows variable-dose or non-rectangular shots.

The latest versions of our source code and benchmark suite are available publicly (http://impact.ee.ucla.edu/maskFracturing/Benchmarks). We hope that this will stimulate further research toward development of improved mask fracturing heuristics.

Algorithm 8 Get shot corner points (C_{main}) to construct a candidate main boundary segment

Require: *Ordered set of points* V_{main}, *CD tolerance* γ, *the distance between corner point of shot and its image* $\zeta(\theta)$, *and Gaussian proximity model* σ

Ensure: *Set of shot corner points* C_{main}

If $v_{main,1}$ has largest x-coordinate in V_{main} reverse order of V_{main}
$V^{sft}_{main} \leftarrow$ Shift every point of V_{main} such that every line segment between consecutive points is shifted by $\zeta(\theta) + \gamma$
$C^{l}_{main} \leftarrow$ getCnrPtsFrmSrt(V^{sft}_{main})
$V^{sft,rvr}_{main} \leftarrow$ Reverse order of V^{sft}_{main}
$C^{r}_{main} \leftarrow$ getCnrPtsFrmSrt($V^{sft,rvr}_{main}$)
Reverse order of C^{r}_{main}
if $(L(c^{l}_{main,1},\ c^{r}_{main,1}) \leq \gamma)$ **then**
    $C_{main} = C^{l}_{main}$
else
    Insert $c^{l}_{main,1}$ to C_{main}
    for $i = 2$ to $|C^{l}_{main}| - 1$ **do**
        $c_{main,i} \leftarrow (c^{l}_{main,i} + c^{r}_{main,i})/2$
        Insert $c_{main,i}$ to C_{main}
    end for
    Insert $c^{r}_{main,|C^{r}_{main}|}$ to C_{main}
end if
return C_{main}

Procedure: *getCnrPtsFrmSrt*(Ordered set of points V^{sft}_{main})

Output: *Set of shot corner points* C

$V_{samp} \leftarrow$ Sample points on all the line segments obtained from V^{sft}_{main}
Insert $v_{samp,1}$ to C
$c_{prev} \leftarrow v_{samp,1}$
for $i = 2$ to $|V_{samp}|$ **do**
    if $(L(v_{samp,i}, c_{prev}) \geq L^{\theta}_{th})$ **then**
        Insert $v_{samp,i}$ to C
        $c_{prev} \leftarrow v_{samp,i}$
    end if
end for
return C

Algorithm 9 Determine opposite corner points for given set of shot corners.

Input : *Shot corner points* C_{main}, *input shape* $t_{ori,i}$

Output: *A set of opposite corner points* C_{main}^{opp}

for all $c_{main} \in C_{main}$ **do**
    $maxCover \leftarrow 0$
    $C_{main}^{can\_opp} \leftarrow$ Candidate opposite shot points for c_{main}
    for all $c \in C_{main}^{can\_opp}$ **do**
        $s \leftarrow$ Shot with opposite corners c_{main} and c
        $cover \leftarrow area(XOR(s, t_{ori,i}))$
        if $cover > maxCover$ **then**
            $c_{main}^{opp} \leftarrow c, maxCover \leftarrow cover$
        end if
    end for
    Add c_{main}^{opp} to C_{main}^{opp}
end for

CHAPTER 5

Defect Avoidance Techniques to Mitigate EUV Mask Defects

5.1 Introduction

As described in Chapter 1, EUV mask blanks suffer from hard-to-repair buried defects. Hence, the ability to tolerate some of these defects without any impact on yield is a very attractive proposition. Defect avoidance based techniques have emerged as a very effective means to tolerate mask defects. These techniques rely on inspection of mask blanks to first determine defect locations. The position of the design pattern, which needs to be written on the mask, can then be shifted relative to the mask to avoid the defects. There are three degrees of freedom that can be exploited to avoid mask blank defects, which is illustrated in Figure 5.1:

- *Pattern Shift* requires moving the entire mask field pattern relative to the defective mask blank to avoid defects. Several prior approaches look at methods to exploit pattern shift to avoid defects [BA10, ZD12, WB12, YL12].
- *Rotation* involves rotating the entire mask pattern about the center of the mask blank. Most approaches consider rotation only in multiples of 90 degrees [BA10, WB12, YL12]. However, Zhang et al. [ZDW12] propose small-angle rotation as well. Although this additional flexibility can improve the chances of using a defective mask blank, it is unclear whether EUV scanners will be able to support non-orthogonal rotation of the mask.
- *Mask Floorplanning* avoids defects by moving each die copy inside the mask field independently. Recently, Du et al. [DZW12b] propose methods to per-

form mask floorplanning together with pattern shift. There are two key issues that could hinder the use of mask floorplanning as a defect avoidance method. First, it can lead to gaps between die copies (scribe area), which is area wasted on the wafer. However, this wasted scribe area is less than 1% of the total area of the die copies according to our experimental results in Section 5.5.3. Second, different layers of the same design must be moved simultaneously. Consequently, mask floorplanning can help improve defect avoidance only if the number of critical design layers patterned using EUV lithography is small.

Elayat et al.[ETS12] and Jeong et al.[JKP11] provide a cost-benefit assesment of different defect avoidance and reticle planning strategies, respectively. The low accuracy of mask blank inspection tools is a serious limitation for defect avoidance based mitigation. Recent techniques have also looked at methods that can tolerate defect position inaccuracy [DZW12a]. Alternate approaches for mask defect mitigation that rely on correcting the absorber pattern after mask write have also been proposed [Cli10, MG13], but they may be less effective than defect avoidance techniques [JH10], especially for large defects or high defect density.

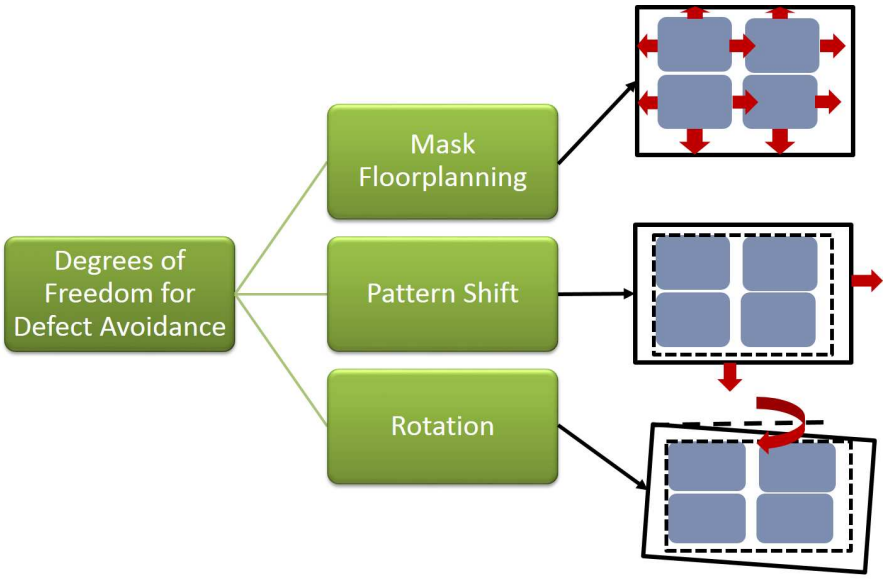


Figure 5.1: Summary of three degrees of freedom for avoiding EUV mask defects.

In this chapter, we present two methods for defect avoidance. First we describe a simulated annealing based method that uses pattern shift and mask floorplanning to avoid mask defects. However, this method is unable to handle small-angle rotation. Moreover, it requires discrete moves which which limits the solution space to avoid defects.

To address the limitations of simulated annealing based defect avoidance, we propose a global optimization based method that allows simultaneous and continuous optimization of all the three degrees of freedom offered by defect avoidance: pattern shift, rotation and floorplanning. We formulate the problem as a non-convex optimization problem and then solve it using a combination of hit-and-run based random walk and gradient descent.

The remainder of this chapter is organized as follows. Section 5.2 describes our method for estimating the wafer CD impact of EUV mask defects. Next, we describe our simulated annealing based defect avoidance method in Section 5.3. Section 5.4 describes our global optimization method. Section 6.5 then shows some simulation results where we compare our methods to prior art and analyze the impact of several different manufacturing scenarios. Finally, we conclude this chapter in Section 5.6. All notation used in this paper is summarized in Table 6.1.

5.2 Modeling CD Impact of Buried Defects

Estimating the impact of buried defects on wafer has been extensively studied through experimental work (wafer exposure followed by inspection) [TYT10] and lithography simulations [CN09] for different defect dimensions and optical conditions. These approaches typically study minimum pitch grating patterns and look at printability and CD change caused by these mask defects for different defect height, width and position relative to the absorber pattern. Using their EUV lithography simulator, Clifford and Neureuther [CN08] propose a simple linear model to estimate the CD change of a grating pattern as a function of defect height for a fixed width and position. Using this model as starting point, with the assumption that it

Table 5.1: Glossary of Terminology

Term	Description
γ	Correction factor of CD impact model of EUV mask defect.
L	Number of design layers patterned using EUV
l	Particular design layer under consideration
BD_l	Set of buried defects in layer l
d	Current defect under consideration
W_d	Full width half maximum of mask defect d
H_d	Height of mask defect d
(X_d, Y_d)	Coordinate of center of mask defect d
Δ	Maximum position inaccurary of EUV mask blank inspection tool
A_l	Set of absorber edges in layer l of design
e	Absorber edge under consideration
$dist(e, d)$	Distance between absorber edge e and the center of defect d
$DefHeight(e, d)$	Height of defect d at location of absorber edge e
$CD_{def}(e, d)$	Change in critical dimension (CD) of absorber edge e caused by defect d
$CD_{tol}(e)$	CD tolerance of absorber edge e
IR_d	Region of influence of buried defect d
$W_D(H_D)$	Width (height) of die
$W_M(H_M)$	Width (height) of usable area of mask blank
$W_F(H_F)$	Width (height) of mask field size
$R(C)$	Number of rows (columns) of die copies in mask field
$r(c)$	Row (column) number under consideration
$Cost_{SA}$	Overall CD impact cost function for defect avoidance using simulated annealing method
$Xp_l(Yp_l)$	X(Y)-coordinate of center of mask field relative to mask blank center for layer l
Θ_l	Angle by which the mask field pattern is rotated relative to the mask blank coordinates
$Xf_r(Yf_c)$	X(Y)-coordinate of $r^{th}(c^{th})$ row of dies
$X(e)$	X-coordinate of a vertical absorber edge e relative to die center
$\hat{X}(e)$	X-coordinate of a vertical absorber edge e relative to mask field center
$Y_{low}(e)$	Bottom y-coordinate of a vertical absorber edge e relative to die center
$\hat{Y}_{low}(e)$	Bottom y-coordinate of a vertical absorber edge e relative to mask field center
$Y_{high}(e)$	Top y-coordinate of a vertical absorber edge e relative to die center
$\hat{Y}_{high}(e)$	Top y-coordinate of a vertical absorber edge e relative to mask field center
$u(.)$	Unit step function. $u(y) = 1$ if $y \geq 0$, $u(y) = 0$ otherwise
$Cost_{GO}$	Overall CD impact cost function for defect avoidance using global optimization method
N_G	Number of iterations of gradient descent for each starting point
S	Step size of gradient descent

is valid even for non-grating layout patterns, we make the following assumptions to evaluate the CD impact of buried defects on a general layout pattern:

- All defects have a 3D symmetric Gaussian shape as shown in Figure 5.2. The application of a smoothing process during the multi-layer deposition [CN08] step for EUV mask manufacturing makes this a fairly accurate assumption for defect modeling.

- The CD impact of a defect on a particular absorber is assumed to be proportional to the height of the defect at the closest edge of the absorber. Hence as a defect moves away from an absorber, its impact reduces exponentially. But, as shown in Figure 5.3, this assumption implies that two defect locations D1 and D2 lead to the same CD impact. In reality, intensity drop of an aerial image, and hence CD impact, would be more when most of the defect is not covered by the absorber. To correct for this, we apply an additional correction factor to our model (γ). We chose $\gamma = 0.5$ if the center of the defect lies under the absorber, and $\gamma = 1.0$ if the defect center lies outside the absorber, based on simulation results in [Cli10].

- To account for defocus, which can have a significant impact on CD change due to the phase nature of these buried defects [TYT10, CN09], we scale up the values obtained from the linear model by 3×. This is based on existing simulation results for defocus value of $\pm 75nm$ [Cli10].

- A single absorber pattern cannot be affected by more than one defect. This assumption is reasonable, considering that typical defects are randomly distributed across an entire $6in. \times 6in.$ mask. Unless defect density is very high, two defects are unlikely to lie close to a single absorber pattern, a situation illustrated in Figure 5.4.

- Current mask blank inspection tools are unable to accurately locate the position of the defect. In order to make the mask floorplanner robust to positional error, we consider a circular region of uncertainty around the most likely de-

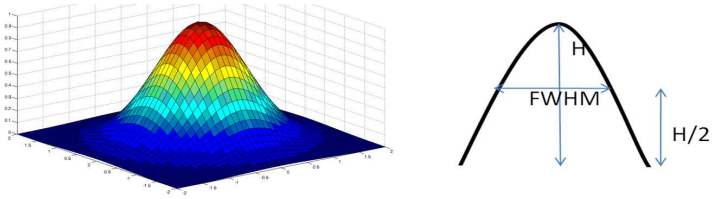


Figure 5.2: A 3D, symmetric Gaussian defect on the left and its planar projection with height H and full width at half maximum FWHM.

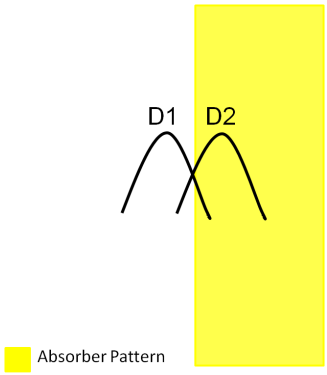


Figure 5.3: Two potential locations of a defect, D1 and D2 relative to absorber edge. We assume that D1 has twice the CD impact of D2.

fect center location (as per the blank inspection tool). We then assume that the distance between the defect and an absorber edge is equal to the smallest distance between the uncertainty region and the absorber. This assumption is illustrated in Figure 5.5.

With these assumptions, the CD impact for the buried defect d, which is at a distance $dist(e,d)$ from an absorber edge e, as shown in Figure 5.6, can be calculated using Equations 5.1 - 5.3. $dist_{\Delta}(e,d)$ is the worst case distance between the defect and the absorber that accounts for the inaccurary of the mask blank inspection tools. $m_{defect} = 0.191nm^{-1}, b_{defect} = 0.094, I_{NoDefect} = 0.3$ and $ImageSlope =$

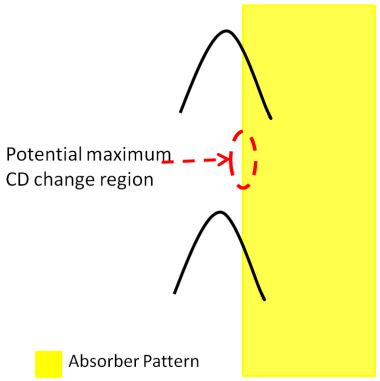


Figure 5.4: A scenario with two defects changing CD of a single absorber. The worst case CD change may not lie at minimum distance edge fragment of either defect.

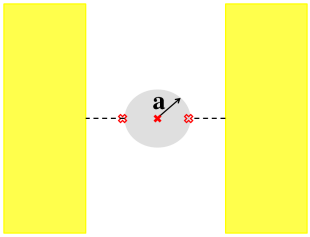


Figure 5.5: Pessimistic approach to model a uncertainty in defect position.

$0.0471nm^{-1}$ are constants whose values are taken from [CN08].

$$dist_{\Delta}(e,d) = max(dist(e,d) - \Delta, 0) \tag{5.1}$$

$$DefHeight(e,d) = H_d e^{-dist^2_{\Delta(e,d)}/(W_d/2)^2} \tag{5.2}$$

$$CD_{def}(e,d) = \frac{3 \cdot \gamma \cdot \sqrt{I_{NoDef}} \cdot (m_{def} \cdot DefHeight(e,d) + b_{def})}{ImageSlope} \tag{5.3}$$

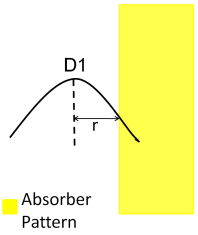


Figure 5.6: A defect and absorber with r as distance between center of defect and closest absorber edge.

5.3 Simulated Annealing Based Defect Avoidance

In this section, we describe a simulated annealing based method for defect avoidance that can exploit two degrees of freedom for defect avoidance, pattern shift and mask floorplanning. We first describe the overall CD impact metric that we minimize during the optimization, after which we describe our algorithm.

5.3.1 Optimization Metric

To find out whether a buried defect will cause a design to fail or not, we also need to know the acceptable CD deviation that each design shape can tolerate. This CD tolerance can be computed using the method we described in Chapter 3 if some design information is available to mask manufacturers. If not, a single conservative CD tolerance can be assigned to each shape in the design. Using a CD tolerance assignment and the CD impact of of each defect on every absorber shape, we develop a concise metric to estimate the overall design impact of buried mask defects, which can then be optimized for by our floorplanner. A design is said to work if $CD_{def}(e,d) < CD_{tol}(e)$ for the all the defects and absorber shapes of each layer of the entire mask pattern. This binary requirement can be treated as a constraint to find a valid floorplan. But a better alternative is to minimize a continuous metric that minimizes the overall CD change of the entire mask so that the impact of defects on the printed patterns is minimized, even if the mask does yield. To do this, we propose a simple cost metric that estimates the design impact of all the buried defects on a mask by aggregating an exponential penalty function across all defects and absorber edges, as shown in Equation 5.4.

$$Cost_{SA} = \sum_{l \in L} \sum_{d \in BD_l} \sum_{e \in A_l} exp^{CD_{def}(e,d) - CD_{tol}(e)} \tag{5.4}$$

$$\tag{5.5}$$

The runtime to compute this metric across all layers is $O(\sum_{l \in L} |BD_l| \cdot |A_l|)$. But instead of computing the cost for each polygon for every defect we can consider only those polygons which lie in a region of influence IR_d from the defect center. This

region is a function of H_d, W_d and defect position error Δ. Finding all polygons which lie within distance IR_d of the defect center can be done in $O(\log |A_l)$ using 2D region query tree data-structure to represent the entire die pattern [Ben75]. Hence the runtime for computing the cost reduces to $O(\sum_{l \in L} |BD_l| \cdot \log |A_l| \cdot |A'_l|)$, where A'_l is the set of polygons inside the region of influence ($|A'_l| << |A_l|$ for typical defect size and alignment error).

Note that this cost metric is not equivalent to yield but it is indicative of the overall electrical impact of buried defects on the design. For example, if a single die has multiple defects, moving the die may not improve yield at all, but it could still reduce this cost metric. Another important point is that although we have used a closed form expression to calculate the CD impact of a buried defect, our floorplanner is agnostic to the defect model. It is possible to use a fast simulator such as RADICAL [CN09] for layout snippets around each buried defect to evaluate the design impact more accurately.

5.3.2 Algorithm

To solve the single project, multiple die reticle floorplanning problem formulated above, we consider only gridded solutions because they guarantee that no die is lost after side-to-side wafer dicing. A non-gridded solution can potentially be more compact, but will usually lose some dies during dicing which need to be accounted for during yield computation. Enforcing a gridded solution also limits the solution space and simplifies the floorplanning algorithm. We chose the simulated annealing framework [KGV83] to solve this optimization problem since previous work on floorplanning [CL03, Kah07] suggests that it is a good heuristic for floorplanning problems.

In simulated annealing based optimization, an initial solution is randomly chosen, which in our case is a floorplan with no space between any die, starting from the center of the usable reticle area. An appropriate perturbation or move is applied to the solution, which increases or decreases the metric we wish to minimize. If a

change or move reduces the cost it is accepted. But the move increases the cost, it is accepted with a finite probability depending on the increase in cost and the number of prior iterations. Temperature is usually used as a parameter that reduces with each iteration of the optimization, in analogy to thermal annealing. So, initially when the system is hot, most moves, even those that increase cost, are accepted. As the system cools down, the optimizer behaves more like a greedy algorithm.

To define moves for gridded solutions, we first define a set of horizontal and vertical gridlines. If we have an initial compact floorplan with R rows and C columns of dies, then we have R horizontal gridlines and C vertical gridlines. Each horizontal(vertical) gridline has its corresponding y(x) coordinate linked to all die whose bottom (left) coordinate is the same. So, each die is linked to two gridlines, one vertical and one horizontal. Both horizontal and vertical gridlines are sorted by their respective coordinates. Each gridline coordinate (and all the linked dies) can be moved by a predefined value $\pm\delta$. This is a move or perturbation for our optimization. Hence any vertical (horizontal) gridline $L_i^V(L_i^H)$ has two possible moves:(1) $x_i(y_i) = x_i(y_i) + \delta$;(2) $x_i(y_i) = x_i(y_i) - \delta$. A move is labeled as valid or invalid based on whether spatial constraints are obeyed after the move is made. The three main types of spatial constraints that must be obeyed by every gridline are listed below, where $W_M(H_M)$ is the usable reticle width (height), $W_D(H_D)$ is die width (height), $c \in \{1, 2...C\}$, $r \in \{1, 2...R\}$ and, $x_1(y_1)$ and $x_n(y_m)$ are the smallest and largest co-ordinates of the gridlines.

- Reticle Boundary Constraints ($c(r) \neq 1$): $x_1(y_1) \geq 0$, $x_C(y_R) + W_D(H_D) \leq W_M M(H_R)$.
- Die Overlap Constraints: $x_c(y_r) - x_{c-1}(y_{r-1}) \geq W_D(H_D)(c(r) \neq 1)$.
- Maximum Allowed Field Size Constraints: $x_C(y_R) + W_D(H_D) - x_1(y_1) \leq W_F(H_F)$

Figure 5.7 graphically illustrates these moves and their validity. There are a total of $2 * R + 2 * C$ potential moves and $4 + (R - 1 + C - 1) + 2$ spatial constraints which

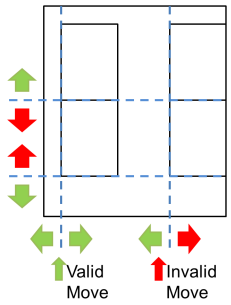


Figure 5.7: Illustration of valid and invalid moves

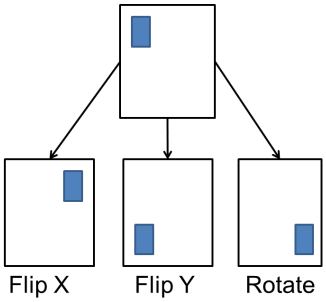


Figure 5.8: Illustration of various orientations for a die

must be checked to determined which of the potential moves are valid.

Apart from moving dies, their orientation can also be changed. Each die can have four possible orientations as shown in Figure 5.8[1]. However, these orientation changes can have significant manufacturing overheads. Flipping the die would lead to dies with different pin locations and hence require a different package. Rotation by 180^o makes wafer testing significantly harder (potentially requiring a different probe-card). Due to these manufacturing overheads, we have disallowed any orientation changes in our algorithm.

Although die level orientation changes are disallowed, rotation of the entire mask pattern (all $R \times C$ dies) will not suffer from any of the manufacturing issues discussed above. In order to allow this orientation change, we apply our simulated annealing based floorplanning described above to four rotated versions (default, 180^o, flipX, flipY) of the entire mask pattern. We then choose the best solution among them.

[1] $90^o/270^o$ rotation is not considered due to lithographic patterning constraints.

We noted earlier that floorplanning incurs an overhead in the form of wasted scribe and consequently, wafer area. It is possible that for a certain defect distribution on the mask, just shifting the entire mask pattern is sufficient. In order to circumvent this limitation of floorplanning, we first perform pattern shift and then check if the mask works. If the mask does not work after pattern shift, we perform floorplanning. Additionally, the minimum CD impact position returned by pattern shifting is used as a starting solution for reticle floorplanning.

Algorithm 10 summarizes the complete algorithm. Lines $1-2$ define an initial partition where dies are placed in a compact grid on the reticle such that the mask pattern is at the center of the usable reticle area. Lines $3-12$ iterate over the four orientation options for the mask pattern and performs floorplanning for each orientation and the best orientation is chosen in Line 13. Lines $4-7$ incorporate the step of shifting the entire mask pattern by calling the function PATTERNSHIFT(). Reticle floorplanning is then performed in Line 11 by calling FLOORPLAN(), if the mask still fails.

The function PATTERNSHIFT() in Lines $14-20$ of Algorithm 10 essentially merges all the dies on the mask to create a single larger pseudo-die, $D_{fullMask}$. With this new die, it calls the existing simulated annealing based FLOORPLAN() function. The final shifted position of the pseudo-die is returned.

The function FLOORPLAN() in Lines $21-41$ of Algorithm 10 is the key function that actually performs the simulated annealing based gridded floorplanning. In each iteration of the while loop the best valid move (maximum cost reduction or minimum increase) is chosen in lines $29-31$. The simulated annealing criteria is then applied to determine if the move should be accepted or not in Lines $32-37$. To improve runtime, we stop the annealing optimization as soon as the mask yields, in Lines $26-28$. This helps reduce the runtime by stopping the optimization when a solution that yields is found.

The runtime of our approach summarized in Algorithm 10 is dominated by the cost computation for each valid move during the FLOORPLAN() function. Among

Algorithm 10 Simulated Annealing Based EUV Mask Defect Avoidance Method

Require: Width (W_M) and Height (H_M) of reticle, width(W_D) and height (H_D) of each die, location/size of defects on mask blank BD_l and all design layout shapes (A_l) with CD tolerances for all design layers $l \in L$

Ensure: Location of die such that number of defects in critical areas is minimized.

1: $R = H_M/H_D$ rows of dies, $C = W_M/W_D$ columns of dies.
2: Place $R \times C$ dies on the reticle such that the reticle field is at the center of the usable reticle area.
3: **for all** orientation $\in$ (default, 180^o, flipX, flipY) **do**
4: **if** Number of die > 1 **then**
5: **for all** $l \in L$ **do**
6: $(Xp_l, Yp_l) \leftarrow$ PATTERNSHIFT(BD_l, A_l)
7: Shift all $d \in BD_l$ by (Xp_l, Yp_l)
8: **end for**
9: **end if**
10: **if** Mask works **then**
11: Exit for loop, choose current solution
12: **end if**
13: $D_{fp}(orientation) \leftarrow$FLOORPLAN($D$, BD, CD_{tol})
14: **end for**
15: $D_{final} = argmin(Cost(D_{fp}))$.

Algorithm 11 Functions used in Algorithm 10

function PATTERNSHIFT(BD, S).
    Merge every $die \in D$ into one large die $D_{fullMask}$.
    $D_{shift} \leftarrow$FLOORPLAN($D_{fullMask}$, BD, S).
    $X = D_{shift} -> left - D_{fullMask} -> left$.
    $Y = D_{shift} -> bottom - D_{fullMask} -> bottom$.
    Return (X, Y).
end function

function FLOORPLAN(D, BD, S)
    Define vertical gridlines for each column of dies in D.
    Define horizontal gridlines for each row of dies in D.
    $T = T_{initial}$, cr is cooling rate .
    while $T > T_{final}$ **do**
        if Mask works **then**
            Exit while loop, choose current solution.
        end if
        Find all valid gridline moves.
        Compute cost change $\Delta Cost$ for each valid move.
        $c^* = min(\Delta Cost)$, $m^* = argmin(\Delta Cost)$
        if $c^* <= 0$ **then**
            Accept m^*.
        end if
        if $c^* > 0$ **then**
            Accept m^* with probability $P = exp(-c^*/T)$.
        end if
        $T = T * cr$.
    end while
    Return D with updated coordinates.
end function

the $2 * (R + C)$ potential moves, we first find the set of valid moves and then evaluate the cost change of each valid move. Although this cost computation is done incrementally in the sense that cost needs to be computed only for the dies which move, at worst it needs to be done for each defect on the mask. For a simulated annealing schedule with initial temperature $T_{initial}$, final temperature T_{final} and cooling rate cr the overall complexity of this approach is therefore, $O(log_{cr}(\frac{T_{final}}{T_{initial}}) \times (k_1 * (R + C) \times f_{cost}))$, where k_1 is a constant and f_{cost} is the time to calculate the cost function of Equation **??** for one die which is $O(\sum_{l \in L} |BD_l| \times \log |A_l|)$.

5.4 Global Optimization Based Defect Avoidance

5.4.1 Problem Formulation for Defect Avoidance

In this work, we focus on EUV mask defect avoidance of single-project masks. This is because EUV is likely to be economically viable only for high volume designs where single-project masks are used. Moreover, we shall assume that the floorplan of the die copies on the mask is gridded. Although this restricts the potential solution space, it guarantees full dicing yield. The number of die copies inside the mask is kept fixed. This contrasts with Du's approach [DZW12b], where the number of die copies on the defective mask is maximized.

Since pattern shift and rotation can be done independently for each layer we define (Xp_l, Yp_l, Θ_l) as the coordinates of the center of the mask field relative to the center of the mask itself and rotation of each layer, l. Mask floorplanning, on the other hand, must be done together for all the layers to ensure layer alignment. Hence we define the relative coordinates of the r^{th} row of dies relative to the zeroth row as Yf_r $(r \in 1, 2...(R-1))$, and the relative coordinate of the c^{th} column of dies relative to the zeroth column is Xf_c $(c \in 1, 2, ...C-1)$. The goal of EUV mask defect avoidance is to determine this set of $3L + R + C - 2$ variables such that the impact of defects is minimized.

In order to ensure that the final mask is manufacturable, certain spatial con-

straints need to be satisfied by any defect avoidance solution. The various types of constraints are the following:

(i) *Reticle boundary constraints* ensure that the entire mask field is inside the usable area of the mask. These spatial constraint must account for rotation, and must be applied for each EUV layer of the design. In order to make these constraints linear, we make the small angle assumption ($\sin\Theta \approx \Theta, \cos\Theta \approx 1$).

$$\pm Xp_l \pm \frac{W_F}{2}\Theta_l \leq \frac{W_M - W_F}{2} \qquad \pm Yp_l \pm \frac{H_F}{2}\Theta_l \leq \frac{H_M - H_F}{2} \qquad (5.6)$$
$$\text{for} \quad l \in \{1, 2, \cdots, L\}$$

(ii) *Maximum field constraints* ensure that mask floorplanning does not move the die copies too far apart causing the field size to become too large.

$$Xf_{C-1} + W_D \leq W_F \qquad Yf_{R-1} + H_D \leq H_F \qquad (5.7)$$

(iii) *Die overlap constraints* ensure that the die copies do not overlap.

$$Xf_1 \geq W_D \qquad Xf_{c+1} - Xf_c \geq W_D \quad \text{for} \quad c \in \{1, 2, \cdots, (C-2)\} \qquad (5.8)$$
$$Yf_1 \geq H_D \qquad Yf_{r+1} - Yf_r \geq H_D \quad \text{for} \quad r \in \{1, 2, \cdots, (R-2)\} \qquad (5.9)$$

(iv) *Maximum allowed rotation* restricts the maximum angle by which we can rotate the mask blank.

$$-\Theta_{max} \leq \Theta_l \leq \Theta_{max} \quad \text{for} \quad l \in \{1, 2, \cdots, L\} \qquad (5.10)$$

This leads to a total of $8L + 2 + (C-1) + (R-1) + 2L$ linear constraints.

A key part of this defect avoidance methodology is to model the CD impact of defects as a function of these pattern shift, rotation and mask floorplanning variables. Suppose a mask defect d, with center (X_d, Y_d) relative to the mask center, lies on the mask blank corresponding to layer l. To account for pattern shift and rotation, the defect coordinates can be modified as shown in Equation 5.11 and Equation 5.12.

$$\hat{X_d} = X_d cos(\Theta_l) - Y_d sin(\Theta_l) - Xp_l \qquad (5.11)$$

$$\hat{Y_d} = X_d sin(\Theta_l) + Y_d cos(\Theta_l) - Yp_l \qquad (5.12)$$

Next let us consider one vertical edge of an absorber shape e with x-coordinate $X(e)$ and y-coordinates
$(Y_{low}(e), Y_{high}(e))$, relative to the die origin. If the absorber edge is a part of a die in the r^{th} row and c^{th} column, we can write the coordinates of the edge relative to the mask origin as shown in Equation 5.13.

$$\hat{X}(e) = X(e) + Xf_c$$
$$\hat{Y}_{low}(e) = Y_{low}(e) + Yf_r \qquad \hat{Y}_{high}(e) = Y_{high}(e) + Yf_r \tag{5.13}$$

We can then compute the distance of the edge from the defect using Equation 5.14, where $u(y)$ is the step function that is one if $y \geq 0$, else it is zero. Using this distance, we can then compute the CD impact of the defect on the layout shape using Equation 5.2 and Equation 5.3.

$$dist(e,d)^2 = (\hat{X}_d - \hat{X}(e))^2 + (\hat{Y}_d - \hat{Y}_{low}(e))^2 u(\hat{Y}_{low}(e) - \hat{Y}_d)$$
$$+ (\hat{Y}_d - \hat{Y}_{high}(e))^2 u(\hat{Y}_d - \hat{Y}_{high}(e)) \tag{5.14}$$
$$\tag{5.15}$$

In order to ensure that the die works, we must ensure that the CD impact of the defect is less than the CD tolerance for every absorber edge. Since the number of mask defects is significantly smaller than the number of absorber edges in the field pattern, we assume that a single absorber edge is not affected by more than one defect. Moreover, a defect only impacts a small set of absorber edges around it, so for any given floorplan solution, we only need to look at the absorber shapes within a certain distance of the defect. For this method, we take this distance as $3 * W_d$. This significantly reduces the overhead of checking every defect-absorber edge pair of the mask pattern. The CD tolerance value for any absorber edge could be a single value assigned to all absorber shapes, or design-aware, as done in mask inspection (Chapter 3).

5.4.2 Random Walk + Gradient Descent Based Solution Method

The objective of EUV mask defect avoidance is to determine a feasible value of $Xp_l, Yp_l, \Theta_l, Xf_c$ and Yf_r such that all the spatial constraints and CD tolerance constraints are obeyed. The spatial constraints are simple linear constraints. However, the CD tolerance constraints are non-convex as proven below.

Theorem: For any absorber edge defect pair, the constraint $CD_{def}(e,d) \leq CD_{tol}(e)$ is non-convex.

Proof: Consider a left vertical edge of an absorber shape as shown in Figure 5.9 below. Let us consider the multi-variable function $f(Xp_l, Yp_l, \Theta_l, Xf_c, Yf_r) = CD_{def}(e,d) - CD_{tol}(e)$. By analytically computing the partial second derivative with respect to any of the pattern shift (Xp_l, Yp_l), rotation (Θ_l) or floorplanning variables $(Xf_c,\ Yf_r)$ we find that it is not guaranteed to be positive for all possible defect-absorber edge positions. This proves that all the CD tolerance constraints are non-convex. Geometrically, we can consider two potential defect locations relative to this edge, as shown in Figure 5.9. Both defect locations obey the CD constraint, but the line segment connecting them contains potential defect locations which would cause a CD violation. This implies that the geometric space of feasible defect locations relative to a single absorber edge is non-convex.

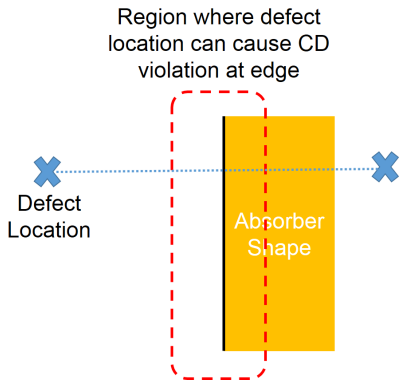


Figure 5.9: Illustration of non-convexity of CD constraint showing that two feasible defect locations and the segment connecting them crosses through the prohibited region for an absorber edge.

Since handling non-convex constraints is very hard in optimization, we relax the CD tolerance constraints by converting it into an objective function that we then minimize. We use the sigmoid penalty function to relax every CD constraint ($sigmoid(x) = \frac{1}{1+e^{-\alpha x}}$, $\alpha = 4.0$ for this work). As a result, the cost function for our optimization problem is the sum of sigmoids for all the relevant defect-absorber edge pairs, as shown in Equation 5.16. Hence our overall optimization problem is to find the pattern shift, rotation and mask floorplanning variables to minimize this sigmoid cost function while obeying the linear spatial constraints of Equations 5.6-5.10.

$$Cost_{GO} = \sum_{l \in L} \sum_{d \in BD_l} \sum_{e \in A_l} sigmoid(CD_{def}(e,d) - CD_{tol}(e)) \tag{5.16}$$

To solve the non-convex optimization problem for EUV mask defect avoidance, we use a combination of random walk and gradient descent. Random walk is used to perform a coarse grained search over the multi-dimensional linear polytope formed by the spatial constraints. For each of the sample points generated by random walk, we use gradient descent for local search in the vicinity of the sample. The overall method is summarized in Figure 5.10.

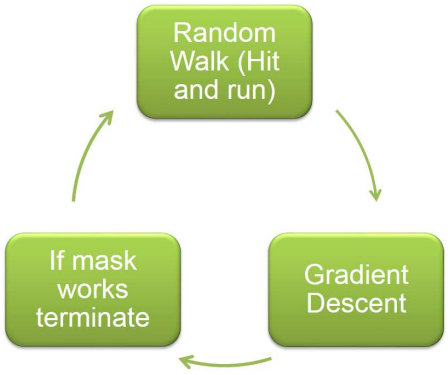


Figure 5.10: Illustration of the method used to solve the EUV mask defect avoidance problem.

We use hit-and-run based Markov chain random walk, which is known to mix fast [Lov98]. Starting from an initial point inside the linear polytope, hit-and-run finds new points inside the linear polytope using the following steps:

(i) Draw a line in a randomly chosen direction passing through the given point.

(ii) Find the two points where this line intersects the linear polytope.

(iii) Pick a random point on the line segment connecting the two points above.

(iv) Go back to Step 1 with this new random point.

In order to apply gradient descent to each sample point generated by random walk, we need to analytically compute the gradient of the cost function of Equation 5.16. The analytical expression for gradient of one defect and vertical absorber edge pair is shown in Equation 5.17. The intermediate variables Z_1, Z_2 and U_Y are shown in Equations 5.18, 5.19 and 5.20, respectively. Note that the discontinuity of the cost function at $\hat{Y}_{low}(e)$ and $\hat{Y}_{high}(e)$ is handled by function U_Y in Equation 5.20 by assuming that only one of the three conditions will hold during a round of gradient descent. Since gradient descent moves in small steps, this assumption is reasonable.

$$\begin{array}{rcl}
\frac{\partial Cost_{GO}}{\partial(Xp_l)} & = & \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} \frac{\partial(dist(e)^2)}{\partial(Xp_l)} = -2Z_1Z_2 \cdot (\hat{X}_d - Xf(e)) \\
\frac{\partial Cost_{GO}}{\partial(Yp_l)} & = & \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} \frac{\partial(dist(e,d)^2)}{\partial(Yp_l)} = -2Z_1Z_2 \cdot U_Y \\
\frac{\partial Cost_{GO}}{\partial(\Theta_l)} & = & \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} \frac{\partial(dist(e,d)^2)}{\partial(\Theta_l)} = -2Z_1Z_2 \cdot (X_d \sin\Theta_l + X_d \cos\Theta_l) \\
\frac{\partial Cost_{GO}}{\partial(Xf_c)} & = & \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} \frac{\partial(dist(e,d)^2)}{\partial(Xf_c)} = -2Z_1Z_2 \cdot (\hat{X}_d - Xf(e)) \\
\frac{\partial Cost_{GO}}{\partial(Yf_r)} & = & \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} \frac{\partial(dist(e)^2)}{\partial(Yf_r)} = -2Z_1Z_2 \cdot U_Y
\end{array} \tag{5.17}$$

$$Z_1 = \frac{\partial Cost_{GO}}{\partial CD_{def}(e,d)} = \alpha \cdot sig(CD_{def}(e,d) - CD_{tol}(e)) \cdot (1 - sig(CD_{def}(e,d) - CD_{tol}(e))) \tag{5.18}$$

$$Z_2 = \frac{\partial CD_{def}(e,d)}{\partial(dist(e,d)^2)} = \frac{3\gamma \cdot \sqrt{I_{NoDef} m_{def}}}{ImageSlope} \cdot DefHeight(e,d) \cdot \frac{-1}{(W_d/2)^2} \tag{5.19}$$

$$U_Y = \begin{cases} (\hat{Yd}_l(n) - \hat{Y}_{low}(e)), & \text{if } \hat{Yd}_l(n) \leq \hat{Y}_{low}(e) \\ 0, & \text{if } \hat{Y}_{low}(e) \leq \hat{Yd}_l(n) \leq \hat{Y}_{high}(e) \\ \hat{Yd}_l(n) - \hat{Y}_{high}(e), & \text{if } \hat{Yd}_l(n) \geq \hat{Y}_{high}(e) \end{cases} \tag{5.20}$$

For computing the gradient of the cost function, we need to find all the interacting defect-absorber edge pairs, calculate the analytical expressions of Equation 5.17 for each such pair and then add them. Since the number of defects are typically much smaller than the number of absorber shapes on the mask, we do this by iterating over all the defects and finding all the absorber shapes within a certain distance ($3W_d$) of each defect. We then compute the gradient for each absorber edge within this radius of influence of the defect. Finding all absorber shapes within a certain radius of a defect can be done efficiently by storing the entire mask layout in a 2D region query tree data-structure [Ben75].

The running time for computing the gradient during the iterations of local search is dominated by the process of querying the large layout repeatedly. Since only small moves are made during local search, we can avoid this overhead by upfront storing all the absorber shapes that could be affected by any defect when we make small local moves. For examples, if we set the maximum number of gradient descent iterations for each random starting solution as N_G and the gradient step size is S, we can upfront store all the absorber shapes that are within a radius of $2N_GS + 3Wd_l(n)$ of a particular defect. At the start of the gradient descent iterations, we store all such shapes for each defect. As a result, we do not need to query the large layout every time the gradient needs to be computed.

5.5 Results and Discussion

Our proposed EUV mask defect avoidance method has been implemented in C++. OpenAccess API has been used to read and access layout shapes [oa]. Eigen Matrix library is used to handle vectors and matrix operations [Gue10]. All our results are shown for an ARM Cortex M0 processor layout which was synthesized, placed and routed using Cadence Encounter with $32nm$ Synopsys Standard Cell Library. The layout is then scaled to $8nm$ technology node to show our results. We apply defect avoidance to the polysilicon layer, unless otherwise stated. Although we set the CD tolerance of every absorber shape edge to 10% of the technology node

($CD_{tol} = 0.8nm$), it is also possible to make the CD tolerance assignment design-aware, as done in Chapter 3.

We assume a single size for all the defects, with peak height $H_d = 2nm$ and FWHM $W_d = 50nm$ except in Section 5.5.4. Due to the lack of any real data on spatial distribution of buried defects, we assume that defects are uniformly distributed accross the entire usable area of the mask. 100 randomly generated spatial defect maps are considered. The main quality metric for evaluating the efficacy of defect avoidance is *mask yield*, which we define as the percentage of random defect maps that are made usable (i.e. there is no impact on chip yield) by defect avoidance. We show mask yield for different number of defects on the mask to highlight acceptable defect density levels.

We set the number of gradient descent iterations for each random point obtained from hit-and-run to 50 and the step size to $1nm$. We fix the number of random walk iterations as the ratio of volume of the linear spatial polytope and the volume of the multi-dimensional ($3L + R + C - 2$ dimensions) ball that is covered by gradient descent. The rationale behind this choice is to ensure equivalent coverage of the available space when we compare different scenarios. The volume of the linear polytope was computed using the tool VINCI [BE], and the volume of the gradient ball can be computed using a simple analytical expression [nba].

We chose the mask field size such that four rows and three columns of die copies of the Cortex M0 ARM processor can be placed inside the mask field. We allow a maximum pattern shift of $20\mu m$, small-angle rotation of 6^o and maximum allowed scribe area of 1% of the mask field size. *Scribe area* is defined as the difference in area between the total area of all the die copies inside the field pattern and the total field size ($W_F \times H_F$). Since the size of one ARM Cortex M0 layout is $162\mu m \times 159\mu m$, the total field size becomes $486\mu m \times 636\mu m$ and the usable area of the mask is $511\mu m \times 662\mu m$. Note that although this is much smaller than the full field size of $132mm \times 104mm$, we have analyzed smaller layouts in order to get reasonable runtimes, especially since we perform Monte Carlo analysis over 100 random defect maps. *Because our analysis is done for small mask size, the mask yield values*

Table 5.2: Comparison of mask yield after defect avoidance using prohibited region based method with our simulated annealing based method.

Defect Count	Prohibited Region		Simulated Annealing	
	Pattern Shift [ZD12]	Pattern Shift + Rotation	Pattern Shift +	Pattern Shift + Mask Floorplanning
10	100%	100%	100%	100%
20	81%	100%	100%	100%
30	8%	97%	0%	6%
40	1%	11%	0%	0%
50	0%	0%	0%	0%

we report for different defect density levels in this section may not correspond to realistic values in production. Nevertheless, the analysis in this section is sufficient to evaluate the efficacy of our proposed mask defect avoidance method and compare it with prior work.

5.5.1 Comparison with Other Defect Avoidance Methods

Mask yield after defect avoidance using prohibited region based defect avoidance method [ZD12, ZDW12] with our simulated annealing based method is shown in Table 5.2. Prohibited region based defect avoidance methods allow continuous pattern shift and small angle rotation, but cannot handle mask floorplanning. Simulated annealing based defect avoidance method allows pattern shift and mask floorplanning, but small-angle rotation is not possible. With our implementations of both these methods, prohibited region method performs significantly better than simulated annealing method because the prohibited region method allows continuous pattern shift instead of making discrete jumps. As a result, the solution space is explored more efficiently.

Table 5.3 shows mask yield using our defect avoidance method, using the different degree of freedom. Notice that even if defect avoidance is limited to pattern

Table 5.3: Summary of mask yield after our global optimization based defect avoidance method with different degrees of freedom

Defect Count	Pattern Shift	Pattern Shift + Rotation	Pattern Shift + Mask Floorplanning	Pattern Shift + Rotation + Mask Floorplanning
10	100%	100%	100%	100%
20	100%	100%	100%	100%
30	35%	91%	55%	100%
40	3%	10%	9%	74%
50	0%	1%	2%	13%

shift, our method performs better than both the prohibited region and simulated annealing methods. Our method performs significantly better than the prohibited region method because prohibited rectangle construction is inherently pessimistic at corners of absorber shapes, as illustrated in Figure 5.11 (CD impact of defect depends to Euclidean distance from absorber edge). When pattern shift and rotation are both allowed but mask floorplanning is not, our method is slightly worse than the prohibited region method because of the number of random walk iterations that we set. Given enough iterations, our method can always reach the best possible solution. More importantly, by allowing mask makers to exploit all three degrees of freedom for defect avoidance, our method allows significantly better mask yield compared to these earlier approaches. For a 40-defect mask, the mask yield of prohibited region based defect avoidance with rotation is just 11%. Our method is able to improve the mask yield to 74% in this case.

The running time complexity of all the defect avoidance methods we described in this section is $O(\sum_{l \in L} |A_l| \log(|A_l|))$, since the running time depends on the the region query operation to obtain all the layout shapes within a rectangular box. Note that the number of queries depends on the number of defects for all the defect avoidance methods. It also depends on the number of random iterations for the simulated annealing method, and our global optimization method. The average

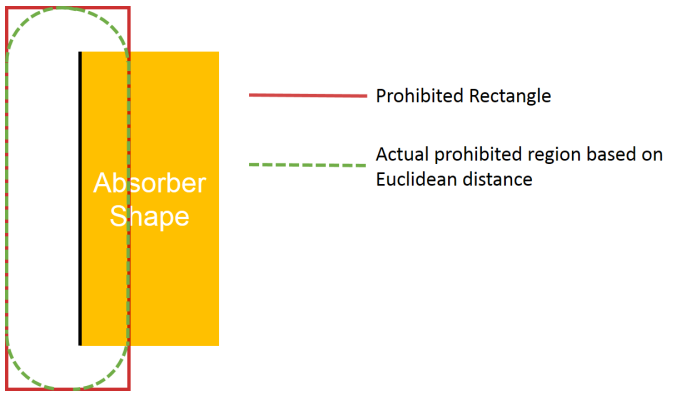


Figure 5.11: Pessimism of prohibited rectangle construction compared to true prohibited region based on Euclidean distance for one absorber edge.

running time across all the 100 random defect maps with 40 defects that we analyzed is shown in Figure 5.12. These results show that the performance of the different methods depends on the degree of freedom. If pattern shift and rotation are the two degrees of freedom that are allowed, our method is faster than the prohibited region method. However, for most other scenarios, our method does require additional computation time to achieve better mask yield. Note that since we consider a smaller field size in this work, the reported running time is much less than the time it would take for a real full-field chip.

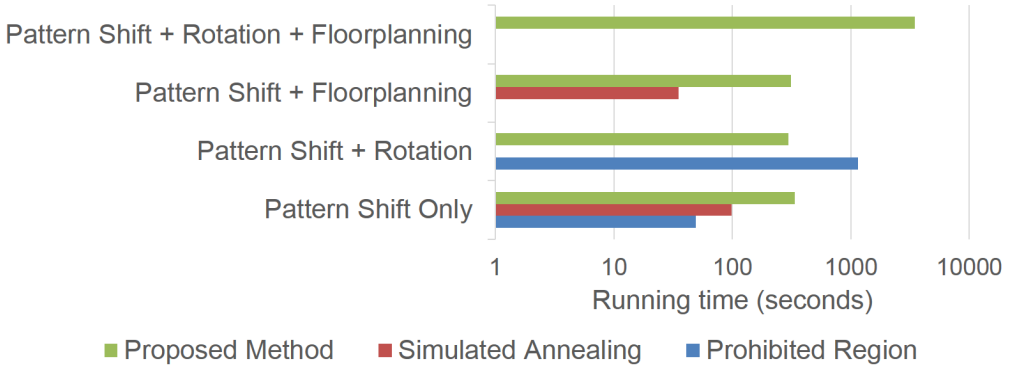


Figure 5.12: Average running time of the three defect avoidance methods with different degrees of freedom for a 40-defect mask. Note that the mask yield of the different methods in reported in Table 5.2 and Table 5.3. Our proposed method has the largest mask yield followed by the prohibited region and simulated annealing methods, respectively.

Although Figure 5.12 may suggest that defect avoidance methods will require considerable running time for a full-field mask, there are several simple techniques that could improve the running time significantly. Our global optimization method can be easily parallelized because gradient descent for each random starting point can be done independently. Hence, the critical region query operation can be performed in parallel which would enable signficant performance improvement. Moreover, by using a hierarchical layout, each query operation can be made significantly faster.

5.5.2 Analysis for Multiple Layer Defect Avoidance

Our earlier analysis focused on just the polysilicon layer, which is typically the most critical layer. If more layers need to be patterned using EUV lithography, defect avoidance needs to be applied for each of the corresponding masks. Although pattern shift and rotation can be done independently for each of these layers, mask floorplanning must be done together to ensure alignment.

As described in Section 5.4.1, our method can handle multiple layer defect avoidance as well. However the number of variables increases by three every time an additional layer is patterned using EUV. If we were to set the number of random walk iterations based on volume of the linear polytope and gradient ball as done earlier, then the number of random walk iteration would be around 10^{10}. Since this would require considerable running time, we decided to fix the number of random walk iterations as 10^7 for all cases in this subsection. This makes the exploration of the solution space less efficient for multi-layer cases.

We have summarized the results for single layer (polysilicon only), two layer (polysilicon and active) and four layer (polysilicon, active, contact and metal 1) scenarios in Figure 5.13. *Note that mask yield here is defined as the percentage of cases where all the layers work.* Consequently, mask yield is lower for multi-layer defect avoidance. Mask yield is close to 100% for all cases, when the number of defects is 30 or less. Then the mask yield for multiple layer cases reduces as we add more layers.

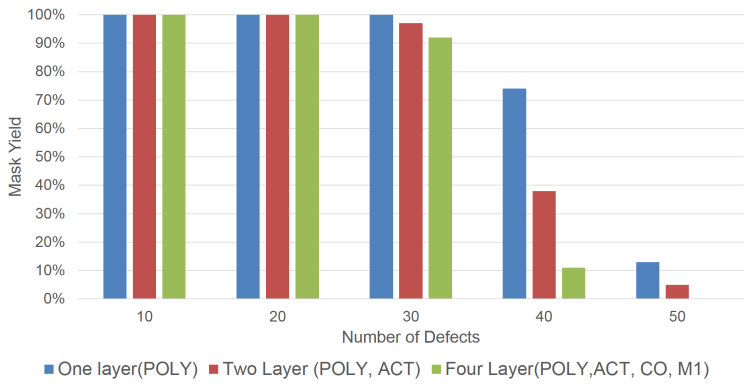


Figure 5.13: Comparison of mask yield after defect avoidance when multiple layers of a design are patterned using EUV lithography.

An important concern for multi-layer defect avoidance is *mask blank assignment*, i.e. determining the mask blank on which each design layer should be patterned. More precisely, given L design layer and L mask blanks with known defect maps, the goal of mask blank assignment is to map each layer to one of the L mask blanks so that the best mask yield can be achieved. For the results shown in Figure 5.13, we use the simple strategy of applying defect avoidance to each possible blank mapping solution and picking the mapping that works, which we refer to as *complete mapping*.

Complete mapping, which is similar to the blank assignment problem explored by Du et al. [DZ11], requires applying defect avoidance $L!$ times, making it slow. However, it gives the best possible mask yield. Figure 5.14 compares complete mapping to *random mapping*, where each layer is assigned to a mask blank randomly and defect avoidance is applied just once.

As confirmed by Figure 5.14, blank mapping can have a huge impact on mask yield. For the four layer defect avoidance with 30-defect mask blanks, the difference is mask yield between complete and random mapping is more than 70%. This is because the mask yield is limited by the regular and unidirectional polysilicon layer as discussed in more detail in Chapter 6. Complete mapping allows the critical polysilicon layer to pick one of four mask blanks, which leads to significantly better mask yield.

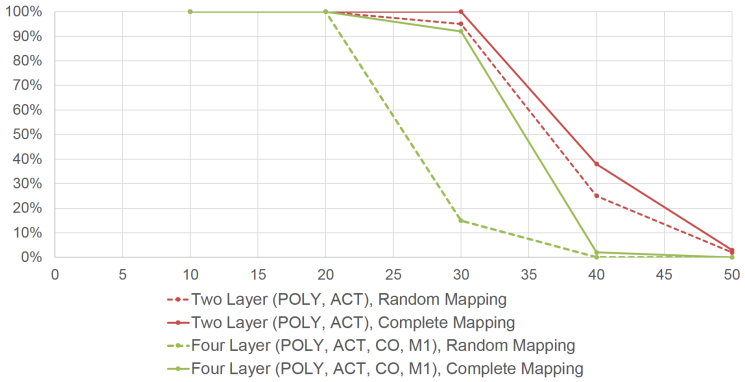


Figure 5.14: Comparison of mask yield for multiple layer defect avoidance with complete blank mapping and random blank mapping.

5.5.3 Impact of Spatial Constraints on Defect Avoidance

There are three key manufacturing constraints (corresponding to each of the three degrees of freedom) that strongly affect the potential benefit of defect avoidance:

(i) *Maximum pattern shift* is the difference between the size of usable area of mask and the size of pattern field ($(W_M - W_F) \times (H_M - H_F)$ in Equations 5.6).

(ii) *Maximum rotation angle* is the largest angle by which the mask blank can be rotated relative to the field pattern. It is the value of Θ_{max} in Equation 5.10.

(iii) *Maximum scribe area* is the difference in area between the total area of all the die copies inside the field pattern and the total field size ($W_F \times H_F$), expressed as a percentage of the total field area.

These three manufacturing constraints limit the solution space available for avoiding defects and hence can affect the mask yield strongly. We shall analyze the impact of each of these constraints in this subsection. For the sake of brevity, we shall only analyze the single layer scenario (polysilicon layer), and we will report the mask yield for 40-defect masks.

The impact of maximum pattern shift is shown in Figure 5.15. Note that all our prior analysis was done assuming a maximum pattern shift of $20\mu m$. Here we look at values ranging from $10\mu m$ to $100\mu m$. For the layout we chose to analyze, mask

yield for 40-defect mask was 100% for pattern shift values larger than $50\mu m$. We also computed the volume of the linear polytope formed by all the spatial constraints of the defect avoidance optimization problem because this volume is a good indicator of the potential mask yield benefit due to the change in the size of the solution space.

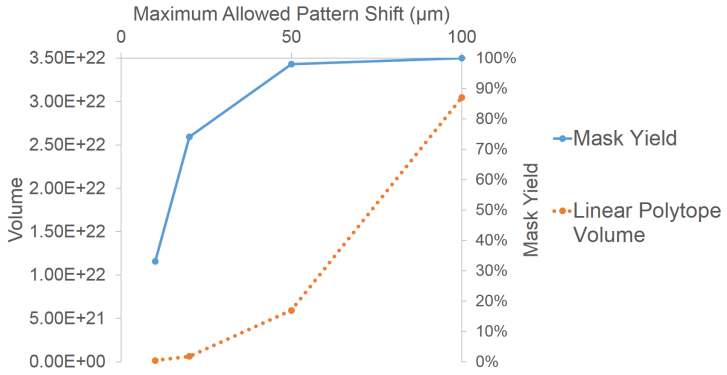


Figure 5.15: Volume of linear polytope and mask yield for 40-defect mask with respect to maximum allowed pattern shift.

Similarly, the benefit of rotation is highlighted in Figure 5.16. Both the mask yield for a 40-defect mask, and linear polytope volume is plotted for maximum rotation angle (Θ_{max}) ranging from 0 degrees to 10 degrees. The interesting thing to note here is that mask yield saturates at around 80%. The reason for this is that the overall solution space does not grow due to reticle boundary constraints, which is confirmed by the polytope volume in Figure 5.16.

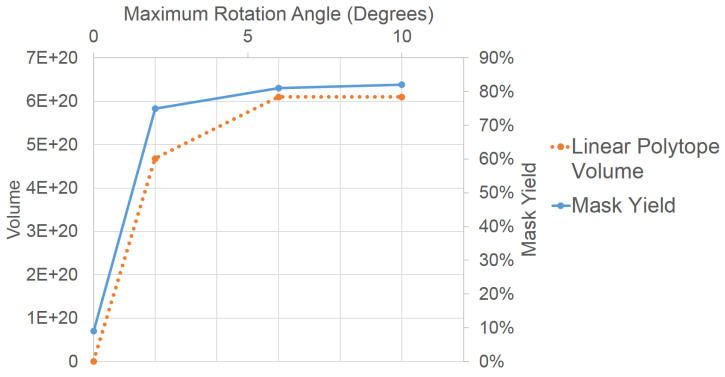


Figure 5.16: Volume of linear polytope and mask yield for 40-defect mask with respect to maximum allowed rotation.

Lastly the impact of scribe area is shown in Figure 5.17. Mask yield can improve up to 100% with scribe area of 5%. However, this improvement comes at the expense of wasted space on the wafer.

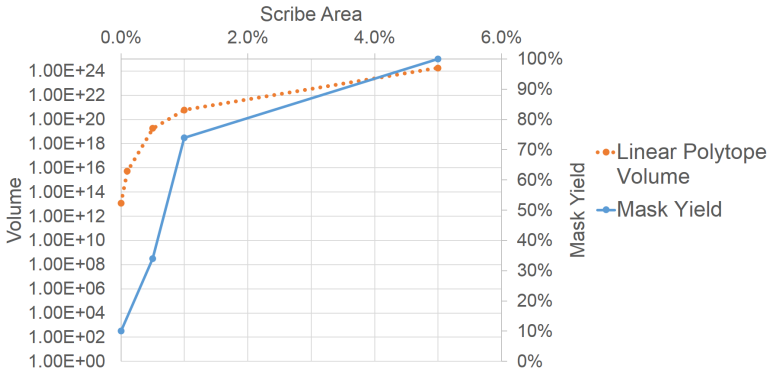


Figure 5.17: Volume of linear polytope and mask yield for 40-defect mask with respect to maximum allowed scribe area.

5.5.4 Impact of Defect Size Distribution

We have assumed that every defect is the same size with $W_d = 50nm$ and height $H_d = 2nm$ so far. However, in real masks, different defects will have different sizes. In this subsection, we shall assume that both the FWHM and height of every defect are independent random variables with probability density function (PDF) as described in Equation 5.21. Although there is little experimental data available on EUV mask defect size distribution, we chose this distribution since it is frequently used to model wafer defect sizes [GP10].

$$P(r) = \begin{cases} \frac{r}{r_0^2}, & 0 \leq r \leq r_0 \\ \frac{r_0^2}{r^3}, & r_0 \leq r \leq \infty \end{cases} \tag{5.21}$$

Here r_0 is a fitted parameter based on distribution statistics, and r is a random variable that corresponds to either height or FWHM of a defect.

Figure 5.18 compares the mask yield for the following three scenarios:

(i) All defects have constant size with height $2nm$ and FWHM $50nm$

(ii) Height and FWHM of each defect is derived from the PDF of Equation 5.21, with mode (value with maximum probability, in this case r_0) equal to $2nm$ and $50nm$, respectively.

(iii) Height and FWHM of each defect is derived from the PDF of Equation 5.21, with expected value (in this case $\frac{4}{3}r_0$) equal to $2nm$ and $50nm$, respectively.

These results show that mask yield is the lowest when the mode of defect height and FWHM is $2nm$ and $50nm$, respectively (Scenario 2). This is because the expected value of defect height and FWHM is $2.67nm$ and 66.67, respectively, which is larger than the other two scenarios. Comparing the two cases with the same expected value of defect height and FWHM (Scenario 1 and 3), mask yield is better when defect size is not constant. Since defect size follows the probability distribution specified in Equation 5.21, most defects are smaller than the expected value and only a few are larger than the expected value. Defect avoidance is able to handle this better than constant sized defects by placing the smaller number of large defects in sparse regions of the mask field pattern.

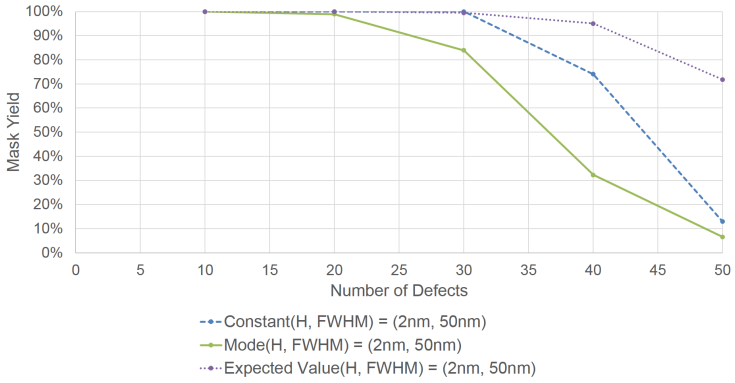


Figure 5.18: Comparison of mask yield for different defect size distributions.

5.6 Conclusions

In this work, we proposed an EUV mask defect avoidance method that can explore all the available degrees to freedom: pattern shift, rotation and mask floorplanning.

Our method can handle multiple layers of a design and explores the continuous solution space instead of discretizing it.

We modeled EUV mask defect avoidance as a global optimization problem with non-convex objective and linear constraints. We then solved the problem using a combinatuon of hit-and-run based random walk and gradient descent. Compared to previously proposed methods for defect avoidance, our method can improve the probability of using a defective mask blank without any yield impact (mask yield) significantly, and hence our methods allows tolerance to a larger number/size of defects than possible with previous methods. For the polysilicon layer of a $8nm$ ARM Cortex M0 layout, our defect avoidance method was able to improve mask yield by more than 60%-point compared to prior approaches for a 40-defect mask.

Using our method, we have also compared the potential mask yield benefit of each degree of freedom. Our analysis shows that pattern shift has the most impact on mask yield since increasing the maximum allowed pattern shift always improves mask yield. Rotation and mask floorplanning also help in improving mask yield but their benefit is not as significant as pattern shift because increasing the maximum allowed rotation angle or scribe area improves mask yield only up to a certain limit.

CHAPTER 6

EUV-CDA: Pattern Shift Aware Critical Density Analysis for EUV Mask Layouts

6.1 Introduction

As discussed in Chapter 5, defect avoidance is likely to play a key role in enabling EUV lithography. A likely design to fabrication flow for EUV masks is illustrated in Figure 6.1. A mask shop will typically have a collection of inspected mask blanks with known defect locations. Since each critical layer of the taped out design must be patterned on a defective mask blank, the mask shop must apply defect avoidance to find a defective mask blank that works for each layer. *Given a defect density and size distribution, the probability of finding a mask blank from a large set of blanks on which the given design layout can be patterned without causing any yield loss is referred to as mask yield.*

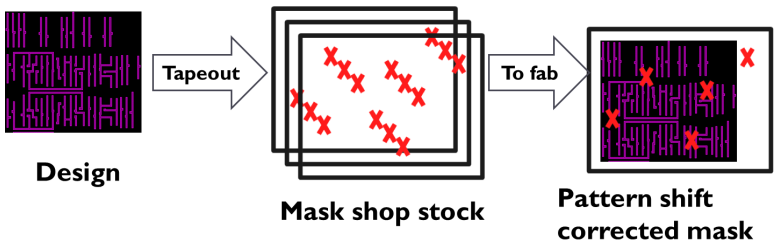


Figure 6.1: Set of steps involved in a EUV mask shop involving pattern shift based mask defect mitigation.

Certain layout topologies may be more capable of tolerating mask defects, and exploiting the benefits of defect avoidance strategies. An understanding of what characteristics of a layout can make it more robust to EUV mask defects can signif-

icantly aid EUV layout design and even the formulation of design rules. In order to develop any layout or design level techniques to create robust layouts, a quantifiable metric that characterizes the robustness of layouts to such mask defects is needed. In addition to layout optimization, such a layout robustness metric can be used for mask blank assignment as well. Layouts with low mask yield can be assigned to a mask blank with lower defect density. This would save the computational effort of performing pattern shift for each layout-blank pair, as done in [DZ11].

In this chapter, we propose a new metric, *critical density*, that evaluates the robustness of EUV layouts to mask defects. To the best of our knowledge, this work is the first attempt towards developing such a metric. This metric allows us to estimate pattern shift aware mask yield for any defect density using a simple analytical expression, and enables us to distinguish between layouts that have different mask yield for any given defect density.

The need for a metric that quantifies robustness of layouts to mask defects bears resemblance to conventional

critical area analysis (CAA) [GP10]. CAA is commonly used to check robustness of layouts to random $wafer$ defects through the use of statistical metrics that estimate chip yield for some random distribution of defects. The key features of this work that also distinguish EUV-CDA and conventional CAA are the following:

- Unlike wafer defects, a single mask defect will print on every copy of the design on the wafer. Hence, the goal of EUV-CDA is to predict mask yield, not chip yield.

- The impact of defect avoidance techniques on mask yield must be probabilistically modeled as a part of EUV-CDA since actual defect locations are not known at the design stage. Pattern shift based defect avoidance is modeled in this work since it is the most popular defect avoidance strategy [ETS12].

- The need to account for pattern shift means that mask failure depends on the simultaneous location and size of several defects on the mask. This is in con-

trast to conventional CAA, where every defect can cause failure independently. This dependence complicates the analysis significantly, requiring much more computational effort and modeling.

The remainder of this chapter is organized as follows. Prohibited region of a layout is described in Section 6.2. In Section 6.3, we propose analytical methods to estimate mask yield for two limited scenarios. We describe our critical density method in Section 6.4. Experimental results are then presented in Section 6.5. We conclude this work in Section 6.6. All notation used in this paper is described in Table 6.1.

6.2 Prohibited Region

We define the prohibited region of a given layout, for a particular defect size s, as the set of polygons PB^s such that if the center of a mask defect lies inside any polygon $p \in PB^s$, the given mask layout pattern will not yield.

The method for constructing prohibited region is the same as proposed by Zhang et. al. [ZD12] with the additional step of merging the constructed rectangles. It is similar to the process of using simple Boolean operations to compute critical area in conventional CAA [GP10]. *But the criteria for determining prohibited region is CD tolerance, in constrast to opens/shorts in conventional CAA.* We chose this pessimistic approach since we are dealing with mask defects. Assignment of this CD tolerance to layout shapes can be done by either setting a single pessimistic value for all the patterns (10% of the technology node, in our case), or by using some design information (timing slack, redundant/dummy patterns) to assign an appropriate CD tolerance, as done in Chapter 3. Although EUV-CDA can be easily applied for such smart CD tolerance, for the sake of brevity we assign a single CD tolerance to all shapes in this work.

If pattern shift was not a part of EUV mask manufacturing, estimating mask yield from prohibited region would be fairly straightforward. Assuming a uniform spatial

Table 6.1: Glossary of Terminology

Term	Description
s	Defect size (Height, full width half maximum pair)
PB^s	Prohibited Region for defect size s
$P(s)$	Probability of occurrence of defect size s
A_M	Mask area
D_P^s	Prohibited region density ($\frac{Area(PB^s)}{A_M}$)
D_P	Expected prohibited region density ($\sum_s P(s)D_P^s$)
K	Number of different defect sizes considered
N_d	Number of defects on mask
Δ_X	Available pattern shift in X direction
Δ_Y	Available pattern shift in Y direction
A_Δ	Total pattern shift area ($\Delta_X \times \Delta_Y$)
L_X	Design layout width
L_Y	Design layout height
A_L	Total design area ($L_X \times L_Y$)
ρ	Number of prohibited region shapes per unit area
N_X	Number of discrete X direction shifts
N_Y	Number of discrete Y direction shifts
(X_i, Y_j)	Potential pattern shift solution
E_{ij}^Δ	Event that pattern shift solution (X_i, Y_j) works
$P(E_{ij}^\Delta)$	Probability of event E_{ij}^Δ
PB_{ij}^s	Prohibited region for defect size s, shifted by (X_i, Y_j)
$Ar(PB_{ij}^s)$	Total area of all the polygons in the prohibited region
W_{pl}	Width of periodic parallel line structure
P_{pl}	Pitch of periodic parallel line structure
Y_M^{ct}	Mask Yield of periodic contact array layout
D_{cric}	Critical density
Y_M^{true}	Accurate mask yield (Monte Carlo method)
N_d^{min}	Minimum defect count considered
N_d^{max}	Maximum defect count considered
AC	Autocorrelation matrix
ACF	FFT of AC
pix	Pixel size (Sampling size) for computing AC
N_F	Number of terms from ACF used for predicting D_{cric}

distribution of defects on the mask, mask yield could be estimated as $(1 - D_P^s)^{N_d}$ since every defect must lie outside the prohibited region[1]. This simple approach would imply that any two design layouts with the same prohibited region density will have the same mask yield. But if pattern shift is used to avoid mask defects, layout topology may also affect mask yield. To confirm this suspicion, we created four $20\mu m \times 20\mu m$ layouts such that their prohibited region is a set of parallel lines with pitch $80nm$. The width of the lines was treated as a Gaussian random variable with mean $20nm$. Different values of variance (σ) were used to construct the four layouts. We compared the mask yield of these four layouts, which is estimated using rigorous Monte Carlo simulation (described in Section 6.3). The results, shown in Figure 6.2, highlight the huge difference in post pattern shift mask yield between the layouts which have very similar prohibited region density (confirmed by the pre-pattern shift mask yield of the four layouts, which are almost same).

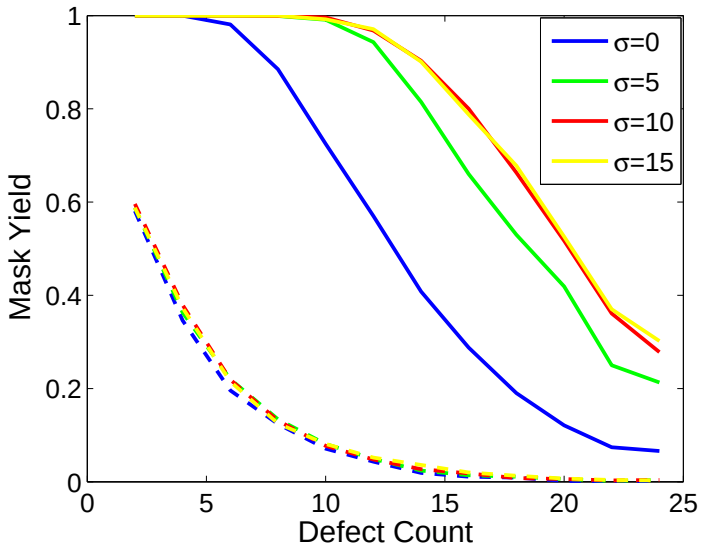


Figure 6.2: Comparison of pre-pattern shift (dashed lines) and post-pattern shift (solid lines) mask yield of four parallel line layouts with same prohibited region density but different σ of Gaussian width.

[1] All defects are assumed to be of size s in this example.

6.3 Approximate Analytical Methods

Monte Carlo based mask yield estimation is a simple, but computationally expensive strategy of generating random defect maps and performing pattern shift for each defect map. Mask yield can then be computing as the ratio of samples for which the final mask works, i.e. every defect on the mask is avoided. The Monte Carlo method starts off by constructing the prohibited region for different defect sizes. We then generate a defect map with N_d defects, assigning a size to each defect based on the given defect size distribution $P(s)$. Pattern shift is then applied for this defect map to determine if a feasible solution exists. Note that for each Monte Carlo iteration, N_d defects need to be generated since pattern shift makes mask failure dependent on location of several defects on the mask. This dependence necessitates a large number of Monte Carlo iterations to achieve convergence. The methodology we used for pattern shift is the same as the approach proposed by Wagner [WB12], which is optimal with respect to mask yield.

This naive method of estimating mask yield, although accurate, is cumbersome and slow, requiring many Monte Carlo iterations to give accurate results. Therefore this method of estimating mask yield is impractical for realistic layouts. Moreover, the method does not provide any design insights that could help improve the mask yield of a given layout. Despite these limitations, the accuracy of this method makes it appropriate for validating the faster, approximate method that we shall propose in this paper.

6.3.1 Inclusion-Exclusion Method

In this section, we propose a method to estimate the mask yield with one simplifying assumption: pattern shift picks a feasible solution from a finite set of alternatives.

Let us first discretize all the potential defect sizes, into K discrete defect sizes, $s_1, s_2, ..., s_K$ with respective probabilities of occurrence, $P(s_1), P(s_2), ..., P(s_K)$. For a uniform spatial distribution of defects, we can then calculate the probability that a particular pattern shift solution (X_i, Y_j) works using Equation 6.1. Note that

$\frac{Ar(PB_{ij}^{s_k})}{A_M}$ is equal to the prohibited region density for defect size s_k since shifting the polygons does not change area.

$$P(E_{ij}^{\Delta}) = \left(\sum_k P(s_k) \left(1 - \frac{Ar(PB_{ij}^{s_k})}{A_M} \right) \right)^{N_d} \tag{6.1}$$

Pattern shift aware mask yield can be estimated as the union of all the events E_{ij}^{Δ} since the mask will yield if any of the potential solutions work (Note that a solution is picked once the defect locations are known). Calculating this union of events can be done using inclusion-exclusion principle as shown in Equation 6.2.

$$P(\cup_{i,j} E_{ij}^{\Delta}) = \sum P(E_{mn}^{\Delta}) - \sum P(E_{pq}^{\Delta} \cap E_{mn}^{\Delta}) \ldots \tag{6.2}$$

The second order intersection term $P(E_{pq}^{\Delta} \cap E_{mn}^{\Delta})$ corresponds to the event that all defects lie in the non-prohibited region of both solution E_{pq}^{Δ} and E_{mn}^{Δ}. Hence, it can be computed using a polygon Boolean OR operation as shown in Equation 6.3.

$$P(E_{pq}^{\Delta} \cap E_{mn}^{\Delta}) = \left(\sum_k P(s_k) \left(1 - \frac{Ar(PB_{pq}^{s_k} \cup PB_{mn}^{s_k})}{A_M} \right) \right)^{N_d} \tag{6.3}$$

Computation of all the inclusion exclusion terms comprising $N_X \times N_Y$ orders (only first two order are shown above), is a $\#P$-complete combinatorial enumeration problem since it requires the computation of $2^{N_X \times N_Y}$ terms [KLS96]. This computational limitation, along with the quantization error incurred due to the assumption that the pattern shift solution space is discrete, make this method unsuitable for estimating mask yield for any realistic layouts.

Despite the impracticality of the inclusion-exclusion method, it does provide one interesting insight: *in addition to prohibited region density, mask yield depends on autocorrelation of the prohibited region of the input layout.* The second order terms in Equation 6.3, $Ar(PB_{pq}^{s_k} \cup PB_{mn}^{s_k})$, are linearly related to $Ar(PB_{pq}^{s_k} \cap PB_{mn}^{s_k})$, which measures the degree of overlap between the prohibited region PB^{s_k} with a shifted transform of PB^{s_k} (shifted by $(X_p - X_m, Y_q - Y_n)$). Each such overlapping area corresponds to one entry of the autocorrelation matrix of the 2D binary prohibited

region signal, PB^{s_k}. Moreover, mask yield depends on the weighted sum of the prohibited region autocorrelation for different defect sizes. We will later leverage this dependence of mask yield on the weighted autocorrelation of prohibited region for critical density computation.

6.3.2 Spacings Method

In this sub-section, we will show that *if the prohibited region of a given layout is regular, the problem of finding the post-pattern shift mask yield can be mapped to the maximal spacing distribution problem, a classic geometric probability problem.* Note that pattern shift is assumed to be continuous here, unlike the previous sub-section. Also, we consider only one defect size s here, and the assumption on regularity is for the prohibited region, PB^s.

First, suppose the prohibited region of the entire layout is a periodic parallel line structure. Assuming the lines are infinitely long and parallel to Y axis, any pattern shift in Y direction will not improve mask yield. Hence, only the X coordinates of the defects is relevant, and we can map all the defects to a single line. The periodicity assumption implies that the X coordinates of all the defects can be mapped to a modulo P_{pl} space with a single line of width W_{pl}. An optimal pattern shift based defect avoidance technique can successfully avoid all the defects, if and only if there exists a gap or spacing of size W_{pl} with no defect inside it. This mapping is illustrated in Figure 6.3.

This problem is equivalent to finding the probability of existence of a gap larger than $\frac{W_{pl}}{P_{pl}}$ on a unit circle with a uniform distribution of points [2]. This geometric probability, also referred to as the one dimensional maximal spacing problem [Pyk65], was first computed exactly by Stevens [Ste39], which allows us to estimate pattern shift aware mask yield of a parallel line layout.

Similar to the parallel line case above, we can show that the the mask yield for a regular, square contact array pattern is equivalent to the two-dimensional maximal

[2]Since there is no mask defect distribution data available, we assume uniform spatial distribution of defects as it usually gives more pessimistic yield estimates compared to clustering [JKP11, WB12]

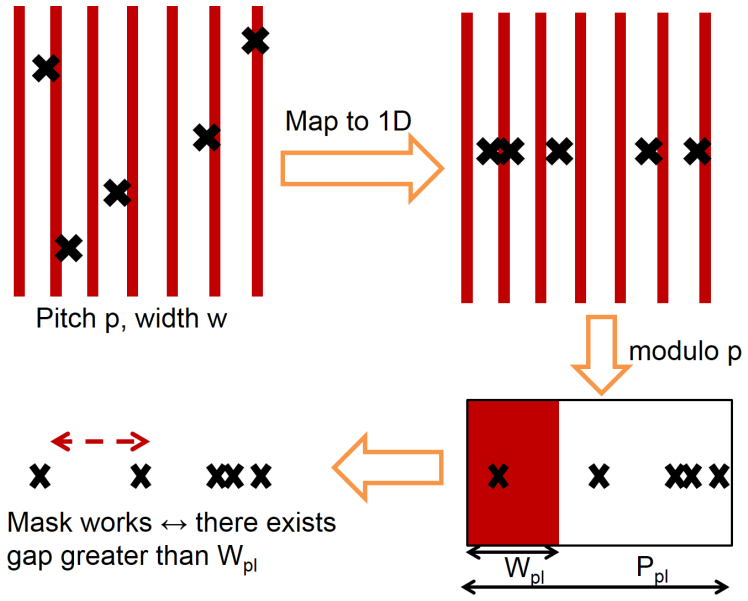


Figure 6.3: Mapping mask yield estimation of parallel line to maximal spacing distribution.

spacing distribution. Janson derived an asymptotic analytical expression for the multi-dimensional maximal spacing problem [Jan87], that holds true as the number of random points (defects, in our case) tend to infinity. Using his expression, we can estimate pattern shift aware mask yield for an infinite contact array layout as shown in Equation 6.4. An additional condition is included for $N_d \leq \frac{2}{D_P}$ to correct for the anomaly that mask yield increases with increase in the number of defects. This analytical expression will be referred to as Janson's formula in the rest of this paper.

$$Y_M^{ct} = \begin{cases} 1 - e^{-N_d^2 D_P e^{-N_d D_P}} & \text{if } N_d \geq \frac{2}{D_P} \\ 1 & \text{otherwise} \end{cases} \tag{6.4}$$

6.4 Critical Density Method

Although the periodicity assumption of parallel line and contact arrays enables us to map the yield estimation problem to a maximal spacing distribution problem, deriving such analytical expressions for random layouts is not straight-forward. In order to address the issue of estimating the yield of realisitic layout patterns, we

propose a two-step model that applies principles from Section 6.3.1 and 6.3.2.

We first define critical density of a layout as follows. *For any given input layout, critical density is the value of D_P such that Equation 6.4 can most accurately predict the actual mask yield of the layout for any number of defects.* Mathematically, we can use the ubiquitous least squares as the criteria for accuracy and thereby define critical density as given in Equation 6.5.

$$D_{crit} = \underset{0 \leq D_P \leq 1}{\operatorname{argmin}} \sum_{N_d = N_d^{min}}^{N_d = N_d^{max}} (Y_M^{ct}(N_d, D_P) - Y_M^{true}(N_d))^2 \tag{6.5}$$

With this definition of critical density, our two-step model first estimates critical density of an input layout using the weighted autocorrelation matrix of the prohibited region as the predictor variables (motivated by the derivation in Section 6.3.1). Equation 6.4 is then used to estimate mask yield.

The use of critical density as a part of a two-step model to estimate mask yield abstracts out defect density thereby providing a defect density-independent metric that depends solely on the prohibited region of a given layout. This metric can be used to compare the robustness of different layouts to EUV mask defects. Additionally, critical density is similar to probability of failure (ratio of critical area to chip area) in conventional CAA.

For a realistic full chip layout, using the entire autocorrelation matrix of a layout as a feature set to predict critical density is infeasible, both from the perspective of computing the autocorrelation, and fitting a model ("curse of dimensionality"). We propose the following set of steps to reduce the dimension of the autocorrelation matrix which is then used to predict the critical density:

- *Limited autocorrelation:* Based on the derivation in Section 6.3.1, only the first $\frac{\Delta_X}{pix} \times \frac{\Delta_Y}{pix}$ entries of the autocorrelation matrix of size $\frac{L_X}{pix} \times \frac{L_Y}{pix}$ need to be considered as a part of the feature set. Moreover, we scale all the entries of the autocorrelation matrix by the reticle area, to make it independent of design size.

- *Compression:* Only the low-frequency Fourier components of the limited autocorrelation matrix are used as features for predicting critical density. This is reasonable since layouts are dominated by lower frequency components due to design rule constraints.

With this reduced autocorrelation based feature set, we apply a simple multivariate linear regression model to predict critical density of any given layout. Since the autocorrelation matrix size depends on the maximum available pattern shift and is scaled by reticle size, it is independent of the layout size. Therefore the linear regression model can be trained using small layout clips, and the trained model can then be applied to large realistic designs. This makes the training of the model manageable.

The operations involved in computing critical density, and then mask yield, of a given random layout are specified in Algorithm 12. *Note that the two-step critical density model does not assume discrete pattern shift solutions.* It actually accounts for the optimal continuous pattern shift since the linear model that estimates critical density is fitted using the Monte Carlo method that uses the optimal continuous pattern shift.

The runtime for estimating critical density is dominated by the polygon Boolean operations to compute the autocorrelation matrix. Each polygon Boolean operation takes $O(\rho A_L \log \rho A_L)$ runtime. With a sampling pixel size of pix, fast fourier transform can be performed in $O(\frac{A_\Delta}{pix^2} \log \frac{A_\Delta}{pix^2})$. Hence, the runtime order complexity to compute the critical density is $O(K \times \frac{A_\Delta}{pix^2} \times \rho A_L \log \rho A_L + \frac{A_\Delta}{pix^2} \log \frac{A_\Delta}{pix^2})$. In constrast, the order complexity of Monte Carlo is $O(N_d \times (\rho A_\Delta \log \rho A_\Delta + (\log \rho A_L)^3)))$ per iteration[3]. This method can also be easily parallelized by computing each entry of the autocorrelation matrix independently.

[3]Number of Monte Carlo iterations also depends on A_L and A_Δ

Algorithm 12 Steps for estimating critical density

Require: Design layout of size $L_X \times L_Y$ and total shift size permitted $\Delta_X \times \Delta_Y$.

Tunable parameters: sample size for autocorrelation pix, and number of fourier order to pick N_F

1: Construct prohibited region of layout PB^s for $s \in s_1, s_2, ...s_K$

2: Define matrix AC of size $\frac{\Delta_X}{pix} \times \frac{\Delta_Y}{pix}$

3: reticleArea $= (L_X + \Delta_X) \times (L_Y + \Delta_Y)$

4: **for all** $X_i \in \{0, p, 2p, ...\Delta_X\}$ **do**

5: **for all** $Y_j \in \{0, p, 2p, ...\Delta_Y\}$ **do**

6: $AC(\frac{X_i}{pix}, \frac{Y_j}{pix}) = \sum_k P(s_k) \frac{Area(PB^{s_k} \cup PB_{ij}^{s_k})}{reticleArea}$

7: **end for**

8: **end for**

9: $ACF = fft(AC)$

10: Pick all terms of ACF with Fourier order less than or equal to N_F

11: Apply fitted linear model to get critical density

6.5 Experimental Results

Both the Monte Carlo method, and our proposed critical density method are implemented in C++. OpenAccess API [oa] is used to read and query layouts. The polygon Boolean operations are performed using Boost Polygon Library [boo]. Fourier transform of the autocorrelation matrix is done using FFTW library [fft], and matrix operations are done using Eigen [Gue10]. OpenMP is used to parallelize both the Monte Carlo Method[4] and the autocorrelation matrix construction step of our critical density method, with eight threads for execution. All our computation has been done on a high performance compute cluster. The reported runtime for the various testcases is the wall time on the compute nodes of the cluster.

The number of Monte Carlo iterations is kept fixed at $20,000$ for all the training clips and test layouts. For all our testcases, the Monte Carlo method is run 15 times, with defect density ranging from 10 defects to 150 defects. All of our reported

[4]Thread-safe random number generation is done here [Hoe07]

average, maximum and average root mean square error (RMSE) values are across this range of defects.

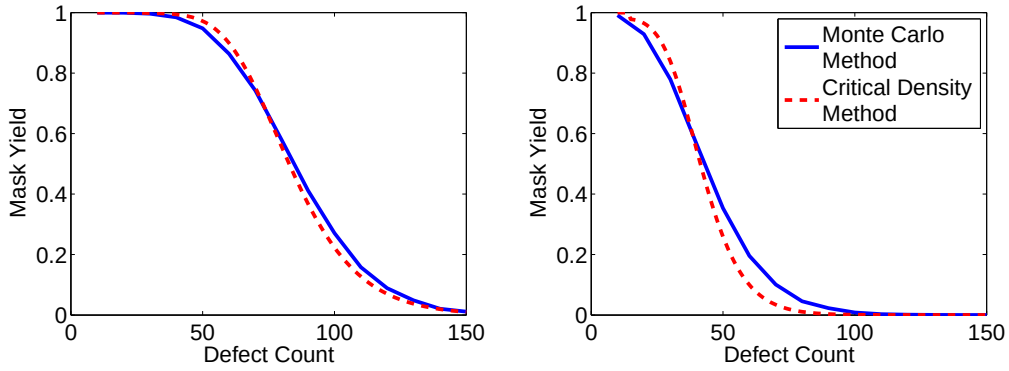
All our analysis in this section is done on designs created using Synopsys $32nm$ standard cell library [syn10], and scaled down to $8nm$. The designs were synthesized, placed and routed using Cadence Encounter [enc08] with 90% cell utilization, unless otherwise stated.

All mask defects are taken as 3D Gaussian-shaped, with three discrete height values $(0.5nm, 1.0nm, 2nm)$ and three discrete full width half maximum values $(25nm, 50nm, 75nm)$. The corresponding nine discrete defect sizes are assigned probability of occurrence inversely proportional to their respective volume. The CD tolerance of all the shapes was set at $0.8nm$.

The available pattern shift $(\Delta_X \times \Delta_Y)$ is taken as $0.5\mu m \times 0.5\mu m$. Current literature on pattern shift suggest that the total available shift is around $200\mu m \times 200\mu m$ [YL12]. A smaller shift value is chosen to demonstrate our methodology since the runtime of the validation Monte Carlo method becomes too slow with large shift area ($\geq 5,000$ hours based on the runtime of [ZD12]). Moreover, the shift area we have chosen is sufficient for comparing different layouts since it is large enough to cover several pitches at $8nm$ node. pix is set to $20nm$ [5], and the size of the autocorrelation based feature vector for each layout is 17. It comprises all the entries from the FFT matrix of the limited autocorrelation matrix with both row and column indices less than 4, and a constant.

The linear regression model to estimate critical density of a layout is trained using $5\mu m \times 5\mu m$ layout clips obtained from a large $8nm$ layout. We used a total of 400 layout clips from each of the four critical layers of a design as the training set: polysilicon, metal 1, contact and active. For these small layout clips, we used the Monte Carlo method to estimate the true mask yield for different number of defects. Using this data, we computed the critical density of the layout clips by solving Equation 6.5 in MATLAB with the interior point method. The autocorrelation

[5] Pixel size only dictates the shift values for which the Boolean AND operations are performed.



(a) Metal 1 layer

(b) Polysilicon layer

Figure 6.4: Mask yield versus defect density for two layers of s1423 design

based features are computed for each clip, and used to train the linear regression model for critical density in MATLAB.

6.5.1 Model Validation

The trained linear regression model is then applied on different layers of four benchmark layouts from ISCAS'89 [iscb], and one RISC processor layout. All these layouts are different from the training clips used to fit the model. The results are summarized in Table 6.2 [6].

Compared to the rigorous Monte Carlo method, our critical density method is able to predict critical density fairly accurately for all the test layouts with a runtime improvement ranging from 300× to 1300× The average RMSE in estimating mask yield ranges from $0, .08\% - 6.44\%$. Moreover, the method is able to track the general trend of how mask yield changes with defect density fairly accurately. This is illustrated in Figure 6.4, which plots the mask yield versus defect density of the Monte Carlo and the critical density method for the polysilicon and metal 1 layers of $s1423$.

[6]The validation Monte Carlo method is too slow to be run for the larger processor layout, hence the missing entries in the table.

Table 6.2: Validation of critical density method. Mask yield RMSE is avg.RMSE between the mask yield estimate of Monte Carlo and proposed method across the defect range. Runtime is total wall time required to compute mask yield for the defect range.

Design	Layer	Number of shape edges	Layout density	Monte Carlo method Runtime (sec.)	Critical density method		
					Critical density	Mask yield RMSE	Runtime (sec.)
s349-syn32nm Utilization 90%	POLY	4084	0.16	10742	0.14	4.15%	19
	ACT	2384	0.32	14725	0.04	4.67%	16
	CO	20132	0.03	57432	0.03	0.32%	62
	M1	9432	0.22	48382	0.08	2.84%	84
s1423-syn32nm Utilization 90%	POLY	20672	0.16	66651	0.14	4.24%	86
	ACT	11348	0.33	24895	0.04	4.54%	79
	CO	101856	0.03	239572	0.03	0.50%	349
	M1	47554	0.22	261816	0.08	1.98%	465
s1196-syn32nm Utilization 90%	POLY	11788	0.16	28201	0.14	4.28%	47
	ACT	5800	0.29	23522	0.03	3.40%	39
	CO	60448	0.03	253287	0.03	0.54%	192
	M1	26128	0.19	198613	0.07	2.91%	222
s1196-syn32nm-u70 Utilization 70%	POLY	11788	0.12	34597	0.10	6.44%	47
	ACT	6176	0.23	12797	0.03	0.94%	41
	CO	62336	0.03	255483	0.03	0.08%	233
	M1	26502	0.16	208513	0.06	4.15%	227
Cortex M0 Utilization 90%	POLY	333748	0.15	NA	0.13	NA	858
	ACT	178396	0.29	NA	0.03	NA	999
	CO	1746968	0.03	NA	0.03	NA	4743
	M1	767380	0.19	NA	0.07	NA	5852

6.5.2 Impact of Layout Density and Regularity

The critical density of a layout is strongly influenced by the layout density. This is confirmed by comparing two version of s1196 in Table 6.2 with cell utilization of 90% (default) and 70%. Reducing the cell utilization reduces the layout density of all the layers, thereby reducing the critical density as well. Moreover, note that different designs constructed with the same utilization have almost equal critical density for each respective layer. This suggests that critical density depends on global layout characteristics instead of local hotspot-like regions. Hence, improving this metric may necessitate changes in design rules or physical design techniques.

Although layout density plays an important role in determining critical density, it is not the only factor that affects critical density. Polysilicon layer (which is a fixed pitch regular grating) has higher critical density compared to irregular metal 1 layer of each design, despite lower layout density. This indicates that random layout patterns are better suited to exploit the benefits of pattern shift due to better autocorrelation properties. To further highlight this impact of regularity, we constructed two regular layouts, parallel line and contact array, which have the same layout density as the polysilicon layer of $s1423$ and metal 1 layer of $s1196 - u70$. The results, shown in Figure 6.5, highlight two key aspects of layouts that affect mask yield:

- 1D layout topology (parallel line and polysilicon), are significantly worse than 2D topology (contact array and metal 1) because 2D layouts can benefit from pattern shift in both X and Y directions, whereas 1D layouts benefit only in the direction perpendicular to the parallel lines.
- An irregular 2D layout like metal 1 is much better suited to derive the benefit of pattern shift compared to a regular 2D contact array.

Most manufacturing processes, especially lithography, favor 1D regular layouts. This has led to increasing layout regularity with each technology node. But our results show that the reduced mask yield of such regular layouts can significantly increase

mask cost for EUV lithography. Hence, a systematic co-optimization to balance these two competing requirements may be required for EUV layouts.

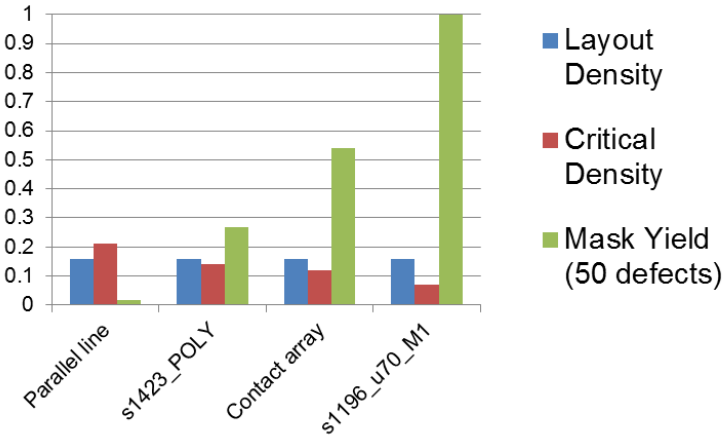


Figure 6.5: Illustration of impact of regularity on critical density (and consequently, mask yield) for layouts with same density.

6.6 Conclusions

In this work, we proposed a new metric for evaluating the robustness of EUV layouts with respect to mask defects, critical density. Using critical density, layout designers and mask makers can quickly estimate the probability of finding a defective EUV mask blank on which the layout can be safely patterned for any defect density(mask yield). Our method accounts for the impact of pattern shift based defect avoidance technique on mask yield, which is the most challenging part of this methodology. We first solved the problem assuming discrete pattern shift solutions using inclusion-exclusion. Then we mapped the problem to a classic geometric probability problem, maximal spacings, for the limited case of parallel line and contact array patterns. Using principles from both these approaches, we defined our critical density metric and proposed a novel method to estimate critical density, which can then be used to estimate the pattern shift aware mask yield for any arbitrary layout. Our method was shown to be $300-1300\times$ faster than the rigorous Monte Carlo method for estimating mask yield and was able to predict mask yield with $0.08\% - 6.44\%$ average RMSE

across a range of defect density for four critical layers (polysilicon, active, contact and metal1).

The methodology for estimating critical density shows that mask yield is a strong function of the autocorrelation of the layout. Our analysis indicates that irregular 2D layouts have better mask yield for the same layout density, which is contrary to most manufacturing processes that demand layout regularity. By using dummy features to make irregular layouts regular with respect to printability, this problem can be addressed since defects on dummy features do not matter.

Fast estimation of critical density enabled by our method can allow us to develop techniques to improve the mask yield of EUV layouts. Our preliminary experiments to improve critical density by iteratively adjusting the whitespace after placement suggests that $3\% - 7\%$ improvement in mask yield is possible without any area penalty. Since we used a random search based whitespace optimization for each row independently. the method is too slow to be practically useful. In the future, we plan to develop a scalable layout optimization methodology to improve the robustness of EUV layouts. Our future work also includes further enhancing the critical density model to account for other defect avoidance strategies such as pattern rotation or floorplanning, and dealing with non-uniform spatial distribution of defects (if defect data necessitate so).

CHAPTER 7

Conclusions and Future Work

Our current approach for criticality assignment sets a single CD tolerance value for each layout shape. In the case of front-end layer reticles, polysilicon and active, this was done by using the timing slack. This would typically require the designers to pass all the timing paths of the design to the foundry, along with the layout. Considering the large size of today's designs, this might not be feasible. We plan to look at more efficient representations of this timing data. One approach we are planning to look at involves partitioning the layout such that all layout shapes in a partition have the same CD tolerance. Instead of assigning a fixed CD tolerance to each partition, these CD tolerances could be bound by some linear constraints. Interestingly, timing constraints already provide one such representation, if we treat each cell as a separate partition. Hence, solving this problem reduces to essentially clustering groups of neighboring cells to the same CD tolerance so the number of CD tolerance variables, and the linear constraints that bind them, are reduced.

References

[Ach09] Tobias Achterberg. "SCIP: Solving constraint integer programs." *Mathematical Programming Computation*, **1**(1):1–41, 2009. `http://mpc.zib.de/index.php/MPC/article/view/4`.

[Aro03] Sanjeev Arora. "Approximation schemes for NP-hard geometric optimization problems: a survey." *Mathematical Programming*, **97**(1-2):43–69, July 2003. Cited by 0091.

[ASH05] Valery Axelrad, Andrei Shibkov, Gene Hill, Hung-Jen Lin, Cyrus Tabery, Dan White, Victor Boksha, and Randy Thilmany. "A novel design-process optimization technique based on self-consistent electrical performance evaluation." volume 5756, pp. 419–426. SPIE, 2005.

[BA10] John Burns and Mansoor Abbas. "EUV mask defect mitigation through pattern placement." Proc. SPIE/BACUS, 2010.

[BAB05] F. Boeuf, F. Arnaud, C. Boccaccio, F. Salvetti, J. Todeschini, L. Pain, M. Jurdit, S. Manakli, B. Icard, N. Planes, N. Gierczynski, S. Denorme, B. Borot, C. Ortolland, B. Duriez, B. Tavel, P. Gouraud, M. Broekaart, V. Dejonghe, P. Brun, F. Guyader, P. Morini, C. Reddy, M. Aminpur, C. Laviron, S. Smith, J.P. Jacquemin, M. Mellier, F. Andre, N. Bicais-Lepinay, S. Jullian, J. Bustos, and T. Skotnicki. "$0.248\mu m^2$ and $0.334\mu m^2$ conventional bulk 6T-SRAM bit-cells for 45nm node low cost - general purpose applications." In *VLSI Technology, 2005. Digest of Technical Papers. 2005 Symposium on*, pp. 130 – 131, june 2005.

[BE] Benno Bueler and Andreas Enge. "VINCI version 1.0.5." `http://www.math.u-bordeaux1.fr/~aenge/software/vinci/manual.pdf`.

[BEL08a] S. Banerjee, P. Elakkumanan, L. Liebmann, and M. Orshansky. "Electrically Driven Optical Proximity Correction Based on Linear Programming." In *Proc. ICCAD*, 2008.

[BEL08b] Shayak Banerjee, Praveen Elakkumanan, Lars W. Liebmann, and Michael Orshansky. "Electrically driven optical proximity correction based on linear programming." In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '08, pp. 473–479, Piscataway, NJ, USA, 2008. IEEE Press.

[Ben75] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching." *Commun. ACM*, **18**:509–517, September 1975.

[boo] "Boost Polygon Library." `http://www.boost.org/doc/libs/1_52_0/libs/polygon/doc/index.htm`.

[Bra65] R.N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill Ltd., 1965.

[cal08] "Mentor Calibre." `http://www.mentor.com/`, 2008.

[CCN10] Chris H. Clifford, Tina T. Chan, and Andrew R. Neureuther. "Compensation methods for buried defects in extreme ultraviolet lithography masks." volume 7636, p. 763623. SPIE, 2010.

[CG03] Nicolas B. Cobb and Yuri Granik. "Using OPC to optimize for image slope and improve process window." volume 5130, pp. 838–846. SPIE, 2003.

[CG10] Tuck-Boon Chan and P. Gupta. "On Electrical Modeling of Imperfect Diffusion Patterning." In *VLSI Design, 2010. VLSID '10. 23rd International Conference on*, pp. 224 –229, jan. 2010.

[CGG10a] Tuck-Boon Chan, R.S. Ghaida, and P. Gupta. "Electrical Modeling of Lithographic Imperfections." In *VLSI Design, 2010. VLSID '10. 23rd International Conference on*, pp. 423 –428, jan. 2010.

[CGG10b] Tuck-Boon Chan, R.S. Ghaida, and P. Gupta. "Electrical Modeling of Lithographic Imperfections." In *VLSI Design*, 2010.

[Chu11] *Optimization of mask shot count using MB-MDP and lithography simulation*, volume 8166, 2011.

[Chv79] V. Chvatal. "A Greedy Heuristic for the Set-Covering Problem." *Mathematics of Operations Research*, **4**(3):233–235, 1979.

[CJH06] Soo-Han Choi, A-Young Je, Ji-Suk Hong, Moon-Hyun Yoo, and Jeong-Taek Kong. "MEEF-based correction to achieve OPC convergence of low-k1 lithography with strong OAI." volume 6154, p. 61540P. SPIE, 2006.

[CL03] Shih-Ying Chen and Eric C. Lynn. "Effective placement of chips on a shuttle mask." volume 5130, pp. 681–688. Proc. SPIE, 2003.

[Cli10] Chris Heinz Clifford. *Simulation and Compensation Methods for EUV Lithography Masks with Buried Defects*. PhD thesis, EECS Department, University of California, Berkeley, 2010.

[CLR01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

[CN08] Chris H. Clifford and Andrew R. Neureuther. "Smoothing based model for images of buried EUV multilayer defects." volume 7122, p. 71221X. Proc. SPIE Photomask, 2008.

[CN09] Chris H. Clifford and Andrew R. Neureuther. "Fast simulation methods and modeling for extreme ultraviolet masks with buried defects." *SPIE JM3*, **8**(3):031402, 2009.

[CR88] J.C. Culberson and R.A. Reckhow. "Covering polygons is hard." In *, 29th Annual Symposium on Foundations of Computer Science, 1988*, pp. 601–611, October 1988.

[CRX03] Jason Cong, Michail Romesis, and Min Xie. "Optimality and Stability Study of Timing-Driven Placement Algorithms." In *Proceedings of the 2003 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '03, pp. 472–, Washington, DC, USA, 2003. IEEE Computer Society.

[DGS08] Frank A. J. M. Driessen, Jamila Gunawerdana, Yakuko Saito, Hideo Tsuchiya, and Yoshitake Tsuji. "Flexible sensitivity inspection with TK-CMI software for criticality-awareness." volume 7122, p. 712222. Proc. SPIE Photomask, 2008.

[DL02] Y. Deng, B. L.-Fontaine, et al. "Performance of repaired defects and attPSM in EUV multilayer masks." Proc. SPIE/BACUS, 2002.

[DMI08] Aditya Dayal, Bo Mu, Venkat Iyer, Phillip Lim, Arosha Goonesekera, and Bill Broadbent. "Results from the KLA-Tencor TeraScanXR reticle inspection tool." volume 7122, p. 71223G. Proc. SPIE Photomask, 2008.

[DZ11] Y. Du, H. Zhang, et al. "EUV mask preparation considering blank defects mitigation." Proc. SPIE/BACUS, 2011.

[DZB06] Arndt C. Dürr, Axel M. Zibold, and Klaus Böhm. "An advanced study for defect disposition through 193-nm aerial imaging." volume 6152, p. 61522M. Proc. SPIE, 2006.

[DZW12a] Yuelin Du, Hongbo Zhang, and Martin D. F. Wong. "Linear time EUV blank defect mitigation algorithm considering tolerance to inspection inaccuracy." Proc. SPIE Photomask, 2012.

[DZW12b] Yuelin Du, Hongbo Zhang, Martin D. F. Wong, et al. "Efficient multi-die placement for blank defect mitigation in EUV lithography." 2012.

[enc08] "Cadence SOC Encounter." `http://www.cadence.com/`, 2008.

[ETS12] Ahmad Elayat, Peter Thwaite, and Steffen Schulze. "EUV mask-blank defect avoidance solutions assessment." Proc. SPIE Photomask, 2012.

[fft] "FFTW." `http://www.fftw.org/`.

[FK84] Deborah S. Franzblau and Daniel J. Kleitman. "An Algorithm for Constructing Regions with Rectangles: Independence and Minimum Generating Sets for Collections of Intervals." In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pp. 167–174, New York, NY, USA, 1984. ACM.

[FKK10] Aki Fujimura, David Kim, Tadashi Komagata, Yasutoshi Nakagawa, Vikram Tolani, and Tom Cecil. "Best depth of focus on 22-nm logic wafers with less shot count." volume 7748, pp. 77480V–77480V–9, 2010.

[Fra89] D. S. Franzblau. "Performance Guarantees on a Sweep-Line Heuristic for Covering Rectilinear Polygons with Rectangles." *SIAM Journal on Discrete Mathematics*, **2**(3):307–321, August 1989. Cited by 0038.

[fre09] "FreePDK 45nm Design Rules." `http://www.eda.ncsu.edu/wiki/FreePDK45:Contents/`, 2009.

[GG83] K. Gourley and D. Green. "A Polygon-to-Rectangle Conversion Algorithm." *IEEE Computer Graphics and Applications*, 1983.

[GKS03] P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. "A Cost-Driven Lithographic Correction Methodology Based on Off-the-shelf Sizing Tools." In *Proc. DAC*, 2003.

[GKS07a] Puneet Gupta, Andrew B. Kahng, Dennis Sylvester, and Jie Yang. "Performance-driven optical proximity correction for mask cost reduction." *Journal of Micro/Nanolithography, MEMS and MOEMS*, **6**(3):031005, 2007.

[GKS07b] Puneet Gupta, Andrew B. Kahng, Dennis Sylvester, and Jie Yang. "Performance-Driven Optical Proximity Correction for Mask Cost Reduction." *SPIE JM3*, 2007.

[GMM09] Justin Ghan, Ning Ma, Sandipan Mishra, Costas Spanos, Kameshwar Poolla, Norma Rodriguez, and Luigi Capodieci. "Clustering and pattern matching for an automatic hotspot classification and detection system." volume 7275, p. 727516. SPIE, 2009.

[GP10] Puneet Gupta and E. Papadopoulou. "Yield Analysis and Optimization." In *The Handbook of Algorithms for VLSI Physical Design Automation*. CRC Press, 2010.

[Gue10] Gaël Guennebaud et al. "Eigen v3." http://eigen.tuxfamily.org, 2010.

[HK08] Kunihiro Hosono and Kokoro Kato. "PMJ panel discussion overview on mask complexities, cost, and cycle time in 32-nm system LSI generation: conflict or concurrent?" volume 7122, p. 712207. Proc. SPIE Photomask, 2008.

[HL07] Laura Heinrich-Litan and Marco E. Lbbecke. "Rectangle covers revisited computationally." *J. Exp. Algorithmics*, **11**, February 2007. Cited by 0005.

[HLE08] Shad Hedges, Chin Le, Mark Eickhoff, Mark Wylie, Tim Simmons, Venu Vellanki, and Jeff McMurran. "Novel mask inspection flow using Sensitivity Control Layers (SCL) on the TeraScanHR-587 platform." volume 7122, p. 71221G. Proc. SPIE Photomask, 2008.

[Hoe07] Mark Hoemmen. "Generating random numbers in parallel." 2007.

[Hsp] "Synopsys HSPICE." http://www.synopsys.com/.

[ISCa] "Iscas-85 Benchmark Circuits Verilog Files." http://www.pld.ttu.ee/~maksim/benchmarks/iscas85/verilog/.

[iscb] "Iscas-89 Benchmark Circuits Verilog Files." http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/verilog/.

[itr09] "International Technology Roadmap for Semiconductors(ITRS)." http://www.itrs.net/, 2009.

[Jan87] Svante Janson. "Maximal Spacings in Several Dimensions." *The Annals of Probability*, 1987.

[JH10] R. Jonckheere, D. Heuvel, et al. "EUV Mask Defectivity Status and Mitigation Towards HVM." International Symp. on EUL, 2010.

[JH11] R. Jonckheere, D. V. Heuvel, et al. "Evidence of printing blank-related defects on EUV masks missed by blank inspection." 2011.

[JKP11] Kwangok Jeong, Andrew B Kahng, and Christopher J Progler. "Cost-driven mask strategies considering parametric yield, defectivity, and production volume." *SPIE JM3*, 2011.

[JZ11] Shangliang Jiang and Avideh Zakhor. "Application of signal reconstruction techniques to shot count reduction in simulation driven fracturing." pp. 81660U–81660U–14, 2011.

[JZ14] Shangliang Jiang and Avideh Zakhor. "Shot overlap model-based fracturing for edge-based OPC layouts." volume 9052, pp. 90520L–90520L–19, 2014.

[KAC01] Linard Karklin, Mark M. Altamirano, Lynn Cai, Khoi A. Phan, and Chris A. Spence. "Automatic defect severity scoring for 193-nm reticle defect inspection." volume 4346, pp. 898–906. Proc. SPIE, 2001.

[Kah07] Mandoiu I. I. Xu X. Zelikovsky A. Z. Kahng, A. B. "Enhanced Design Flow and Optimizations for Multiproject Wafers." *IEEE TCAD*, **26**(2):301 –311, Feb. 2007.

[KCG06] Azalia Krasnoperova, James A. Culp, Ioana Graur, Scott Mansfield, Mohamed Al-Imam, and Hesham Maaty. "Process window OPC for reduced process variability and enhanced yield." volume 6154, p. 61543L. SPIE, 2006.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing." *Science, Number 4598, 13 May 1983*, **220, 4598**:671–680, 1983.

[KLS96] Jeff Kahn, Nathan Linial, and Alex Samorodnitsky. "Inclusion-exclusion: Exact and approximate." *Combinatorica*, 1996.

[KLZ08] Tung-Yaw Kang, Hsin-Chang Lee, H. Zhang, K. Yamada, Y. Kitayama, K. Kobayashi, and Peter Fiekowsky. "Auto-classification and simulation of mask defects using SEM and CAD images." volume 7122, p. 71221F. Proc. SPIE Photomask, 2008.

[KNO07] Kaoru Koike, Kohichi Nakayama, Kazuhisa Ogawa, and Hidetoshi Ohnuma. "OPC to reduce variability of transistor properties." volume 6521, p. 65210J. SPIE, 2007.

[KSJ06] In-Sung Kim, Sungsoo Suh, Sunggon Jung, Eunmi Lee, Young-Seog Kang, Sukjoo Lee, Sang-Gyun Woo, and HanKu Cho. "Toward DFM: process worthy design and OPC through verification method using MEEF, TF-MEEF, and MTT." volume 6156, p. 61560L. SPIE, 2006.

[LD97] S. Levasseur and F. Duvivier. "Application of a Yield Model Merging Critical Areas and Defectivity to Industrial Products." In *Proc. IEEE Defect and Fault Tolerance in VLSI Systems*, 1997.

[Lev09] Harry J. Levinson. "Extreme ultraviolet lithography's path to manufacturing." *Journal of Micro/Nanolithography, MEMS and MOEMS*, **8**(4):041501, 2009.

[LMH06] Lars Liebmann, Scott Mansfield, Geng Han, James Culp, Jason Hibbeler, and Roger Tsai. "Reducing DfM to practice: the lithography manufacturability assessor." volume 6156, p. 61560K. SPIE, 2006.

[Lov98] Laszlo Lovasz. "Hit-and-Run Mixes Fast." *Math. Prog*, 1998.

[MBM08] Hideyuki Moribe, Takeshi Bashomatsu, Kenichi Matsumura, Akira Uehara, and Hiroyuki Takahashi. "Improvement of image quality and inspection speed in LM7500 reticle inspection system." volume 7028, p. 70282K. Proc. SPIE Photomask, 2008.

[MG13] Gregory McIntyre, Emily Gallagher, et al. "Through-focus EUV multilayer defect repair with nanomachining." Proc. SPIE, 2013.

[MJL99] Chris A. Mack, Sven Jug, and Dale A. Legband. "Data analysis for photolithography." volume 3677, pp. 415–434. SPIE, 1999.

[Nak05] Junichi Nakamura. *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Inc., 2005.

[nan] "Nangate Open Cell Library." `https://www.nangate.com/`.

[Nau11] Patrick Naulleau. "EUV Systems and Patterning." 2011.

[nba] "Volume of an n-ball." `http://en.wikipedia.org/wiki/Volume_of_an_n-ball`.

[oa] "OpenAccess API." http://www.si2.org/.

[ope] "OpenCores." http://www.opencores.org.

[Pav86] J. M. Pavkovich. "Proximity effect correction calculations by the integral equation approximate solution method." *Journal of Vacuum Science Technology B: Microelectronics and Nanometer Structures*, **4**(1):159–163, January 1986.

[PLC03] Linyong Pang, Alex Lu, Jacky Chen, Eric Guo, Lynn Cai, and Jiunn-Hung Chen. "Enhanced dispositioning of reticle defects for advanced masks using virtual stepper with automated defect severity scoring." volume 5256, pp. 461–473. Proc. SPIE Photomask, 2003.

[Pyk65] R. Pyke. "Spacings." *Journal of the Royal Statistical Society. Series B (Methodological)*, 1965.

[RBF02] Alan E. Rosenbluth, Scott Bukofsky, Carlos Fonseca, Michael Hibbs, Kafai Lai, Antoinette F. Molless, Rama N. Singh, and Alfred K. K. Wong. "Optimum mask and source patterns to print a given shape." *Journal of Microlithography, Microfabrication, and Microsystems*, **1**(1):13–30, 2002.

[RG09] Dominic Reinhard and Puneet Gupta. "On comparing conventional and electrically driven OPC techniques." volume 7488, p. 748838. SPIE, 2009.

[sem14] "SEMATECH Achieves Breakthrough Defect Reductions in EUV Mask Blanks." http://public.sematech.org/pages/newsrelease.aspx?NewsID=2, 2014.

[SRM07] Dvori Stoler, Wayne Ruch, Weimin Ma, Swapnajit Chakravarty, Steven Liu, Ray Morgan, John Valadez, Bill Moore, and John Burns. "Optimizing defect inspection strategy through the use of design-aware database control layers." volume 6730, p. 67303L. Proc. SPIE Photomask, 2007.

[Ste39] W. L. Stevens. "Solution to a Geometrical Problem in Probability." *Annals of Eugenics*, **9**(4):315–320, 1939.

[SW80] H.W. Six and D. Wood. "The Rectangle Intersection Problem Revisited." *BIT Numerical Mathematics*, 1980.

[syn10] "Synopsys 32nm Library.", 2010.

[THT08] S. Teh, C. Heng, and A. Tay. "Design-Process Integration for Performance-Based OPC Framework." In *Proc. DAC*, 2008.

[TTN08] Hideo Tsuchiya, Masakazu Tokita, Takehiko Nomura, and Tadao Inoue. "Die-to-database mask inspection with variable sensitivity." volume 7028, p. 70282I. Proc. SPIE Photomask, 2008.

[TYT10] Tsuneo Terasawa, Takeshi Yamane, Toshihiko Tanaka, Osamu Suga, Takashi Kamo, and Ichiro Mori. "Actinic phase defect detection and printability analysis for patterned EUVL mask." volume 7636, p. 763602. SPIE, 2010.

[VHR03] William W. Volk, Carl Hess, Wayne Ruch, Zongchang Yu, Weimin Ma, Lisa Fisher, Carl Vickery, and Z. Mark Ma. "Investigation of smart inspection of critical layer reticles using additional designer data to determine defect significance." volume 5256, pp. 489–499. Proc. SPIE Photomask, 2003.

[WB12] Alfred Wagner, Martin Burkhardt, et al. "Mitigation of extreme ultraviolet mask defects by pattern shifting: Method and statistics." *J. Vac. Sci. & Technol., B*, 2012.

[XCP08] Guangming Xiao, Tom Cecil, Lingyong Pang, Bob Gleason, and John McCarty. "Source optimization and mask design to minimize MEEF in low k1 lithography." volume 7028, p. 70280T. SPIE, 2008.

[YGP13] Bei Yu, Jhih-Rong Gao, and D.Z. Pan. "L-shape based layout fracturing for e-beam lithography." In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pp. 249–254, January 2013.

[YL12] P.-Y. Yan, Y. Liu, et al. "EUVL multilayer mask blank defect mitigation for defect-free EUVL mask fabrication." 2012.

[YLK09] Ellyn Yang, Cheng He Li, Xiao Hui Kang, and Eric Guo. "Model-based retarget for 45nm node and beyond." volume 7274, p. 727428. SPIE, 2009.

[ZA07a] Qiaolin Charlie Zhang and Paul van Adrichem. "Determining OPC target specifications electrically instead of geometrically." volume 6730, p. 67303V. SPIE, 2007.

[ZA07b] Qiaolin Charlie Zhang and Paul van Adrichem. "Determining OPC target specifications electrically instead of geometrically." volume 6730, p. 67303V. SPIE, 2007.

[ZD12] H. Zhang, Y. Du, et al. "Efficient pattern relocation for EUV blank defect mitigation." In *Proc. ASP-DAC*, 2012.

[ZDW12] Hongbo Zhang, Yuelin Du, Martin D. F. Wong, et al. "Layout small-angle rotation and shift for EUV defect mitigation." In *Proc. ICCAD*, 2012.