Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing

by

Nicolas Bailey Cobb

B.S. (University of Colorado at Boulder) 1991
M.S. (University of California at Berkeley) 1994

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

in

Engineering: Electrical Engineering and Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Avideh Zakhor, Chair
Professor A.R. Neureuther
Professor Kameshwar Poolla

Spring 1998

Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing

Abstract

Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing

by

Nicolas Bailey Cobb
Doctor of Philosophy in Engineering: Electrical Engineering and Computer Science
University of California at Berkeley
Professor Avideh Zakhor, Chair

Optical Proximity Correction (OPC) is used in lithography to increase the achievable resolution and pattern transfer fidelity for IC manufacturing. The fundamental idea behind OPC is to modify the mask itself in order to correct for non-idealities that occur during pattern transfer. OPC has been used in IC manufacturing in some shape or form for many years. In the past and even currently, hand modifications to small sections of the layout were made to improve the aerial image. However the slow speed of existing aerial image simulators prohibits this method from being applied to an entire chip. To circumvent this problem, one approach has been to apply "rule-based techniques" to correct an entire chip. While rule-based schemes are fast and therefore can be applied to a whole layout, they are not as accurate as desired because the corrections are not directly based on simulation.

In this thesis, we first look at the OPC problem and define the goals, constraints, and techniques available. Then, a practical and general OPC framework is built up using concepts from linear systems, control theory, and computational geometry. A simulation-based, or model-based, OPC algorithm is developed which simulates the optics and processing steps of lithography for millions of locations. The key contributions to the OPC field made in this thesis work include: (1) formulation of OPC as a feedback control problem using an iterative solution, (2) an algorithm for edge movement during OPC with cost function criteria, (3) use of fast aerial image simulation for OPC, which truly enables full chip model-based OPC, and (4) the variable threshold resist (VTR) model for simplified prediction of CD based off aerial image.

A major contribution of this thesis is the development of a fast aerial image simulator which is tailored to the problem of OPC. In OPC applications, it is best to compute intensity at sparse points. Therefore, our fast aerial image simulator is tailored to computing intensity at sparse points, rather than on a regular dense grid. The starting point for the fast simulation is an established decomposition of the Hopkins partially coherent imaging equations, originally proposed by Gamo[14]. Within this thesis, the decomposition is called the Sum of Coherent Systems (SOCS) structure. The numerical implementation of this decomposition using Singular Value Decomposition (SVD) is described in detail. Since each of the coherent systems is a linear shift invariant (LSI) system, convolution can be used to compute their outputs. Since the input to these systems is a function consisting of polygons with a limited number of tranmission values, e.g. binary masks, the aerial image calculations can be further sped up. A lookup technique for general convolution of mask functions with arbitrary convolution kernels is outlined. The lookup tables require storage of $\mathcal{O}(N \times N)$ complex numbers for each $N \times N$ convolution kernel in the SOCS. When the lookup technique is combined with the SOCS structure, a highly efficient aerial image computation technique emerges. Using this technique, aerial image intensity calculation speeds of 6.3 msec/pt can be achieved for sparsely chosen simulation points.

Another contribution of this thesis is the development of a variable threshold resist model (VTR). Traditionally, the normalized 0.3 intensity contours are used as a way to predict the edge placement of 3-D structures on wafers. While the constant threshold model is fast, it is only approximate. On the other hand, accurate resist and etch simulation models are typically too slow for whole chip corrections. The basic premise of the VTR model is to vary the 0.3 threshold as a function of the aerial image in order to accurately predict edge placement on the wafer. The model uses the aerial image peak intensity and image slope along a cutline to deduce the development point of the resist, and has two primary benefits: (1) it is fast (2) it can be fit to empirical data. The VTR can be fit to empirical data with observed accuracy of less than 10 nm error which make it good for accurate OPC.

We combine the fast aerial image simulator and the VTR model in an iterative feedback loop to formulate OPC as a feedback control problem. A cost function controls the movement of edges by way of a greedy optimization algorithm. The simulation system and OPC algorithm are separated to allow the OPC algorithm to be independent of the simulation models. Therefore, as new models for lithography are developed, they can be

inserted into an existing OPC framework.

Simulation and experimental results are presented which demonstrate potential benefits of OPC. Results indicate that OPC can decrease critical dimension (CD) errors, increase CD uniformity, move process windows, and challenge the traditional bounds of optical lithography.

Professor Avideh Zakhor
Dissertation Committee Chair

Contents

List of Figures

List of Tables

Acknowledgements

I want to thank my advisor, Professor Avideh Zakhor for her great ideas, discussions, encouragement and enthusiasm throughout my research at Berkeley. I was also fortunate to have Professor Andy Neureuther as almost a "second advisor" to whom am I indebted for many insightful discussions and ideas. Also, thanks to the other members of my quals committee: Professor Kameshwar Poolla, whose thought-provoking ideas in control theory and system identification made an noticeable impact on my thesis work, and Professor Jeffrey Bokor whose advanced optics class made a lasting impression. Also, thanks to Professor Oldham whose involvement in lithography research at UCB has been a benefit to my research.

I would like to express my thanks to the TCAD and image simulation (SPLAT) research group at Berkeley, especially David Newmark, Bob Socha, and Derek Lee for input and discussions. The existence of SPLAT has been a great benefit to this research. Much of the research highlighted in this work is a result of the LIDAS (LIthography Design and Analysis System) project at Berkeley in which the LIDAS OPC software was built. The erstwhile LIDAS project members are Eugene Miloslavsky who provided invaluable assitance and numerous great ideas early-on, and Dong-Wu Zhao and Henry Lee who provided critical contributions at the right times. Also, it has been a great pleasure working with the members of the Video and Image Process (VIP) Group, including Nelson Chang, whom I thank for frequent consultations, Ralph Neff who provided timely insights, Yong Liu whose work in mask design was critical to my early research, Nimish Shah, Emile Sahouria, Ron Galicia and honorary members Grace Chang and Claire Tomlin.

Several industrial mentors and contacts have helped the progression of this research immensely. I'd like to thank Chris Spence at AMD for feedback and guidance. From SEMATECH, I'd like to thank Frank Schellenberg, Jim Morrow, and Hua Zhang for experimental results. Also, thanks to Hua-yu Liu, Mehran Reihani, and Farvardin Jahansooz from HP for experimental results on OPC. I have learned a great deal from the UC Berkeley industrial partners. This research was sponsored by SEMATECH and SRC Contract No. 94-MC-500.

Chapter 1

Introduction

Optical microlithography is the key technology used in VLSI circuit fabrication. Its success can be attributed to the ease of transferring layout patterns to silicon by optical projection printing, its high throughput, and its high yield. Over the past fifteen years, improvements to lithography have brought device sizes down from above 1 μm to below 0.25 μm design rules [45]. The driving force behind the miniaturization effort is the desire for faster and smaller circuits. If the progress in lithography can be sustained, device sizes could go down to 0.10 μm and below.

Before reaching these dimensions, however, we reach the limits of lithography and pattern transfer. At small dimensions, loss of image quality in optical lithography erodes design-to-wafer fidelity on silicon. Yet, equipment costs, established production knowledge, and a tight production schedule act as conservative forces on the side of keeping optical lithography viable in this range. The present outlook is that optical lithography will remain an essential part of pattern transfer for both economic and technology reasons well into the 21st century.

To further the lifetime of optical lithography, integrated circuit (IC) manufacturers are seeking ways of enhancing resolution. Traditionally, the way to increase resolution, and therefore shrink dimensions, is by using smaller wavelengths and better optics. However, the speed at which technology is developing at smaller wavelengths is slower than the speed at which designs are shrinking.

An alternative strategy to smaller wavelengths and improved illumination is to increase resolution and/or compensate for pattern transfer non-idealities at the mask level—mask engineering. The approach of making systematic modifications to mask geometries to compensate for pattern transfer non-idealities is typically called Optical Proximity Correction (OPC). In this work we formalize and propose a systematic approach to OPC and present simulation models, and processing techniques that allow OPC to become a part of the IC design flow.

1.1 Optical Proximity Correction

The advanced mask engineering technique called OPC can be used to increase layout-to-wafer pattern fidelity. The goal of the OPC is to *enhance* optical characteristics by making adjustments to the mask. This is accomplished by compensating mask geometry for known effects which will occur during imaging or subsequent processing. To reiterate this more formally, as our problem statement:

Problem Statement: Given a desired geometric pattern on the wafer, find a mask design such that the final pattern remaining after the complete lithography process is as close as possible to the desired pattern.

Conceptually, from a systems viewpoint OPC can be imagined as an "inverse problem", as depicted in Figure 1.1, where the "OPC" block can be viewed as an approximate inverse to the "litho" block. Changes to the mask can be viewed as "pre-compensation" which will result in a wafer image that matches what is desired.

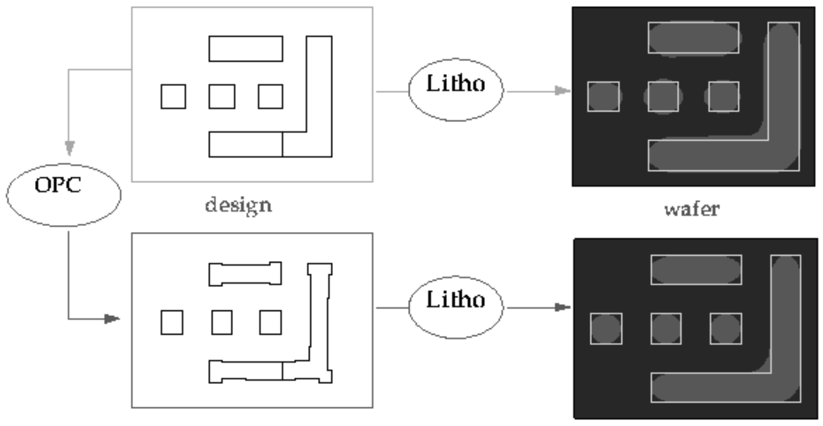


Figure 1.1: Conceptual approach to OPC

Within the lithography system, the imaging system alone, described by the Hopkin [19] equations for partial coherence, is a non-linear system and therefore a simple "transfer function" does not exist to relate mask to wafer. The conceptual "lithography inverter" does not exist because achieving perfect 3-D structures in the wafer with vertical walls and sharp corners is physically not possible. This is easily explained due to the bandlimited nature of the optics. Even if such a inverter existed, its mathematical description is beyond the current knowledge of the process physics involved in lithography. Given that, we will begin with the assumption that a unique, perfect OPC mask cannot be found in closed form.

However, we can certainly *improve* the pattern transfer performance by selecting an optimized mask design. In fact, it is an established practice in the IC industry to manually optimize wafer structures by adding small corrections to the shapes of mask polygons.

If we recognize that simulation can be used to assist with the addition of corrections, then a natural solution to OPC is to put very fast and highly accurate lithography simulation tools into a feedback system. The system moves mask edges to improve the simulated results on silicon. Therefore, OPC is transformed into a as a non-linear feedback control problem. In Figure 2.1, the OPC algorithm is the controller we seek to formulate. We will also need to build special simulation tools to achieve the speed that is required for full-chip automated OPC.

1.2 Benefits of OPC

The benefits include more accurate critical dimensions (CDs) and better edge placement. Moreover, OPC introduces the ability to shift process windows for different types of structures so that the overlap of process windows is enlarged. This allows more reliable pattern transfer at lower k_1 values. Tangible benefits for the IC industry can be measured as:

- Higher yield for a given minimum feature size, due to enhanced process windows
- Better circuit performance for a given minimum feature size, due to linewidth uniformity allowing faster clock rates
- Adoption of smaller design rules

1.3 Lithography Model

All steps in IC processing cause errors during pattern tranfer of the original layout design to silicon. Therefore, the realization of a circuit does not match the design. Some of the most significant process error sources can be listed as:

- mask writing
- optical diffraction
- resist development
- etch

When the systematic component of these errors is characterized, then the OPC system can potentially correct for them and improve the design-to-wafer pattern transfer fidelity. The next sections serve as brief introductions to some of the process effects that OPC can help to correct.

1.3.1 Mask writing

Just as optical lithography results in imperfect pattern transfer from mask to wafer, the mask manufacturing process introduces errors in the physical mask which cause it to deviate from the idealized mask. This is seen clearly in Figure 1.2 in which an OPC mask has been manufactured. The serifs and jogs on the OPC design do not transfer perfectly to the physical mask.

1.3.2 Optical System

The optics of lithography is one of the most well-understood parts of the process [24]. Assuming that optical diffraction is the size-limiting step in the entire lithographic process, the following optical scaling laws govern the final printed features. The minimum optically resolvable linewidth, L_{lw}, and the depth of focus (DOF), Z_{dof}, are given by:

$$L_{lw} = k_1 \frac{\lambda}{NA} \tag{1.1}$$

$$Z_{dof} = k_2 \frac{\lambda}{NA^2} \tag{1.2}$$

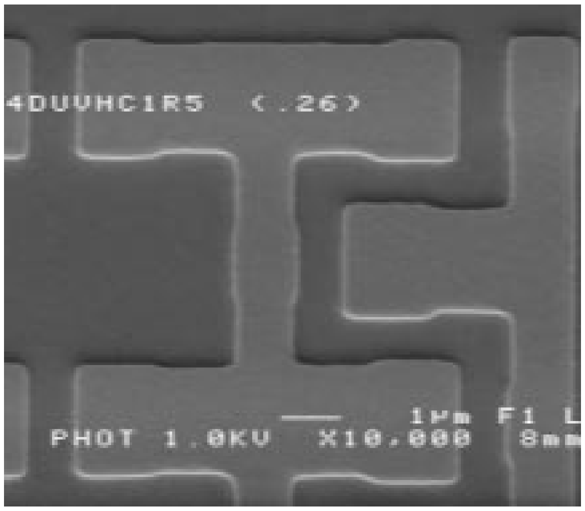


Figure 1.2: Example of OPC mask. The non-ideal reproduction of OPC corners and jogs on the mask is noted.

where λ is the light wavelength, NA is the numerical aperture of the optical system, $k_1 < 1$ is a number indicating technology improvements to the optical system, and k_2 is a scaling factor. Current process dimensions in the 0.25–0.5 μm range are in production with technology factor typically hovering somewhere below $k_1 = 0.7$ to as low as $k_1 = 0.5$. The goal of mask optical proximity correction is to improve the mask image at the resist surface, thereby compensating to some extent for loss of image quality caused by optical diffraction and effectively reducing k_1 to the range of 0.4 which would push DUV optical lithography into the sub 0.2 μm realm.

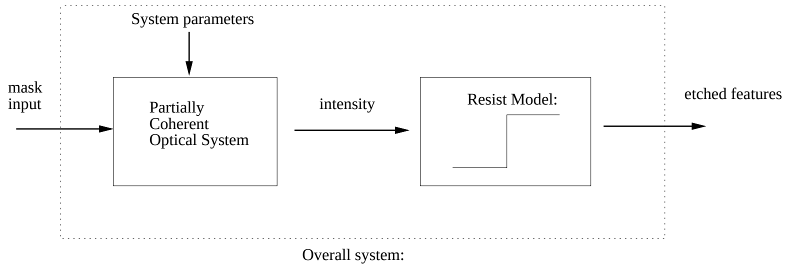


Figure 1.3: Naive model of resist (and etch), in which the wafer structures are assumed to print at the 0.3 contour of the aerial image.

1.3.3 Resist development

Resist development is fairly well-understood for many resists at DUV (248 nm) and i-line (365 nm). Resist behaves somewhat like a thresholder, for which a critical energy of impinging optical radiation causes the resist to "activate" and levels below which cause nothing to happen. In fact, one simple model for resist is simply to use a 0.3 level threshold of the normalized aerial image, as depicted in Figure 1.3. A 0.3 contour of an aerial image is shown in Figure 1.4.

For more accurate OPC, more sophisticated resist modeling is required. A few of the factors which affect resist are standing waves, post-exposure bake, and acid diffusion. Moreover, these effects are not the same for all resists, making resist modeling something far more complicated than the "naive model" of a simple thresholder.

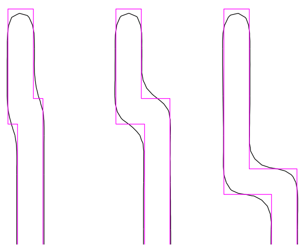


Figure 1.4: Example of aerial image prediction of wafer features by computing the 0.3 aerial image contour.

1.3.4 Etch

The wafer etch is an important step in the lithography process which can cause errors during pattern transfer. Loading effects caused by non-uniform concentration of etchant in different areas of the wafer have both large and small radii of influence. Although correction for etching effects is not considered a major goal of OPC, in contrast to more localized optical and resist effects, etch should be considered in the OPC algorithm. A simplified view of etch is that the etch process is well-behaved and introduces only a bias error to features on the wafer.

1.4 Mask Design Factors and Constraints

There are many issues relevant to mask design in order to ensure a robust and technologically feasible solution. Some of the most important issues are summarized below.

1.4.1 Accuracy/Critical Dimension

In placing patterns onto silicon via lithography, critical dimension accuracy (CD) is the fundamental concern. Achieving accurate CD means that the sizes of the final silicon patterns match the desired sizes. A 10% error tolerance in CD is normally cited as acceptable. Different types of accuracy measures are often used:

CD uniformity

The CD uniformity is a one-dimensional criteria that applies to linewidths of long lines. The term refers to variations in printed linewidth observed for a given target linewidth as the spacing to adjacent lines is varied. The iso/dense bias is an example of a CD uniformity issue.

CD linearity

Linearity refers to the accuracy at which linewidths are printed for a range of different target values.

line-end shortening

A large pullback is often seen at line-ends at the smallest dimension. This can be a significant problem when overlap between layers is required.

corner rounding

The effect of the bandlimited optics system on corners is that corners become rounded on the wafer. Again, overlay concerns warrant that corner rounding should be minimized.

1.4.2 Exposure and Defocus

Even though the image from a mask may result in good CD accuracy at the focus plane, it may not display good CD out-of-focus. This is a major concern because in reality the resist film has some thickness and topography. Variations from wafer to wafer in the mass production environment only highlight the need for exposure and defocus tolerance. So, for successful manufacturing, the image must display favorable characteristics throughout a range of defocus. Many techniques ranging from spatial filtering methods [13], multiple masks [44], to assist features and phase shift masks [23, 25] have been proposed to increase potential the depth of focus. The range of defocus for which the image stays within the CD tolerance is called the defocus window or defocus latitude.

The optimized masks should also display favorable CD versus exposure characteristics. This means that small variations in exposure level should still result in etched patterns which fall within the 10% tolerance. Good exposure performance is related to the robustness issue and is similarly ensured by high contrast imaging. The range of exposure for which the image stays within the CD tolerance is called the exposure window or exposure latitude.

The terms process latitude and process window refer to the amount of tolerable variation in both focus and exposure level. It is important to consider the process windows in performance analysis of designed masks.

1.4.3 Edge Placement

Edge placement is a term which refers to the placement of edges of lines and other features in comparison to the target placements as specified by the circuit designer. Edge placement errors (EPE) can be normalized to the minimum feature size to convey information similar to CD information. Using this normalization, 5% error for each edge of a line would guarantee CD accuracy within the 10% tolerance.

1.4.4 Mask Manufacturability

Some practical constraints limit the possibilities for what types of masks can be proposed. Two primary manufacturability constraints are:

- limited mask pattern complexity.

- limited number of mask transmission levels (for phase shifting masks)

Limiting the designed mask complexity is an issue most important in OPC where small features can be generated. Masks are made using laser or electron beam equipment which has finite resolution and accuracy. Highly intricate and complicated mask patterns stretch the ability of the mask making equipment to produce the desired mask.

In the design of phase-shift masks, it is important to consider the present technological difficulties of constructing a mask with more than a few different phase values, for example 0˙, 180°, 60°, and 120° on a single mask.

1.4.5 Mask inspection

After fabrication the masks must be inspected for defects. The issue of mask inspection becomes more difficult when mask patterns become complicated. In many cases, the automated mask inspection system might have trouble differentiating mask defects from complicated intentional OPC corrections, as shown in Figure 1.5.

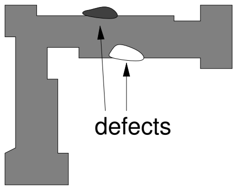


Figure 1.5: OPC features may be difficult to distinguish from real defects.

The OPC modifications should take into account limitations in mask inspection equipment to produce patterns which will not be mistaken for defects.

1.4.6 Data expansion

The OPC process introduces many new vertices to the existing polygons. The data expansion must be controlled inside the OPC algorithm to ensure a manageable file size. The data expansion factor, or DEF, can be expressed as:

$$DEF = \frac{\text{post OPC file size}}{\text{pre OPC file size}} \tag{1.3}$$

where the file size is the number of bytes required to describe a hierarchical chip layout.

1.5 Historical Perspectives on Mask Design and OPC

The concept of OPC has been in existence for many years. For extremely dense chips, circuit designers have used special knowledge to add manual OPC corrections since the 1970's. The idea of automating the OPC, and establishing it as a genuine step in IC manufacturing is a concept which has been evolving gradually. A brief historical perspective on OPC is presented here, with major techniques and advanced in the field highlighted.

1.5.1 E-Beam Proximity Effect Correction

The ideas from e-beam proximity correction [22, 16, 20, 10, 17] were the early inspiration for OPC. E-beam proximity effects, in contrast to optical diffraction effects in a partially coherent imaging system, have a linear system description. E-beam proximity effect correction can be performed by shape corrections, similar to OPC. However, in contrast to a stepper in optical lithography, the dosage of an e-beam machine can be changed from one spot to another, which allows dosage modulation to be used as a correction strategy.

The basic idea of e-beam proximity correction involves inverting the linear model for e-beam scattering to yield the corrected dosage at each pixel value. These methods are not directly extendible to the nonlinear problem of partially coherent imaging.

Many previous approaches to OPC for optical lithography attempted to solve the OPC problem by generalizing on the e-beam methods [35, 36]. The main problem with these approaches is that they use mask description models which often do not meet manufacturability constraints because they either have a continuum of transmission values, or result in patterns that are too complex to manufacture and inspect.

1.5.2 From Manual OPC to Automation

Manual OPC has been in existence in some shape or another for many years. Manual OPC means that an engineer will add serifs using trial and error until the desired pattern on the wafer is obtained. While manual OPC has been effective up until now, as the dimensions of critical features shrink, it has become apparent that the manual approach is not time/cost effective and, as such, systematic ways are needed to enable fast processing of large, complex chips.

A number of automated approaches to OPC have been proposed. Broadly speaking, we can classify these into two classes:

rule based techniques Use geometric rules to add corrections.

model based techniques Use lithography simulation to decide corrections.

Rule-Based OPC

Rule based techniques are an extension of the methods used for manual OPC. Through experiment or simulation the corrections that should be applied in a given geometrical situation are discovered. Then, a pattern recognition system is used to apply the corrections wherever that geometrical situation occurs throughout the entire layout design. While rule-based schemes are fast and therefore can be applied to an entire layout, they are not as accurate as desired. This is because their correction is not directly based on simulation.

An example of a rule-based technique is the method of Otto *et al*[31], in which simulation is used to generate geometric correction rules which can be applied for OPC. Experimental data can supplement the simulation data to "anchor" the rules to the particular lithography system in use. This approach has the benefit of fast correction time, because no simulation is performed as the corrections are being added.

Newmark[30] proposed a combination rules/model-based OPC scheme in which corrections are made to small structures using an iterative model-based algorithm. These form a library of precomputed corrections to selected patterns. Mask corrections are performed by interpolating the appropriate correction from the library. This approach has the advantage of requiring little computation since corrections are pre-stored. However, it is not known whether this method performs well for all patterns, and library generation remains a complicated task.

Model-Based OPC

The model-based OPC techniques are different from rule-based OPC in that simulation models are used to compute the wafer results and modify edges on the mask to improve the simulated wafer results. Model-based OPC is capable of more general corrections, but can require longer OPC time, because simulation is time-intensive.

Some of the early work in developing model-based OPC was done by Rieger and Stirniman[38, 39, 40]. They use experimental data to construct a lumped model of proximity effects from which corrections are decided. The zone sampling technique used by Rieger and Stirniman [38] is used to map directly from geometric structures through simulation to corrections. In their investigations of proximity effects they conclude that experimental data is necessary to fit the zone sampling functions for accurate OPC. With this approach, a model is fit to experimental data and then the model can be used for process-specific OPC. A high speed simulation system is the centerpiece of this OPC correction methodology.

In their efforts at formulating phase-shift mask design and OPC techniques, Liu and Zakhor[27, 26] use a model-based optimization technique with a pixel mask representation to enhance optical characteristics. However, the technique requires extremely long optimization times and can generate complicated mask designs. Subsequent efforts of Liu and Zakhor[28] begin to address issues of mask complexity, but still require impractical optimization times.

In the related area of phase-shift mask design, Pati and Kailath [33] use a double exposure system of masks which is designed by an automated algorithm. Their approach is noted because it has similarities to model-based OPC. The problem with these masks is similar to the problems experienced by Liu and Zakhor, in that the resulting phase shift masks are intricate and may cause difficulty during manufacturing.

Cobb and Zakhor[5, 6, 7, 8] addressed both the manufacturability and computational efficiency problems faced in the earlier model-based techniques of Liu and Zakhor. In their work, OPC has been formulated as an iterative algorithm which involves feedback of corrections. The modeling approach involves setting up a distinct optical model and distinct "black-box" models for major processing steps such as resist development, mask making and etch. The resulting OPC structure is called the simulation feedback optimizer. A resist model called the variable threshold resist (VTR) model [9] has been used to accurately model specific processes. This work will be discussed further in the remainder of this dissertation.

1.5.3 Growing role of simulation

Traditionally, the use of lithography simulation has been for analysis of aerial image and cutlines. Simulation studies of the impacts of changes in illumination or aberrations [42], or on mask defects [37] have improved the overall knowledge in lithography.

Presently, with the model-based OPC, and phase-shifting mask design algorithms being devised, the role of simulation is broadening. The new, broader role includes use of simulation within "mask design synthesis" tools. With this expansion of the role of simulation, the computational demands placed on simulation tools is growing larger and larger. The pioneering aerial image simulation tool from Berkeley, SPLAT [43], requires too much computation time for the large areas being simulated in OPC or phase-shifting mask design.

To speed up aerial image simulation, many techniques have been devised. A simple technique of using the full Hopkins [19] TCC function with windowing [5] limits the growth rate of simulation. In addition, a set of techniques has developed which uses a decomposition and approximation to the Hopkins equations which poses aerial image calculation as convolution operations. The technique is called different names from various authors, but involve similar ideas, of breaking down the Hopkins equations into eigenfunctions. Gamo [14] simply calls the technique a matrix treatment of partial coherence. From Wolf [46], a "spectral representation" is described, using Fourier decomposition. Other work by Saleh[36] describes a "modal expansion" involving a similar decomposition. Rieger and Stirniman [38] cite Gamo [14] and call their technique "zone sampling". They have incorporated it into their OPC techniques. Interesting results have been shown by Bunau [44], in which closed form expressions for the convolution kernels are derived, calling them "Optimal Coherent Approximations", or OCA's. The OCA name is also used by Pati and Kailath [34] in describing how a Mercer expansion can yield the OCA's. In Pati and Kailath's work, the OCA's are then used in the design of phase-shift masks. In the work presented here, a similar technique is called Sum-Of-Coherent Systems or SOCS, and uses a numerical solution involving Singular Value Decomposition.

A more complete description of this useful approximation and speedup technique is described in Chapter 3.1.

1.6 Outline

The outline of this thesis is as follows. In Chapter 2, we discuss the overall OPC system and describe the components required to implement OPC. In Chapter 3, we discuss a fast convolution technique for convolving area-wise constant functions with general convolution kernels. This technique is particularly relevant because aerial image simulation for OPC can be performed using this framework. In Chapter 4, we show how the Hopkins equations for partially coherent optics can be decomposed into a Sum Of Coherent Systems model. In Chapter 5 we discuss a philosophy for lithography simulation modeling suitable for OPC. Also, a variable threshold resist model is proposed which can be used effectively to predict CD. In Chapter 6 we show more simulation studies of OPC. The stability of the OPC algorithm is examined, and the performance past $k_1 = 0.4$ is studied. In Chapter 7 we present experimental results in which OPC was performed on test patterns and printed on silicon.

Chapter 2

System Architecture

2.1 Problem formulation

The proposed OPC system[6, 7, 8] shapes mask geometry to increase placement and CD accuracy of etched structures. The system block diagram is illustrated in Figure 2.1. This can be viewed as a non-linear control problem in which an error signal drives the correction in order to reduce the error. The error signal is an edge placement error (EPE) which is determined by process simulation. The process simulation has been placed inside the feedback loop. This system achieves accurate correction by continuously monitoring the simulated wafer as the mask is modified. With this implementation, specific models can be inserted into in the system to achieve higher accuracy for a given process. In this way, the OPC algorithm itself is independent of the simulation models which drive it.

2.2 Overview

The OPC system operates by first "fragmenting" the mask polygons into edge and corner segments which are to be moved during OPC. Then, sparsely chosen control sites are installed at specific locations on the wafer[7]. The fragments are the optimization variables, which can be offset from their original positions, as shown in Figure 2.2. The EPE at the control sites is used to compute the cost function. As the OPC progresses, small perturbation polygons are added and subtracted from the mask to result in a mask which improves edge placements on the wafer. Because the mask may be subjected to millions of perturbations during OPC, the simulation system is designed especially to handle fast

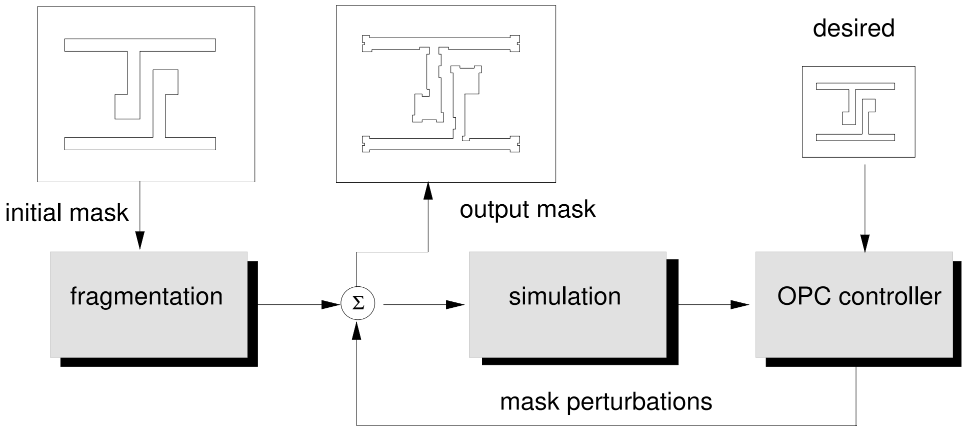


Figure 2.1: OPC system

perturbations at the sparse sites. In summary, the following steps are used in the proposed OPC algorithm:

- Fragmentation to create variables
- Cost function
- Edge updates
- Simulation updates

In the next sections, the steps listed above will be described in detail.

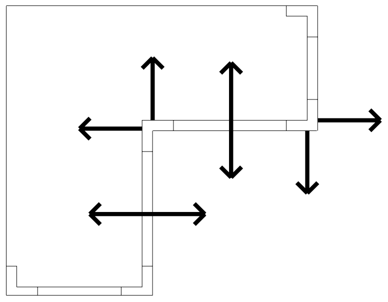


Figure 2.2: Movement of fragmented edges in mask representation

2.3 Fragmentation

Fragmentation is the operation which breaks edges into smaller edge segments to allow more degrees of freedom during the OPC movement. Fragmentation is an important part of OPC, because the number of additional vertices produced by OPC affects not only the algorithm speed, but also many other factors. For instance, mask writing time is dramatically affected by the number of vertices in a layout. Furthermore, mask inspection also becomes more complicated for a mask which has many new corners and jogs. Also, the overall layout database size depends on the total number of vertices in the representation.

There are several variations of fragmentation, from constant to adaptive fragmentation which provides considerable control over how fragmentation is performed.

2.3.1 Constant distance fragmentation

In a constant distance fragmentation approach, polygon edges are broken up into smaller edge segments of a fixed length. For example, all edges could be broken into 0.2 μm sections as in Figure 2.3. During OPC, each of the fragments can move independently to decrease the error.

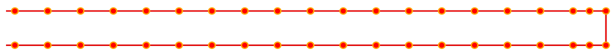


Figure 2.3: Constant distance fragmentation

The disadvantage of this approach is that it typically results in many more edge fragments being generated than are actually needed to achieve accurate results.

The simple example of a long line, above, demonstrates this point. For a long isolated line, the final offset for each edge fragment in the middle should be equal. Therefore, the ability to move many small edge segments is not only superfluous, but could result in significant performance cost.

2.3.2 Adaptive fragmentation

A better alternative to constant distance fragmentation is an adaptive fragmentation approach in which edges are only broken into smaller fragments if more degrees of freedom are required in the local area.

In the one-dimensional case previously seen in Figure 2.3, it was noted that in the middle of the line, all edges should have the same offset value. However, near the corners, the mask edges may need to move with different offsets in order to improve the edge placement error. This insight provides a basic approach to intelligent fragmentation: adaptive fragmentation should add more fragments near corners. Two types of fragmentation are performed:

- intrafeature fragmentation
- interfeature fragmentation

Intrafeature fragmentation

Intrafeature fragmentation causes fragments to be added along the two edges which connect to a given corner, as seen in Figure 2.4. In the figure, the values for `len1`, `len2`, and `len3` are fragmenation distances will will cause vertices to be inserted.

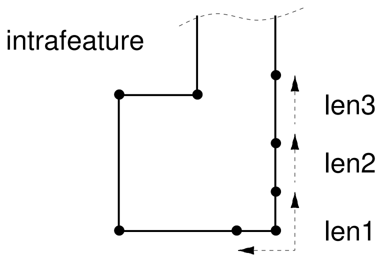


Figure 2.4: intrafeature fragmentation

Interfeature fragmentation

Interfeature fragmentation causes edges to be generated along an edge where corners are close enough to have an interaction. An example of interfeature fragmentation is seen in Figure 2.5. The "ripple" fragments are allowed in order to give additional degrees of freedom.

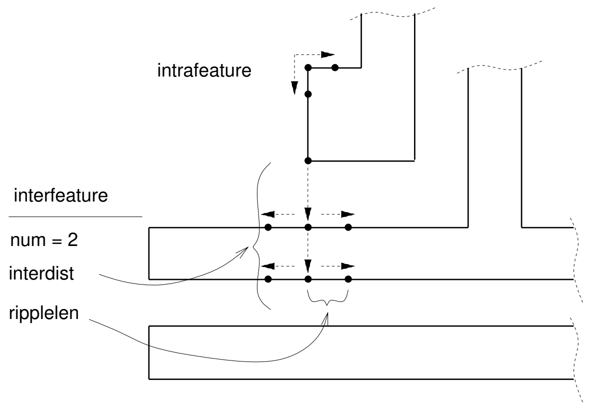


Figure 2.5: interfeature fragmentation

2.4 Cost function

After the fragmentation step is complete, we have the edges which are our optimization variables. These edges move during OPC to minimize a error function, or cost function. Conceptually, the error is the difference between the desired polygons and the simulated wafer structures, as computed by a simulation system, for example the simulation model shown in Figure 2.6.

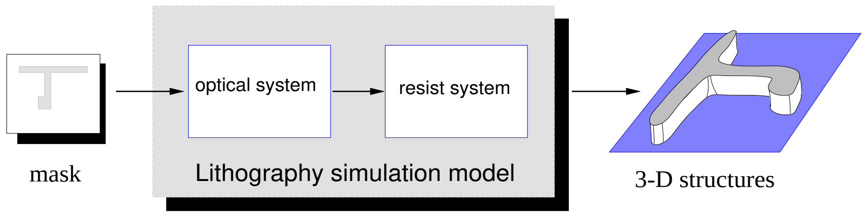


Figure 2.6: Lithography simulation for OPC

At a single point along a contour we can define the Edge Placement Error (EPE):

$$EPE(x) \quad = \quad D(x) - W(x) \tag{2.1}$$

as a parametric function of x, which is the dummy variable which can take values between 0 and 1. Varying x from 0 to 1 moves us along the perimeter of a desired contour $D(\cdot)$ and the contour of the simulated wafer structure $W(\cdot)$. This error metric is easily related to the diagram in Figure 2.7.

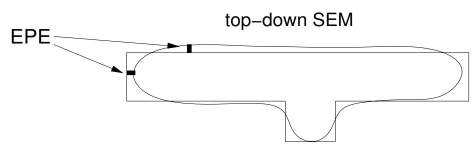


Figure 2.7: Edge placement error

The mean-square-error of the entire contour is given by:

$$cost = MSE = \int |EPE(x)|^2 dx \quad (2.2)$$
$$= \int |D(x) - W(x)|^2 dx \quad (2.3)$$

Instead of computing the cost at all locations along the contour, we use sparse sampling to sample only at a minimal number of locations. A reasonable technique for assigning control locations is to have one control site per fragmented edge. This produces a situation in which control sites are situated as in the example of Figure 2.8. Then, the integral for the cost function becomes a summation over all control sites:

$$cost = MSE = \sum_i |EPE(x)|^2 \quad (2.4)$$
$$= \sum_i |D(x) - W(x)|^2 \quad (2.5)$$

where the index i ranges over all control sites.

2.5 Optimization algorithm

2.5.1 Edge offset update criteria

The optimization variables were identified in Section 2.3 as fragmented edges, and the cost function has been defined in Equation 2.5. A basic gradient descent optimization algorithm is now employed to optimize the mask edge offsets. To find the local mimima, the cost function is simply differentiated with respect to the offset of the edge variable in question, e. The edge offset distance, indicated in Figure 2.2 is then varied according the the result. The partial derivative will indicate the effect of movement of the given edge on the cost function. The direction, and magnitude of movement of the edge is decided by the value of the partial derivative.

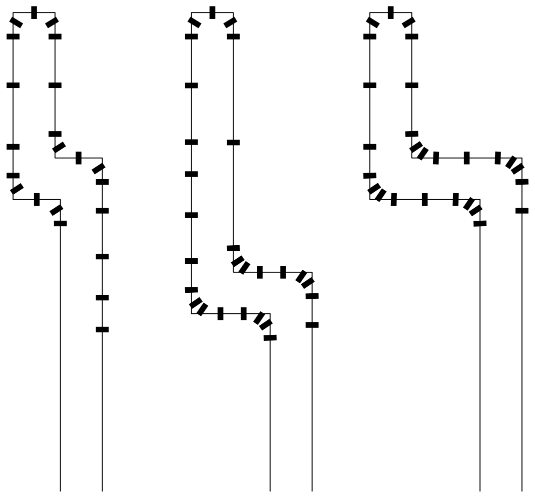


Figure 2.8: Sparse sampling of wafer

$$D = \frac{\partial f}{\partial e} \tag{2.6}$$

The edge offset e will be adjusted according to the following edge offset update criteria, to produce a new value for the offset:

$$e_{new} = \begin{cases} e & \text{if } D = 0 \\ e + step & \text{if } D < 0 \\ e - step & \text{if } D > 0 \end{cases} \tag{2.7}$$

2.5.2 Numerical computation of the gradient

A closed form expression for D is not easily derived due to the complexity of the non-linear Hopkins intensity equations and the process simulation models. Fortunately, no such expression is required, because simulation tools exist to compute the EPE values numerically. Furthermore, our goal is to *separate* the simulation models from OPC algorithm itself, so, we would not use a closed form expression for D even if it existed.

In any case, only the sign of D is needed in order to compute the edge offset updates as described in Equation 2.7. Therefore, edge offset updates can be performed merely by computing the cost function resulting from the edge offset e in each of 3 positions and choosing the one with the minimum error.

An example illustrating the sequence of conditional operations for deciding movement is depicted in Figure 2.9. In this figure, an edge is under consideration for movement. The edge displacement, ν, is the optimization variable. The cost of its initial displacement is recorded with the value $cost = 10$. After first moving the edge outward, the cost increases to $cost = 20$ and therefore the movement is rejected. Next, the edge is moved inward from its original position and the cost is again evaluated. The cost, $cost = 15$, is again higher than the original displacement cost so the object is moved back to its displacement at the beginning of the iteration pass for that edge. In the current implementation, the step size, $\Delta\nu$, is fixed to a value typically around $\Delta\nu = 10$ nm.

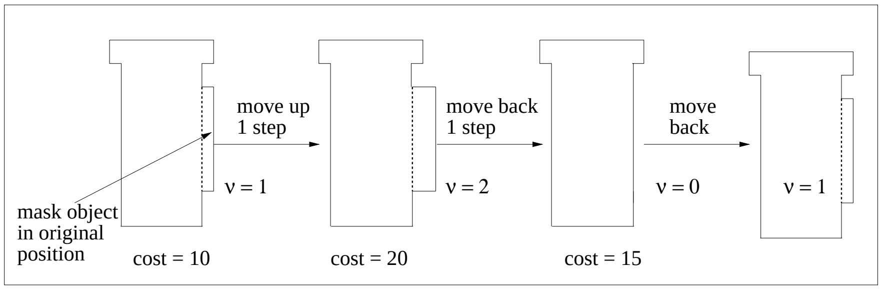


Figure 2.9: Example of algorithm progression

2.5.3 OPC iterations

The overall OPC algorithm is put together by iterations of the edge update criteria described previously for each edge in the mask. We call a single iteration the process of going through each edge in the mask once. Multiple edge update iterations are required to complete the entire OPC run. The iterations can be halted by specifying a fixed number of iterations, such as 8, or by continuing until the cost is less than some pre-specified threshold.

The following procedure summarizes the OPC algorithm:

```
while (edge error too high)
  for (edges in the contour)
    1. compute original cost
    2. perturb edge displacement in outward direction by one stepsize
       compute new cost
    3. perturb edge displacement in opposite direction by one stepsize
       compute new cost
    4. use value for edge displacement which results in lowest cost
  end for loop
end while loop
```

2.6 Simulation updates

Each time an edge is updated, the simulated EPE must either be updated or recomputed. One can interpret the movement of a single edge to be a trapezoidal perturbation to the mask. This perturbation will affect the aerial image of any simulated sites roughly within λ/NA of the perturbation. An example is shown in Figure 2.10 where the rectangular perturbation will affect all control sites with the larger rectangle indicated with dashed lines. In the figure, all control sites are shown using small black boxes.

The SOCS representation discussed in Chapter 3 provides an efficient way of merely *updating* the aerial image due to the perturbation rather than recomputing the entire aerial image from scratch for all control sites. Perturbation of the aerial image is discussed in more detail in Section 3.2.5.

2.7 Constraints for Mask Manufacturability

Because we have adopted the view of OPC as an optimization problem, it is not difficult to modify the constraints of the problem to enforce manufacturability criteria.

2.7.1 Minimum edge length

One constraint which can be easily imposed on the fragmentation stage is a minimum edge length constraint.

Constraint: During fragmentation, do not produce any new edges which are less than d_{min}, the minimum edge length.

2.7.2 Aspect ratio

The "aspect ratio" of a newly added jog can be defined as the length of the jog, over the length of the discontinuity it produces with the neighboring edge. This can be seen more readily in Figure 2.11.

Forcing the aspect ratio to remain higher than some minimum number will cause the algorithm to produce more manufacturable masks because discontinuities at jogs are controlled. Typically, an aspect ratio of 4 or larger results in manufacturable and inspectable

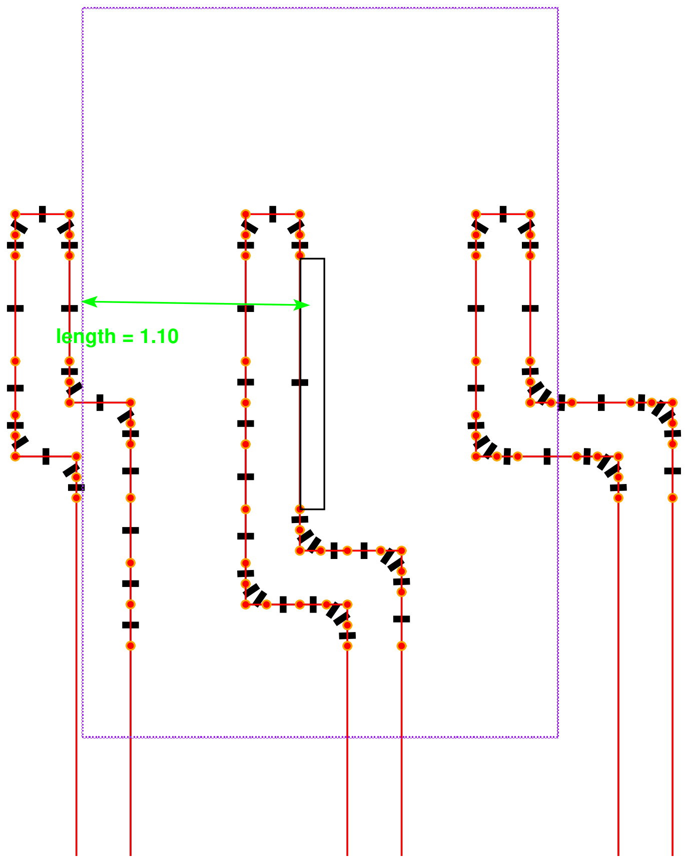


Figure 2.10: Area of influence of a single perturbation. All control sites which are affected must be updated.

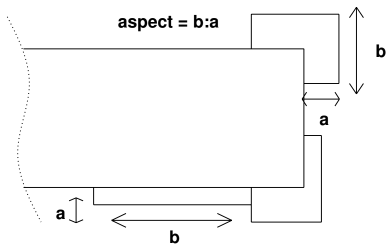


Figure 2.11: Aspect ratio for corrections

jogs. The aspect ratio constraint is one which can be enforced during the edge offset updates inside the OPC algorithm.

Constraint: During edge updates, do not allow movement which will cause the aspect ratio to become smaller than some mimumum number.

2.8 Algorithm complexity analysis

In the section we will briefly outline the order of growth of the outlined OPC algorithm with respect to a straightforward attribute: density of fragmented edge, ρ_e. As described in Section 2.5.3, we loop over the total number of edges in the design, M_e, which is proportional to the density:

$$M_e \propto \rho_e$$

For each edge update, the perturbation will require updating the intensity at a number of sites also proportional to the density of edges, because the number of sites is proportional to the number of edges. Then, if we assume constant time is required to update the intensity at each control site, then we conclude that the overall algorithm grows as follows:

$$\text{COMP} = \mathcal{O}(\rho_e^2) \tag{2.8}$$

In practice the density of fragmented edges is not uniform since we use and adaptive fragmentation algorithm, but this analysis is nevertheless useful in thinking about the growth rate of the OPC problem.

2.9 Results

OPC is run on some examples and the decrease in EPE is shown using simulation. These results show how the OPC algorithm reduces the standard deviation of the EPE for each iteration.

2.9.1 Example 1

The first example is for i-line with 0.36 μm minimum feature size. The following information summarizes the experimental conditions and results.

Summary	
name	pub36
target CD	0.36 μm
lambda	0.365
NA	0.55
sigma	0.65
k_1	0.54
iterations	8
mean EPE	before 3.14 after 0.44
std EPE	before 27.3 after 7.1
file sizeE	before 16.3Kb after 38.9Kb

The original and corrected patterns are shown in Figure 2.15. The convergence of EPE can be observed in Figure 2.12 where the σ of the EPE decreases for each iteration. The EPE for 2000 locations is shown in Figure 2.13 and Figure 2.14 before and after OPC.

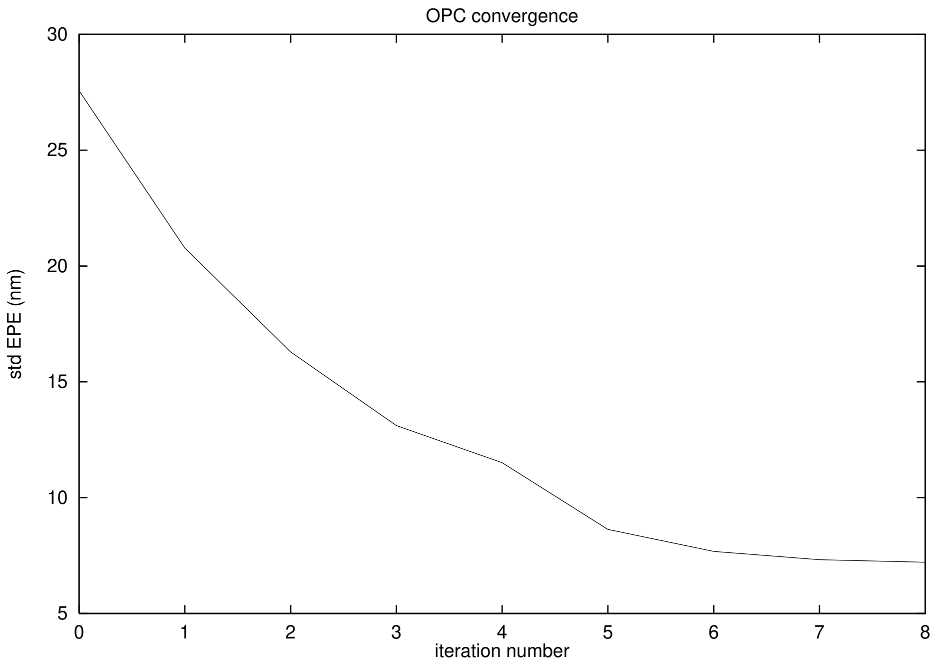


Figure 2.12: Convergence of EPE during OPC iterations

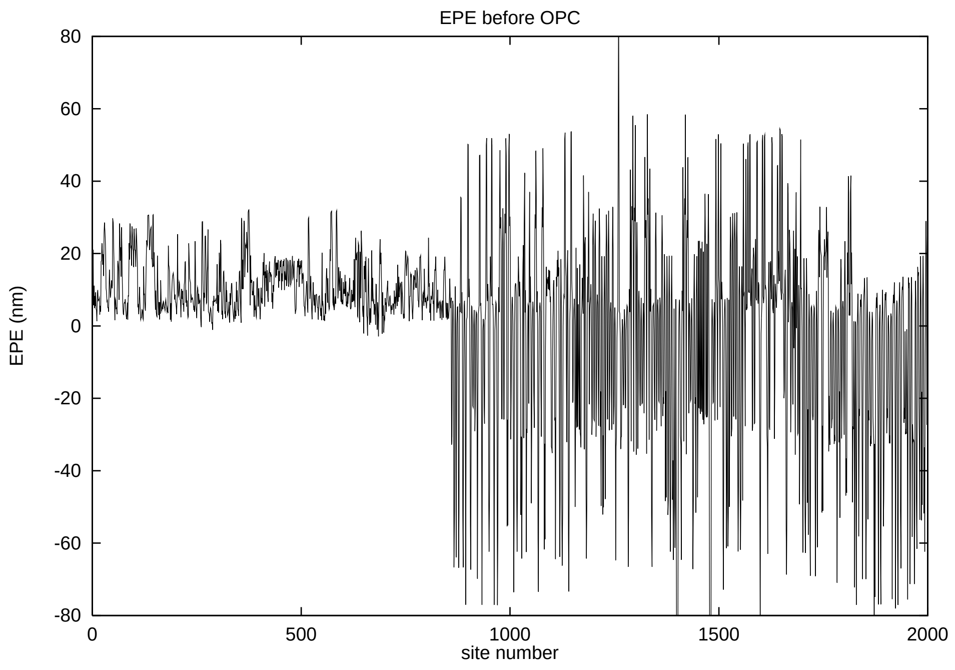


Figure 2.13: EPE at 2000 control sites before OPC for 0.36 μm pattern

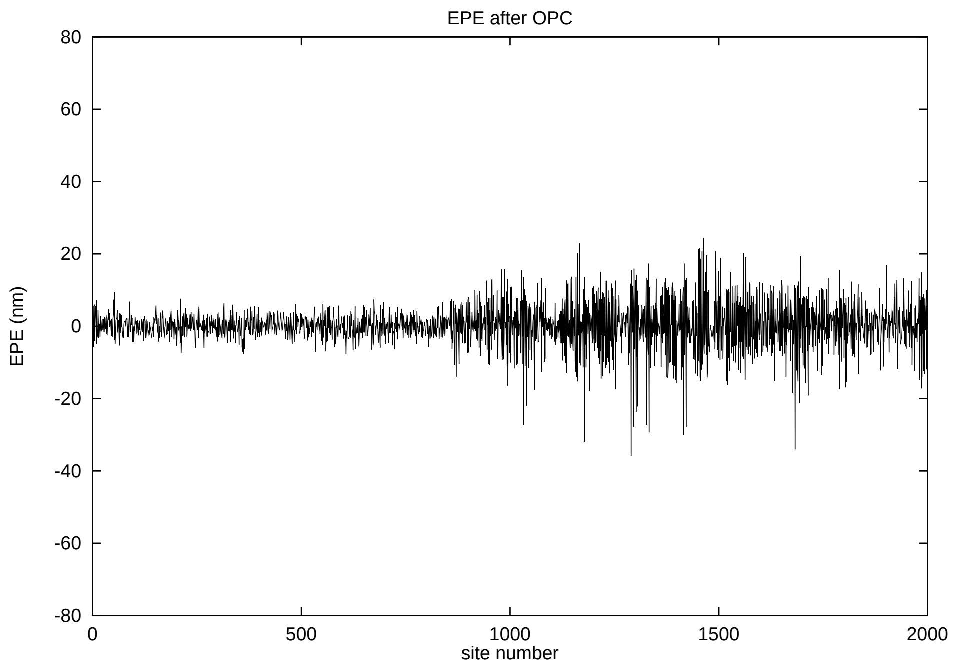


Figure 2.14: EPE at 2000 control sites after OPC for 0.36 μm pattern

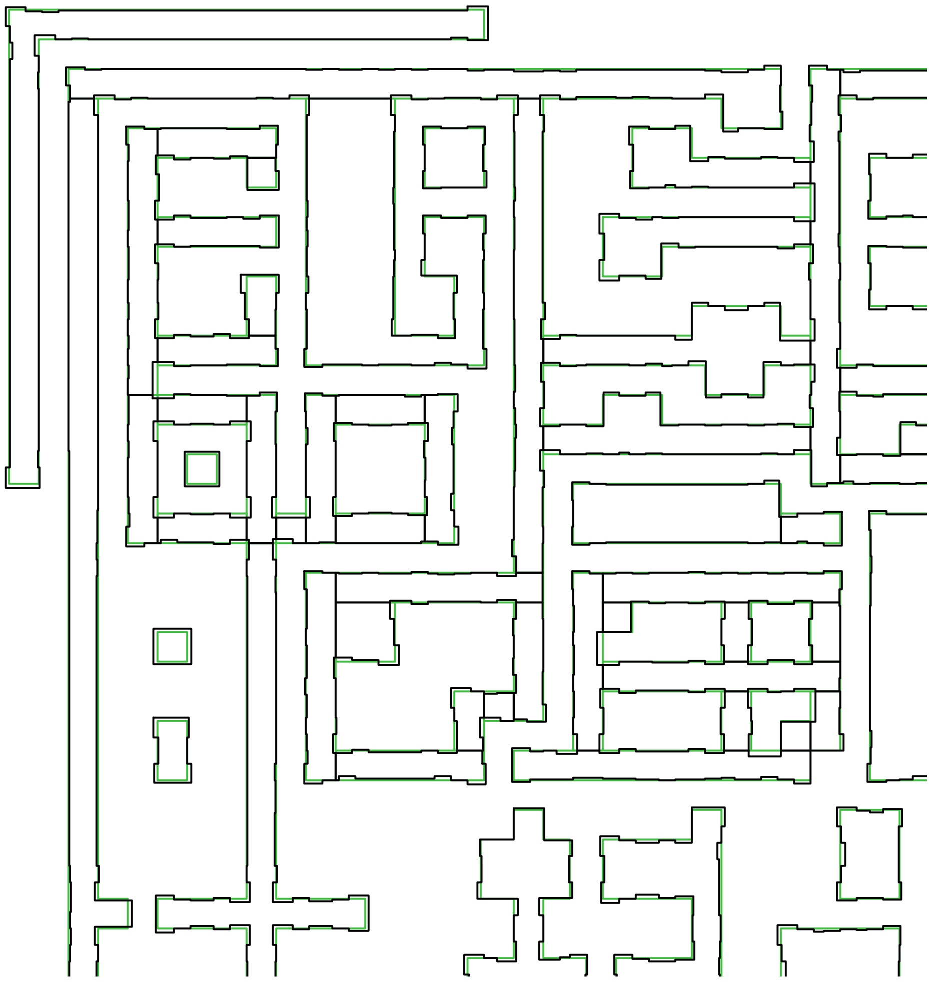


Figure 2.15: OPC correction of a 0.36 μm pattern

2.9.2 Example 2

This example is for DUV with 0.25 μm minimum feature size. The original and corrected patterns are shown in Figure 2.16. The simulated corrected results for a small section are shown Figure 2.17. In this figure, the solid black lines represent the desired wafer results. The gray curved line represents the simulated intensity on the wafer after OPC, and the darker curved line represents the simulated wafer intensity before OPC. The improvement at the line-end is noted in Figure 2.17. The following information summarizes the experimental conditions and results.

Summary	
name	ss25
target CD	0.25 μm
lambda	0.248
NA	0.50
sigma	0.60
k_1	0.50
iterations	8
mean EPE	before 2.4 nm after 0.7 nm
std EPE	before 21.7 nm after 7.7 nm
file sizeE	before 1115 bytes after 3289 bytes

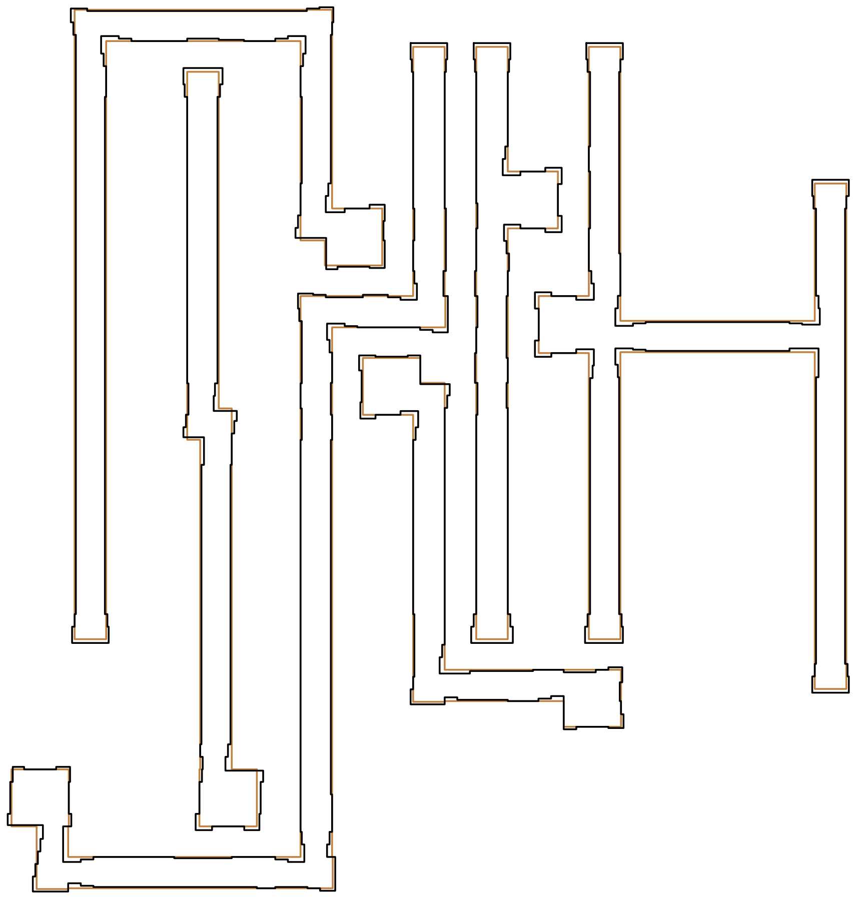


Figure 2.16: OPC correction of a 0.25 μm pattern

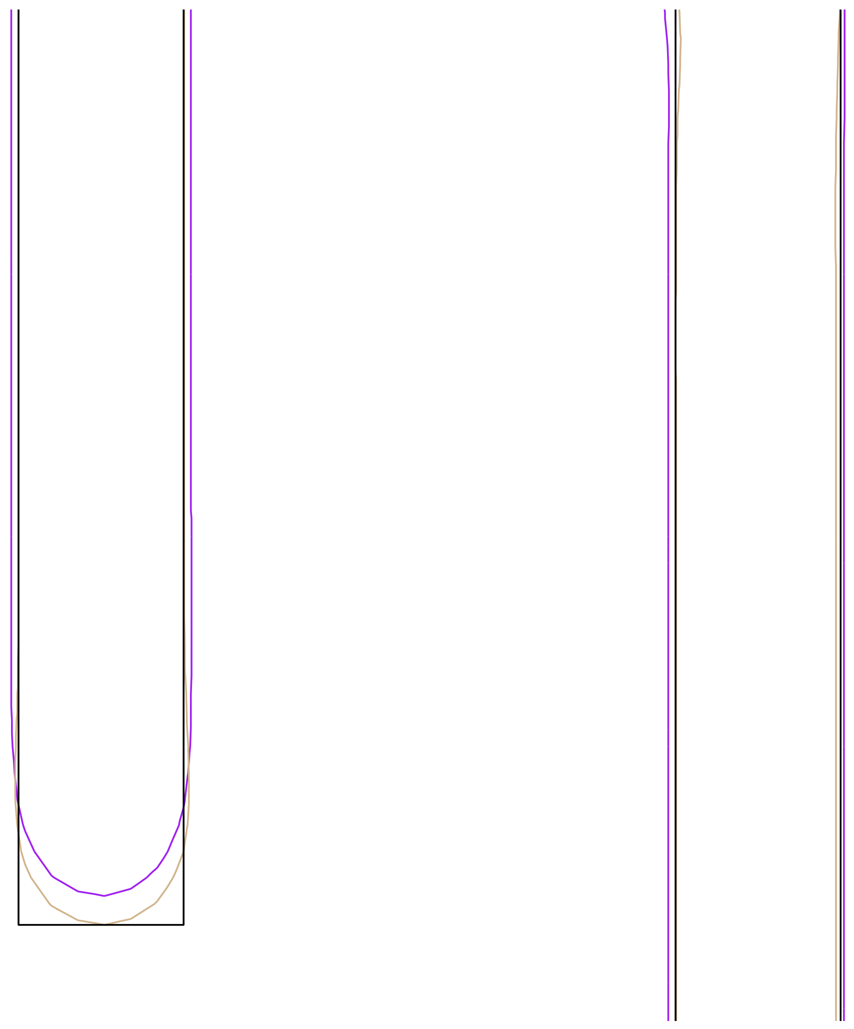


Figure 2.17: Zoom-in of simulated aerial image intensity of a 0.25 μm pattern, before (dark gray) and after (light gray) OPC, superimposed with the design (black).

2.9.3 Example 3

An example of OPC on a 0.18 μm DUV design is presented here. The original and corrected patterns are overlayed in Figure 2.18.

Summary	
name	tp18king
target CD	0.18 μm
lambda	0.248
NA	0.50
sigma	0.60
k_1	0.37
iterations	8
mean EPE	before 2.9 nm after 0.6 nm
std EPE	before 21.9 nm after 7.5 nm

2.10 Conclusions

The OPC algorithm formulated in this section works well in decreasing the EPE. In simulation studies, the EPE converges toward zero after just a few iterations. The results of running OPC on several designs shows that given perfect simulation models, our feedback OPC system is capable of highly accurate corrections, resulting in greatly reduced edge placement error and greater CD uniformity.

The observation that the edge placement error after OPC does not go to zero, even with perfect simulation models, is attributable to several causes. First, the OPC movements are performed with a discrete stepsize, normally 5–10 nm. Given the restriction to 10 nm grids, one would only expect ± 5 nm accuracy at best. Secondly, our OPC algorithm is a optimization approach, using a “greedy” gradient descent algorithm. Hence we would not expect to find a global minimum.

The simulation studies of how the EPE error is reduced after OPC provide good insight into the value of OPC. In practice, unmodeled process phenomena would reduce the correction gains of OPC by a certain margin. For example, if the accuracy of the simulation models is only ± 10 nm, then, the simulated EPE after OPC also has a ± 10 nm uncertainty added. The interpretation of this result is that model accuracy is the primary limitation to OPC. However, this is desirable, since the OPC algorithm is itself independent of the simulation models used. This means that as simulation models are improved, they can be

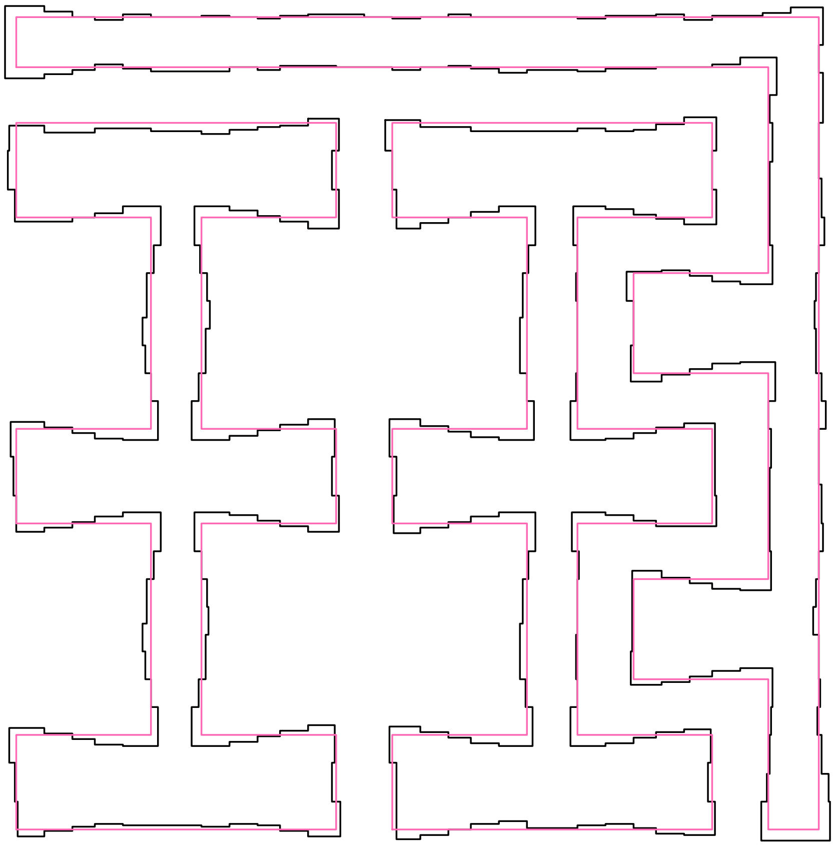


Figure 2.18: 0.18 μm OPC corrected pattern

inserted into the system architecture using the same OPC algorithm, making our proposed OPC algorithm a general and extendable solution.

Chapter 3

Fast aerial image computation

The aerial image has long been used as a first order approximation to the final etched features produced by microlithography. OPC techniques have been used to correct the aerial image for a constant threshold value to improve results on the wafer[6, 31, 39].

To be practical for a full-chip solution, the simulation involved in model-based OPC must be extremely efficient. Fast and accurate aerial image intensity simulation is required by OPC because the feedback nature of the OPC algorithm employs millions of intensity calcuations. In the past, one of the biggest bottlenecks in simulation has been the aerial image intensity simulation, shown in Figure 2.6. Fortunately, the optical sub-system is one of the most easily characterized and most well understood component in the entire lithography process. The structure of this system allows us to speed up the calculations considerably.

In this chapter, we will outline a fast method for aerial image computation derived from the Hopkins model. Two techniques are combined together to achieve a high-speed simulation module:

- Sum-Of-Coherent Systems Decomposition
- Edge lookup for convolution

The first step towards fast aerial image intensity simulation is a eigenfunction decomposition of the Hopkins imaging equations which produces a Sum-Of-Coherent Systems, or SOCS, decomposition of aerial imaging [7, 34, 46], as discussed in Chapter 4. The SOCS decomposition reveals that aerial image can be computed using a number of shift invariant linear systems.

Once the connection to shift invariant linear systems and convolution has been established, a highly efficient, highly general way of performing convolutions on a restricted set of functions will be shown. The convolution technique can be implemented by lookup-tables to result in minimal computational expense. as described by Cobb and Zakhor [7, 8]. The technique of storing lookups for convolutions with "upper right corner rectangles" has the advantage of being applicable to arbitrary layout geometry. It will be shown here that this technique is equivalent to storing lookups with individual edges.

Other work featuring lookup tables for fast convolution The lookup table technique discussed in this thesis is similar to a lookup technique described by Lee *et al*[22, 20]. In their work, lookup tables are pre-computed for the convolution of layout data with kernels which describe e-beam scattering. The convolution results are computed for rectangles of various sizes and stored.

Pati *et al*[32] have proposed a similar lookup table technique for fast aerial image calculation. Their technique uses the OCA convolution kernels as a starting point. Then, building block, or basis images are computed by convolution of the kernels with a small number of polygon structures. The basis functions chosen in their work are step functions with a finite width in one direction, or "corner functions" similar to the "upper right corner rectangles" described in this work. The disadvantage of the finite width step functions is that this they do not allow lookup of arbitrary geometries. The more general "corner function" provide this extra benefit.

3.1 The System Representation

The widely used Hopkins model for aerial image calculation[1, 12, 42] provides a general, parametric scalar imaging formulation. The imaging system model used for our technique is a decomposition of Hopkins. We use an eigenfunction decomposition of the transmission cross coefficient (TCC) function obtained in the Hopkins imaging formulation. This decomposition is discussed more fully in Chapter 4.

In this decomposition, the observed image intensity is described by several coherent fields whose mutual interaction is incoherent, as shown in Figure 3.1 The technique shows a resemblance the point-source integration technique. In the point-source integration technique, the aerial image intensity resulting from partially coherent optics is decomposed into the sum of intensity due to many point sources at the aperture.

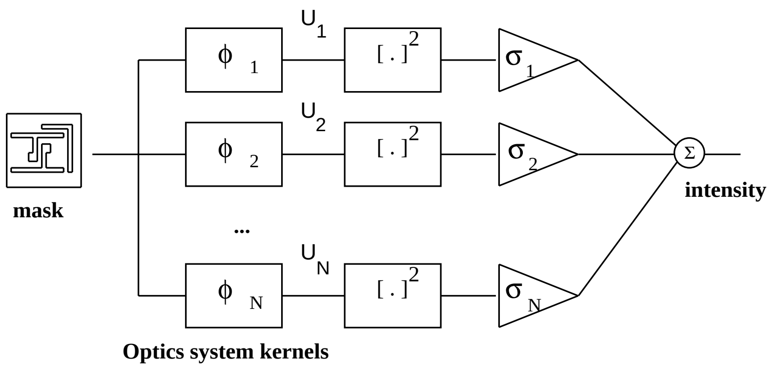


Figure 3.1: Sum of coherent systems representation of Hopkins imaging

In this work, we call the new system a Sum-Of-Coherent Systems (SOCS) representation to emphasize the structure of the decomposition. The block diagram in Figure 3.1 shows the form of the SOCS imaging system, with an individual kernel shown in Figure 3.2. The intention here is to start with the SOCS decomposition and formulate a highly efficient method for aerial image calculation. A complete derivation of the SOCS representation can be found in Chapter 4.

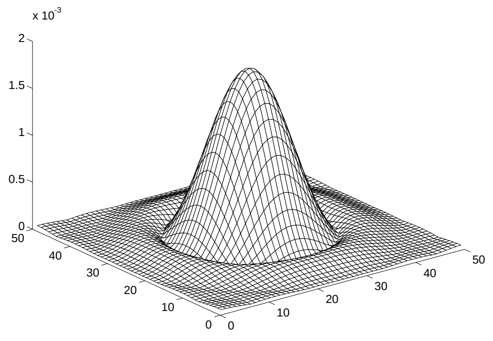


Figure 3.2: Example of a coherent imaging kernel

3.2 Fast Intensity Point Calculation Using Mask Polygons

The technique to be described exploits the imaging system discussed in section 3.1 which consists of a bank of linear systems as shown in Figure 3.1. Using this as a starting point, we develop a method for fast sparse intensity point calculation. For our method, the mask is described by polygons which define area-wise constant functions. As will be shown, the intensity at a given point in the image can be computed by considering the effect of each polygon edge separately and combining the individual edge effects to obtain the intensity value, as depicted in the overall system block diagram shown in Figure 3.3.

The SOCS decomposition is the link between image calculation and convolution. Using the SOCS, image intensity is computed by squaring, scaling, and summing the field values U_i resulting from each linear shift-invariant system. Convolution is the most computationally intensive operation involved. Because of the simple form of the mask transmission function, lookup tables can be utilized to reduce the convolution operation to a lookup operation. Lookup methods for rectangular geometry have previously been applied to e-beam proximity effect correction by Cook and Lee[10] and to OPC by Cobb and Zakhor[7]. The fast lookup method we present here generalizes our previous work and applies to arbitrary polygonal geometry and mask transmission values.

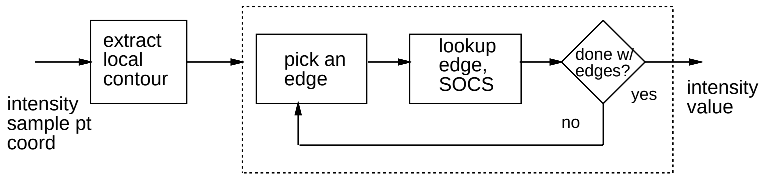


Figure 3.3: Intensity point computation algorithm

3.2.1 Mask Polygons

Polygons are the standard way to represent mask features. The special case of rectangles or boxes is used when dealing with Manhattan geometry masks. Polygons can overlap in the data representation because in the physical realization of a mask these overlaps have no effect and therefore circuit designers often insert overlapping polygons. An example of a mask pattern made up of overlapping polygons is shown in Figure 3.4.

The mask function needed for image calculation is described implicitly by the mask polygons. Thus, the mask function, M, is an area-wise constant function, or more generally an indicator function of the union of the areas covered by the polygons. For nonoverlapping polygons this is simply:

$$M = \sum_{k=1}^{N_p} t_k 1_{p_k} \tag{3.1}$$

where M is the mask function for a mask with N_p polygons, t_k is the transmission value for polygon p_k, and 1_{p_k} is the indicator function for polygon p_k. The indicator for a polygon is simply "on" for points inside the polygon and "off" for points outside the polygon:

$$1_p(x, y) = \begin{cases} 1 & (x, y) \in p \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In what follows, the mask polygons are always assumed to be non-overlapping. In the case where layout data has overlapping polygons, a boolean OR operation described in Appendix A can be used to recompute non-overlapping polygons which represent the union of all covered areas. In our discussion, polygons are always oriented such that the polygon interior is to the right as we traverse around the polygon and and the exterior is to the left. Figure 3.4 shows a section of the mask with overlapping polygons and Figure 3.5 shows contour plot of the indicator function resulting from the union of polygons. The contour is the simple closed directed curve for which the indicator function is "on" to the right of the curve and "off" to the left. The use of polygon boolean operations is an important issue in the computation of aerial image for data which has overlapping polygons, but the derivation of this operation lies outside the scope of the current discussion.

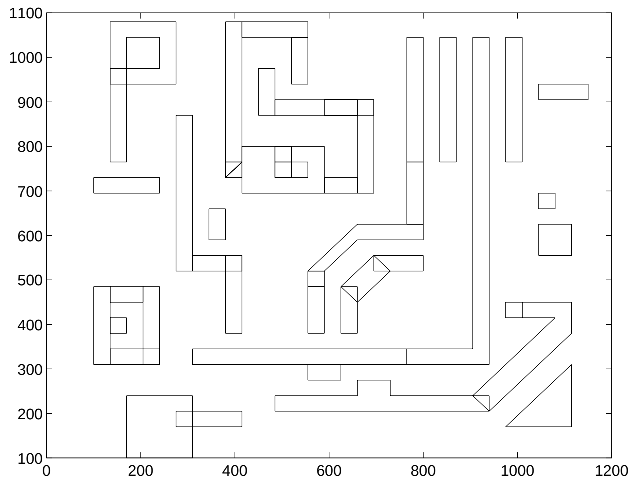


Figure 3.4: Mask representation as (overlapping) polygons

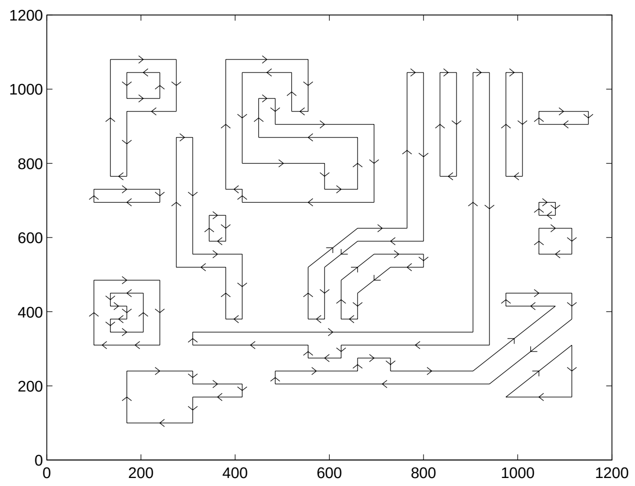


Figure 3.5: Mask contour with overlaps removed

3.2.2 Convolution of general functions with mask data

Having established that a mask function is an area-wise constant function defined on a 2-D domain, we can formulate a general convolution technique for this class of functions. The technique which will be presented describes a *general* method for fast convolution of 2-D area-wise constant functions with arbitrary functions of finite support. The usefulness of these techniques certainly extends beyond OPC.

As stated previously, convolution applies because the decomposition of the Hopkins imaging equation yields a SOCS representation using linear shift-invariant systems whose outputs are squared, scaled, and summed. In considering this system, the rest of this section is confined to computing the output of a single linear system, h, at a single point, an example being point (x_0, y_0) in Figure 3.6. This amounts to an inner product of the mask function with the convolution kernel.

$$\begin{aligned} y(x_0, y_0) &= \ < M, h > |_{(x_0,y_0)} & (3.3) \\ &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} M(x_0 + N/2 - i, y_0 + M/2 - j) * h(i,j) & (3.4) \end{aligned}$$

for an $M \times N$ discrete 2-D kernel.

The linear systems obtained from the SOCS decomposition have point-spread functions, or kernels, with finite support region on the mask. For example, the first order kernel,

$\phi_1(x, y)$, of the approximation is shown in Figure 3.2. Typically, the support region is 1–4 μm in length and width, as dictated by $\frac{\lambda}{NA}$. Therefore, using the SOCS approximation to compute intensity for a given sample point requires knowledge of the surrounding kernel support area on the mask. As already pointed out, the mask contour is the complete description of the mask, hence the contour within the support region is all that is required for image calculation at the sample point. The contour within a point's support region will be called the local contour for the given point. Figure 3.6 shows a mask area and then the local contour within the support region for the indicated sample point, (x_0, y_0). It will be shown that each edge of the local contour contributes a term to the linear shift-invariant system output at that point.

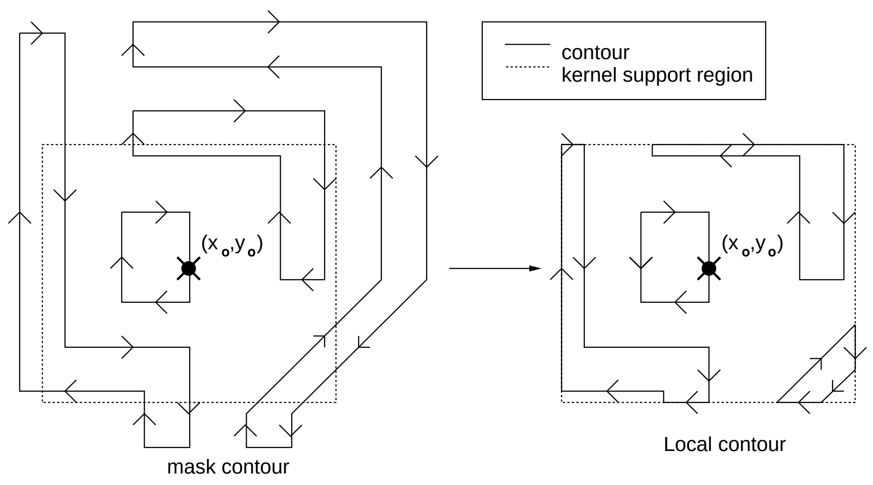


Figure 3.6: Local contour extracted from mask

3.2.3 Polygon convolution as "edge convolution"

The way in which edges are used for the convolution is inspired by a well-known area calculation algorithm for polygons: The area of a polygon can be computed by summing the area of trapezoids created by consecutive points on the polygon. We can define this trapezoidal partitioning within the support region, as shown in Figure 3.7. Each edge in the local contour implicitly defines a trapezoid extending to the base of the support region. The trapezoid is formed by an edge with a "curtain" of area draping down to the floor. An example is shown in Figure 3.7 where the polygon is broken down into several generated trapezoids, with each edge contributing to the total area separately through its generated trapezoid. The fact that we have correctly oriented closed contours ensures that the correct regions are "subtracted out" in the end, as shown in Figure 3.7. In this way, the mask function, M defined within the local contour is given by the superposition of trapezoid indicators, where each edge, $edge_k$, generates a trapezoid, $trap_k$:

$$M(x, y) = \sum_{k=1}^{N} \alpha_k \cdot t_k \cdot 1_{trap_k}$$

Here, $\alpha_k = +1$ for edges directed left to right and $\alpha_k = -1$ for edges directed right to left, and t_k is the transmission value for the interior part of the contour to which the edge belongs. Each α is chosen by taking the sign of the difference between the x-value of the first point of the edge and the second point of the edge, where the edge is directed from first point to second point.

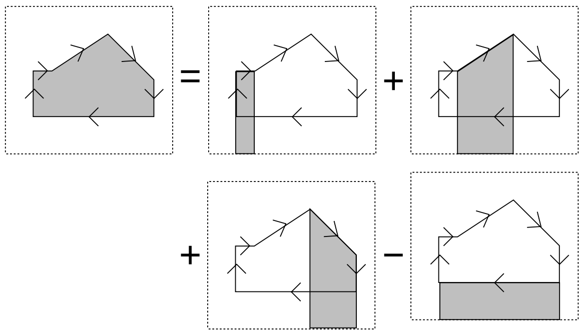


Figure 3.7: Partition of area into generated trapezoids from contour edges

The linearity of convolution allows the output of a linear system with kernel $h(x, y)$ to be expressed as a summation of contributions from the individual edges in the local contour via the generated trapezoids:

$$\begin{aligned} y(x_0, y_0) &= h \star M \\ &= h \star \sum_{k=1}^{N} \alpha_k \cdot t_k \cdot 1_{trap_k} \\ &= \sum_{k=1}^{N} \alpha_k \cdot t_k \cdot (h \star 1_{trap_k}) \end{aligned}$$

With that, the idea of computing intensity edge-wise on contours is complete. By itself, this is an interesting interpretation perhaps, but does not seem useful for fast calculation. After all, if the local contour has M_e edges, this method requires M_e convolutions to produce a value that could be obtained with one convolution on the sampled support region. The *structure* of this interpretation is valuable though, because with it and the linearity of the SOCS approximation, we can precompute all convolutions that we might encounter, so that intensity calculation becomes merely a *lookup operation*, as described in the next section.

3.2.4 Edge Lookup Tables

We have described a convolution method for arbitrary 2-D signal, h being convolved with area-wise constant functions such as a mask function, M. In this method, each polygon edge in the mask descriptions adds linearly to the result of the convolution. The method exploits the linearity of the SOCS approximation described in section 3.1. The linearity is further exploited by noting that the contribution of any possible edge can be precomputed and stored as a single complex number in a lookup table. By precomputing lookup tables, the objective of convolution calculation is achieved by one lookup per contour edge per kernel in the SOCS approximation. By linearity, all lookups for a given kernel are summed together to obtain the linear system output. Then, each of the linear system outputs are combined together to yield the intensity. What has just been described is an $O(N_a \cdot M_e)$ algorithm for intensity computation, where N_a is the number of kernels in the approximation and M_e is the number of edge in the local contour. Combining the outputs of the individual linear systems in the SOCS only requires $5N_a$ flops—3 for the complex magnitude squaring, 1 for scaling, and 1 for summing for each of the N_a kernels.

The only remaining issue is that of memory constraints. How much memory do the proposed lookup tables require? For a discretized domain, typically on a grid of $10nm$ and a kernel size of 1 μm, the addressable space is $(1/.01)^2 = 10,000$ points. Since an arbitrary edge has two points, it would require $10,000^2 = 10^8$ entries in the lookup table for each kernel. Clearly, this exceeds the limits of what can reasonably be stored.

The unreasonable storage requirement can be mitigated with two modifications to the lookup tables just described. First, we will have a separate lookup table for each edge angle that we wish to precompute. Second, we again use linearity to decompose each edge into two *ray edges* whose generated trapezoids are combined as shown in Figure 3.8. Now, only ray edges must be precomputed, and since each ray edge is addressable by a *single* coordinate point, as in Figure 3.9, the table size can be greatly reduced. It is noted that the edge based lookups are equivalent to a generalization of the “upper right corner rectangles”

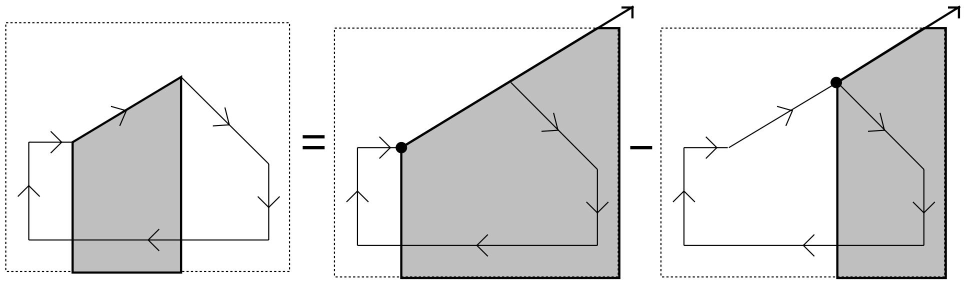


Figure 3.8: Linear combination of generated trapezoids

described in our previous work [7].

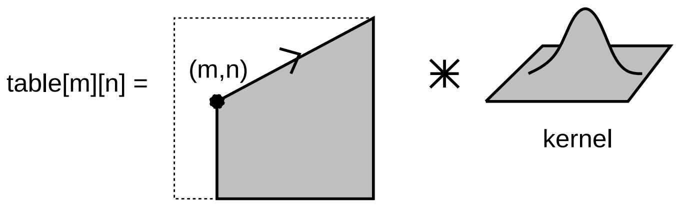


Figure 3.9: Lookup table entries are area indicator functions convolved with kernels

As an example for a 100×100 kernel, the lookup table for all angle 0° ray edges will require only 10,000 entries—one for each addressable point in the kernel support. For other angle edges, the same idea is applied. For a practical situation with 6 kernels and edges slopes of $\{0^\circ, 90^\circ, 45^\circ, -45^\circ\}$, the memory requirements sum to $10^4 \cdot 6 \cdot 4 = 2.4 \times 10^5$ complex numbers, which is small enough to fit into RAM.

An implementation using the lookup tables should merely default down to the slower mode of performing individual convolutions for the rare edges encountered on a mask which have not been stored in the lookup tables. Alternatively, lookup tables can be generated on-the-fly as different mask angles are encountered. In either case, the algorithm becomes fully general and can compute intensity values for an arbitrary mask.

3.2.5 Simulation of Mask Perturbations for OPC

The ability to perform fast, accurate intensity sample calculations has an enormous impact on the way optical proximity correction can be done. It allows the aerial intensity to be directly monitored during OPC, and therefore the model-based OPC scheme is possible.

Any sufficiently fast sparse intensity calculation algorithm grants this ability. The

SOCS structure, however, even goes beyond this requirement by allowing perturbations to existing intensity values. Due to its linearized structure, intensity sample points can be updated in response to mask perturbations at a minimal computational cost. Perturbational approachs have been used in mutual coherence formulations for defect analysis by Socha and Neureuther[37]; Here we use perturbations in the SOCS formulation for OPC.

Suppose that the intensities at specific control points in the image plane have been computed. The effect of a small rectangular perturbation caused by moving an edge or adding a serif on the mask during OPC can easily be included to "update" the intensity. In this way, OPC can keep a running tab on the intensities at control points and never need recompute the intensity. The illustration in Figure 2.10 is an example of a perturbation and the sites which require updating the aerial image.

To update a *single* sample point, a simple rectangular perturbation will require $8N_a$ flops to update the N_a linear system outputs—a complex addition for each of the 4 edges in the rectangle. Then, by the analysis given in section 3.2.4, the cost of updating the intensity is simply $5N_a$ flops for squaring, scaling and summing the linear system outputs. It is noted that with computational costs this minimal, implementation overhead overwhelms the time required for the updates themselves. The overhead is discussed more in section 3.3.

Another point about perturbations is that a single mask perturbation may require *many* intensity updates, corresponding to *all* sparse intensity samples located within the image disturbance area resulting from the mask perturbation. The disturbance area is the entire image region within half the kernel support length, L of the mask perturbation. The calculation of the disturbance area is a simple result of convolution, using the "overlap-add" interpretation: for any point greater than $L/2$ from the perturbation, the convolution kernel centered at the point will not overlap with the perturbation. Therefore, the perturbation has no effect on such points.

3.3 Performance Results and Discussion

In this section we discuss some implementation issues, show performance results for the technique, and examine the error of the SOCS by comparing to SPLAT.

3.3.1 Overhead

A performance consideration largely ignored up to this point is the overhead accrued in our fast intensity by edge lookup algorithm. This overhead is caused by calculation required for local contour extraction for a given point, as in Figure 3.6. We estimate the contour extraction overhead for intensity point calculation to be around 60–80% of the total computational cost and is subject to considerable improvement.

3.3.2 Sparse image intensity

Using the SOCS approximation to Hopkins in Section 3 yields a fast technique for aerial image intensity calculation using convolution implemented as lookup tables. The theoretical development in section 3.2.4 indicates an $O(M_e \cdot N_a)$ time dependence, where M_e is the number of edges in the local support region and N_a is the approximation order of the SOCS. Given this dependence, doubling the kernel support width would sweep in roughly 4X more edges and hence would approximately quadruple the computation time. This is relevant in the following data which reflects an approximation order of $N_a = 8$ and a 1.28 μm kernel support width.

Table 3.1 shows the intensity calculation speed[1] for several masks with varying geometry densities and types. The total number of sample points computed is indicated in the first column. Following that is the total computation time, computation speed, whether or not purely Manhattan geometry was present, the total number of edges, the average number of edges in the local contour of each point (essentially equivalent to edge density), and the size of layout itself. The two Manhattan patterns, mask 1 and mask 2, are the most dense. The speed for calculation on mask 2 is slower than for mask 1 because mask 2 is more dense than mask 1. Because we have only computed lookup tables for angles in the set $\{0°, 90°, 45°, -45°\}$, the algorithm performs up to an order of magnitude slower for non-Manhattan masks than for Manhattan or 45° masks. For the non-Manhattan patterns, direct comparison of the speed is not possible. This is caused by the fact that the densities of non-Manhattan versus Manhattan edges on each mask plays a role in the overall speed. So, although mask 4 is non-Manhattan, it has relatively fewer angled edges than mask 3, and therefore a faster computation speed. We can summarize the results by stating that sparse intensity calculation speed ranges from 1,000–11,000 pts/sec for our examples, depending

[1]Speed on Sparc 10 platform

on edge density and whether non-Manhattan edges are present.

	# points	size (μm^2)	tot edges	avg M_e	time	speed ($\frac{pts}{sec}$)	Mhtn
mask 1	18,964	15 ×24	492	37	1.68 sec	11,280	X
mask 2	13,560	11 ×18	492	50	1.63 sec	8,320	X
mask 3	14,388	23 ×25	483	33	9.2 sec	1,560	O
mask 4	78,056	172 ×256	16824	18	8.4 sec	9,290	O

Table 3.1: Sparse image intensity calculation times

3.3.3 Image intensity perturbation

The least common denominator of our sparse intensity calculation method is the perturbation time, in which we simply add a single rectangle to the mask and update the intensity at all points which were perturbed due to the rectangle. The time required to update a single intensity point in response to a single perturbation does not depend on the figure density, and it is therefore a good candidate for a benchmarking figure.

The perturbation time is the most relevant time in the context of our simulation feedback OPC, where over 90% of the time is spent adding perturbations. Theoretically, the perturbation time has a linear dependence on approximation order N_a. In the graph of Figure 3.10, we plot the perturbation time while varying N_a and indeed witness the predicted linear dependence. For this experiment, we perform updates at 2.9 million points by perturbing the mask at various locations. This makes the perturbation speed for $N_a = 8$ clock in at 51,000 pts/sec. The plot has a y-intercept at around 21 seconds, meaning that 21 seconds out of the total time were spent on implementation overhead. Improvements in implementation could improve the speed by reducing this number and thereby make the algorithm approach the theoretical minimum computation bound of $13 \cdot N_a$ flops discussed in section 3.2.5.

3.3.4 Speed

As an example of the algorithm performance, we calculated the intensity at selected sample points on two test masks. We also perturbed the test masks at various locations, updating the intensity sample values using perturbational updates. The first test mask,

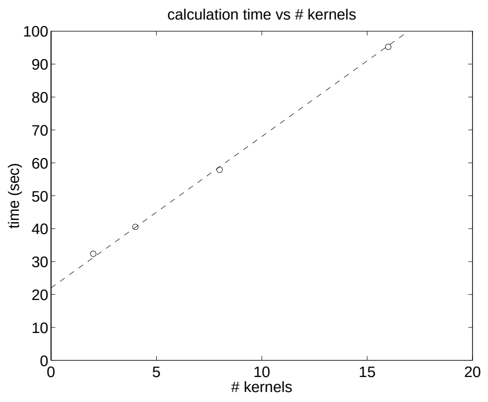


Figure 3.10: Perturbation time versus approximation order N_a

`sp1`, is shown in Figure 3.11. The sparse sample points on this mask are shown at the circled locations in Figure 3.12. The second test mask, `tp7`, is the non-Manhattan mask shown in Figure 3.4, with sample points similarly chosen along the feature boundaries. Table 3.2 summarizes the performance results for calculating sparse intensity values. The table reflects a 6 kernel SOCS approximation with kernel support size of 1.28 μm in length, sampled on a 10 nm grid. The optical system approximated by the SOCS has $\lambda = 0.365$, NA $= 0.5$, and $\sigma = 0.6$. The lookup tables each occupy 131 KBytes, and there are 24 tables total to handle angles in the set $\{0°, 90°, 45°, -45°\}$ for each of the 6 kernels. The computation time to generate the lookup tables is a few seconds per table, but of course once generated, the tables are saved on disk and reused. All run times are relative to a Sun SPARC 10 workstation.

Table 3.2 shows a faster point computation speed for `sp1` because the pattern is much less dense in polygons, hence the number of edges in a given support region is smaller. It is noted that these speeds are subject to improvement via speedup of the contour extraction algorithm.

mask	intensity pts	time	speed
`sp1`	4299	27 sec	6.3 msec/pt
`tp7`	2454	23 sec	9.4 msec/pt

Table 3.2: Intensity point computation speed

The mask perturbations were performed by moving along the contour of the mask, and moving each edge segment on the contour by 20 nm. The average number of points affected by a single mask perturbation is listed in Table 3.3. The average time per mask perturbation indicates the time for updating all intensity points affected by the perturbation and is listed as a separate entry than the update time per intensity point. The higher density of polygons in `tp7` is reflected by a higher density of control points, in column 3. Rectangular perturbations were optimized in code for speed, hence the faster times for `sp1`.

mask	mask perturbations	av. pts per perturbation	mask perturb. speed	intensity update speed
`sp1`	2866	34.6	930 μsec/pert.	26 μsec/pt
`tp7`	1176	55.8	3960 μsec/pert.	71 μsec/pt

Table 3.3: Intensity perturbation speed

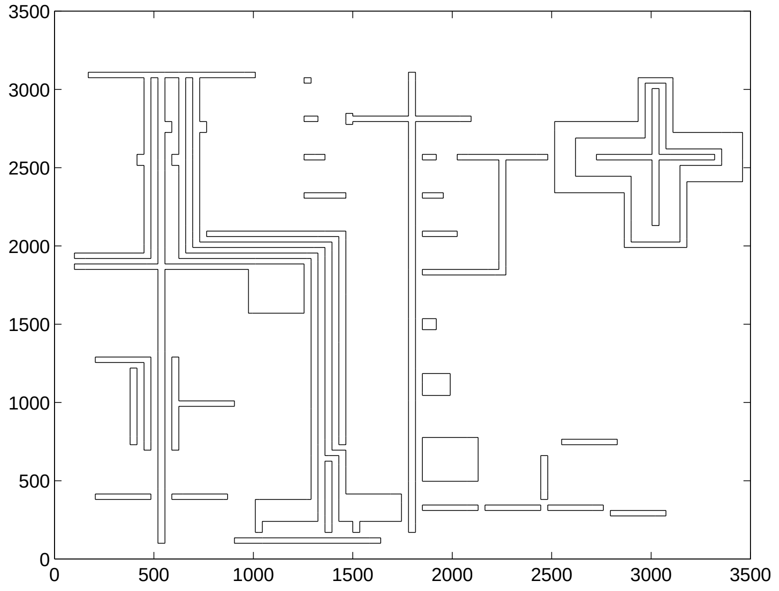


Figure 3.11: Test mask `sp1`

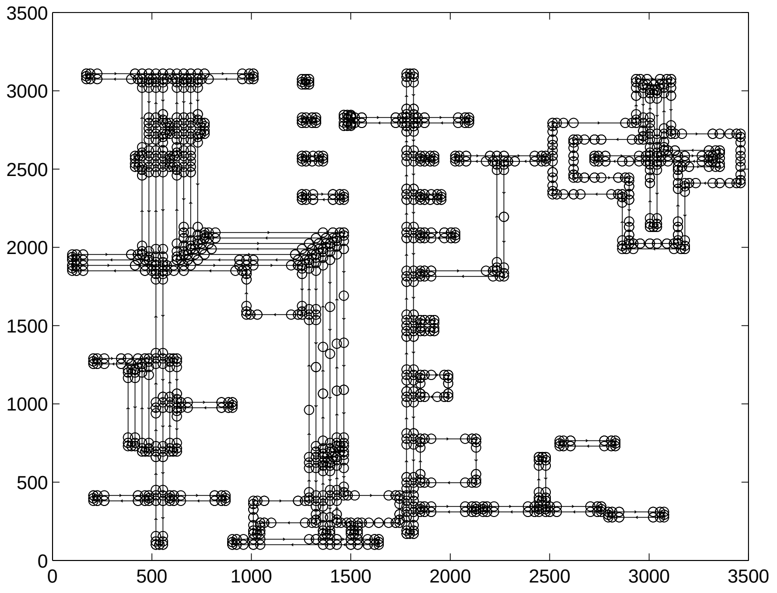


Figure 3.12: Sparse sample point locations in `sp1`

Chapter 4

Sum of Coherent Systems Decomposition

4.1 Introduction

In this section, the Hopkins [19] partially coherent imaging equation is expanded by eigenfunctions. The result is a bank of linear shift-invariant systems whose outputs are squared, scaled and summed. This technique is useful because of the partial linearity of the resulting system approximation. The eigenfunction expansion can be performed numerically using the SVD (Singular Value Decomposition) algorithm. To solve this problem, the Hopkins transmission cross coefficients (TCCs) are first obtained as a matrix, then SVD is used on the matrix. Then the system is truncated at a low order, to obtain an optimal approximation to Hopkins. In effect, the numerical implementation of this using TCCs generated by the SPLAT aerial imaging program [12, 42, 43] amounts to a direct approximation to SPLAT.

Background The use of decompositions to Hopkins imaging equations has an extensive history. In some of the earliest work in this area, Gamo [14] developes technique a matrix treatment of partial coherence. From Wolf [46], a "spectral representation" is described, using Fourier decomposition. Other work by Saleh[36] describes a "modal expansion" involving a similar decomposition. Rieger and Stirniman [38] cite Gamo [14] and describe a "zone sampling" technique. Interesting results have been shown by Bunau [44], in which closed form expressions for the convolution kernels are derived, calling them "Optimal Coherent Approximations", or OCA's. The OCA name is also used by Pati and Kailath [34] in describing how a Mercer expansion can yield the OCA's.

4.2 The Idea

With this section, we will clarify the derivation and implementation of a discrete Hopkins expansion which results if the transmission cross coeficient function can be expressed as a tensor product. To begin, consider the continuous 1-D Hopkins imaging equations [19, 1, 12]:

$$I(f) = \int T(f+f', f')G(f+f')G^*(f')df' \tag{4.1}$$

$$T(f', f'') = \int J_0^-(f)K(f+f')K^*(f+f'')df \tag{4.2}$$

where $T(\cdot,\cdot)$ are the transmission cross coefficients, $G(\cdot)$ is the mask Fourier transform, and $I(\cdot)$ is the intensity function Fourier transform. The function $K(\cdot,\cdot)$ is the coherent transmission function which can be written for no aberrations as:

$$K(f,g) = \begin{cases} exp\left(\frac{2\pi i}{\lambda}\frac{1}{2}\Delta z \lambda^2 (f^2+g^2)\right) & f^2+g^2 < \frac{NA}{\lambda M}^2 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

The function $J_0^-(\cdot,\cdot)$ is the mutual intensity function represented in the frequency domain:

$$J_0^-(f,g) = \begin{cases} \frac{\lambda^2}{\pi s^2 NA^2} & f^2+g^2 < \frac{s^2 NA^2}{\lambda^2} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Suppose that $T(\cdot,\cdot)$ can be expressed as

$$T(f', f'') = T_1(f')T_1^*(f'') \tag{4.5}$$

then the Hopkins equations can be simplified as follows:

$$I(f) = \int T_1(f+f')G(f+f')T_1^*(f')G^*(f')df' \tag{4.6}$$

This is recognized as convolution in the frequency domain, which is rewritten using the convolution operator, $\star$, as

$$I = (T_1 G) \star (T_1 G)^* \tag{4.7}$$

So, back in the spatial domain, the intensity as a function of position can be expressed as

$$i(x) = |(t_1 \star g)(x)|^2 \tag{4.8}$$

where lowercase letters denote the corresponding spatial domain functions. The reason for performing this analysis is that the expression (4.8) is much less computationally demanding than (4.1). Intensity in (4.8) can be computed in on the order of an FFT operation whereas (4.1) requires a convolution-type operation and an FFT. The convolution-type operation is much more demanding than the FFT so we hope to save a considerable amount of unneccessary work *if* we can use this decomposition. It turns out that the decomposition just outlined is *not* true for partially coherent optics, but a very similiar one does hold.

4.3 Hopkins Imaging Equation

By periodicizing the length L_x mask, and taking its Fourier series expansion $\tilde{G}(n)$, as done by Flanner [12] we obtain the following expression for the Fourier series of the resulting periodic intensity, $\tilde{I}(\cdot)$:

$$\tilde{I}(n) = \sum_{n'} \tilde{T}(n+n', n')\tilde{G}(n+n')G^*(n') \tag{4.9}$$

where $\tilde{T}(n', n'') = T(\frac{n'}{L_x}, \frac{n''}{L_x})$. We will be working with this discretized frequency domain formulation throughout this section.

4.3.1 Preliminary results

Claim 1 *$\tilde{T}$ is bandlimited and therefore can be represented by a matrix of values.*

Proof:
In the formulation of $T(\cdot,\cdot)$, the following relations hold

$$K(f) = \begin{cases} \exp \frac{\imath 2\pi}{\lambda} \frac{1}{2} \Delta z NA^2 (f^2) & |f| < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$J_0^-(f) = \begin{cases} \frac{1}{\pi s^2} & |f| < s \\ 0 & \text{otherwise} \end{cases}$$

In the current 1-D problem these two function have support described by a line segment and in 2-D the support region is a circle. Whenever the supports of the three functions in the TCC integral eq (4.2) have no overlap, the TCC is zero. This is true under the following conditions, where $f_{max} = 1 + s$,

$$T(f', f'') = 0, \quad |f'| > f_{max} \quad \text{or} \quad |f''| > f_{max}$$

which can be verified by a simple sketch. Therefore, the discretized $\tilde{T}(n', n'')$ obtained by sampling will be zero for $\frac{n'}{L_x} > f_{max}$ or $\frac{n''}{L_x} > f_{max}$. Let the maximum n' such that $\frac{n'}{L_x} <= f_{max}$ be called N_{max}. Then the matrix of TCC values is $(2N_{max} + 1) \times (2N_{max} + 1)$ in size. □

Claim 2 $\tilde{T}(n', n'') = \tilde{T}^*(n'', n')$ *and therefore it is Hermitian symmetric.*

Proof:

$$\begin{aligned} T^*(f'', f') &= \int J_0^{-*}(f) K^*(f + f'') K(f + f') df \\ &= T(f', f'') \qquad\qquad \text{since } J_0^-(\cdot) \text{ is real.} \end{aligned}$$

□

Claim 3 $\tilde{T}(n', n'')$ *can be written as*

$$\tilde{T}(n', n'') = \sum_{m=1}^{2*N_{max}+1} \lambda_m \Phi_m(n') \Phi_m^*(n'')$$

Proof:

Putting Claims 1 and 2 together allow us to write $\tilde{T}(n', n'')$ as the Hermitian matrix $[\tilde{T}]_{i,j} = \tilde{T}(i, j)$. The dyadic expansion of T in terms of it's eigenvectors is

$$\tilde{T} = \sum_{m=1}^{2*N_{max}+1} \lambda_m \Phi_m \Phi_m^*, \tag{4.10}$$

which is equivalent to the claim. □

4.3.2 Main result

Now, going back to the original problem with the results of claim 3, substituting 4.10 into 4.9, it is possible to write the frequency domain expression:

$$\begin{aligned} \tilde{I}(n) &= \sum_{n'} \sum_{m=1}^{2*N_{max}+1} \lambda_m \Phi_m(n + n') \Phi_m^*(n') \tilde{G}(n + n') G^*(n') &(4.11) \\ &= \sum_{m=1}^{2*N_{max}+1} \lambda_m ((\Phi_m \tilde{G}) \star (\Phi_m \tilde{G})^*)(n) &(4.12) \end{aligned}$$

where the simplication is made by exchanging the order of the summations and recognizing the resulting inner summation as convolution. This can be written in the time domain as

$$\tilde{i}(x) \quad = \quad \sum_{m=1}^{2*n_{max}+1} \lambda_m \left|(\phi_m \star \tilde{g})(x)\right|^2 \tag{4.13}$$

With this we have a Sum-Of-Coherent Systems (SOCS) which describes the action of the partially coherent optical system, as shown in Figure 4.1.

4.3.3 Optimal approximation using truncation

If the expression 4.13 is truncated, we obtain the following approximation to $\tilde{i}$, which will be called $\hat{\tilde{i}}$.

$$\hat{\tilde{i}}(x) \quad = \quad \sum_{m=1}^{N} \lambda_m \left|(\phi_m \star \tilde{g})(x)\right|^2 \tag{4.14}$$

By truncating the summation we get a reduced order approximation to the partially coherent imaging system. Since the eigenvalues decay rapidly in magnitude, the truncation will be a good approximation. The truncation is optimal in the sense that $\|\hat{\tilde{i}} - \tilde{i}\|$ is minimized for the given approximation order.

The error bound on the the truncation is simply:

$$e \quad \leq \quad \sum_{m=N+1}^{2*n_{max}+1} \lambda_m^2 \tag{4.15}$$

which holds because the kernels ϕ have unit norm.

4.4 Interpreting the results

The derivation in section 4.3 shows the 1-D SOCS decomposition of Hopkins aerial image intensity equations. This derivation was motivated by the desire to use convolution to represent the aerial image computation. Computationally, the decomposition can be performed by SVD if we have the computed TCC values, $\tilde{T}$. In this case, we are finding the eigenvectors of the TCC matrix.

In terms of a system view of the model, Figure 4.1 illustrates how aerial image computation has been transformed into convolution operations.

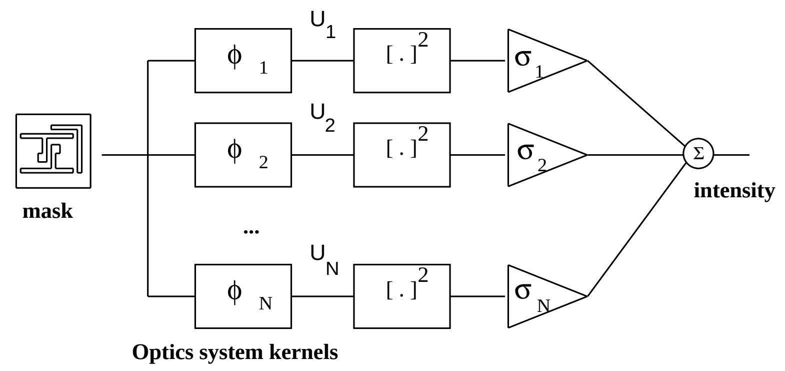


Figure 4.1: Sum of coherent systems approximation to Hopkins imaging

4.5 2-D Kernel Determination

In the 2-D case, we can also use SVD to obtain the 2-D convolution kernels in the decomposition. In the 2-D case, the discrete TCCs can be thought of as a linear mapping from the space of $M \times M$ matrices, $R^{M\times M}$, to itself. So, given a matrix $X \in R^{M\times M}$, the function $\tilde{T}(i,j,k,l)$ can define a linear mapping T whose action is described by:

$$[T(X)]_{(i,j)} = \sum_{k=1}^{M}\sum_{l=1}^{M}\tilde{T}(i,j,k,l)X_{k,l}$$

This linear operation can be "unwound" to be represented as a matrix, $\mathcal{T}$, operating on a column vector from R^{N^2}. The unwinding is performed by stacking the columns of the vector and then writing the operation out as a matrix-vector multiply. The column stacking function $\mathcal{S} : R^{N\times N} \longmapsto R^{N^2}$ is defined by:

$$\mathcal{S}(X_{i,j}) = \mathcal{X}_{j*N+i}$$

For example, let X be a matrix of size $M \times M$:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1M} \\ x_{21} & x_{22} & \dots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{M2} & \dots & x_{MM} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_M \end{bmatrix}$$

The column stacking operation is performed on X, yielding

$$\mathcal{X} = \mathcal{S}(X) = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{bmatrix}$$

.

The operator T defined before can be represented as a matrix, $\mathcal{T}$, operating on the stacked $\mathcal{X}$ vector,

$$\mathcal{T} = \begin{bmatrix} \tilde{T}(1,1,1,1) & \tilde{T}(1,1,2,1) & \dots & \tilde{T}(1,1,N,1) & \tilde{T}(1,1,1,2) & \dots & \tilde{T}(1,1,N,N) \\ \tilde{T}(2,1,1,1) & \ddots & & & & & \\ \vdots & & & & & & \\ \tilde{T}(N,1,1,1) & & & & & & \\ \tilde{T}(1,2,1,1) & & & & & & \\ \tilde{T}(2,2,1,1) & & & & & & \\ \vdots & & & & & & \\ \tilde{T}(N,2,1,1) & & & & & & \\ \vdots & & & & & & \\ \tilde{T}(N,N,1,1) & \dots & & & & & \tilde{T}(N,N,N,N) \end{bmatrix}$$

The contour plot in Figure 4.2 shows the an example of the matrix $\mathcal{T}$. Singular value decomposition applied to this matrix yields the decomposition:

$$\mathcal{T} = \sum_{k=1}^{N} \sigma_k V_k V_k^*, \tag{4.16}$$

Then, the inverse column stacking operation yields the desired functions, Φ_k

$$\Phi_k = \mathcal{S}^{-1}(V_k)$$

and then it it possible to make the approximation:

$$\tilde{T}(n', n'') \approx \sum_{k=1}^{N_a} \sigma_k \Phi_k(n') \Phi_k^*(n'')$$

The spatial domain convolution kernels are the Inverse Fourier Series (IFS) of the Φ_k's. Using the 2-D Inverse Fast Fourier Transform (IFFT) to obtain the ϕ_k's yields:

$$I(x,y) = \sum_{k=1}^{N_a} \sigma_k \left|(\phi_k \star g)(x,y)\right|^2 \tag{4.17}$$

Using the IFFT instead of IFS introduces a small amount of aliasing, but this is necessary in order to limit the time domain convolution kernels to have finite support. A plot of the singular values in Figure 4.3 shows why a reduced order approximation can be very accurate, since the singular values quickly approach zero. The magnitudes plots of the first two convolution kernels obtained after the IFFT are shown in Figure 4.4.

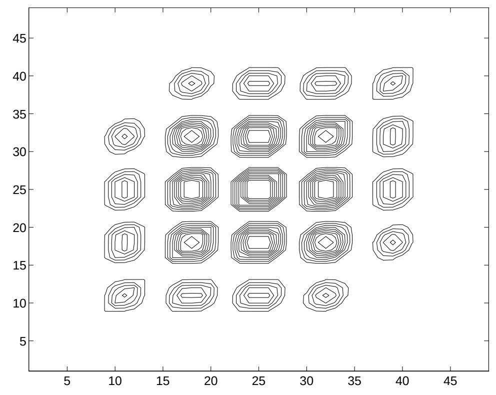


Figure 4.2: Contour plot of TCCs in 2-D matrix form

4.6 Error analysis

As previously mentioned, the error due to truncation to a low order is nicely bounded because the properties of the SVD. An error bound for a single intensity point can be written:

$$e \leq \sum_{m=N+1}^{2*n_{max}+1} \lambda_m^2 \tag{4.18}$$

We will use the theoretical error bound as a reference. As an experiment, computed intensity values from the SOCS approximation and from the output of the SPLAT aerial image program are compared to see how the error compares to the error bound.

We show results of simulation experiments comparing the intensity obtained from the our proposed technique to the intensity given by SPLAT. The simulations were performed on the mask in Figure 3.11 at sparse points along the mask contour. As before, the optical system approximated by the 6 kernel SOCS has $\lambda = 0.365$, NA $= 0.5$, and $\sigma = 0.6$.

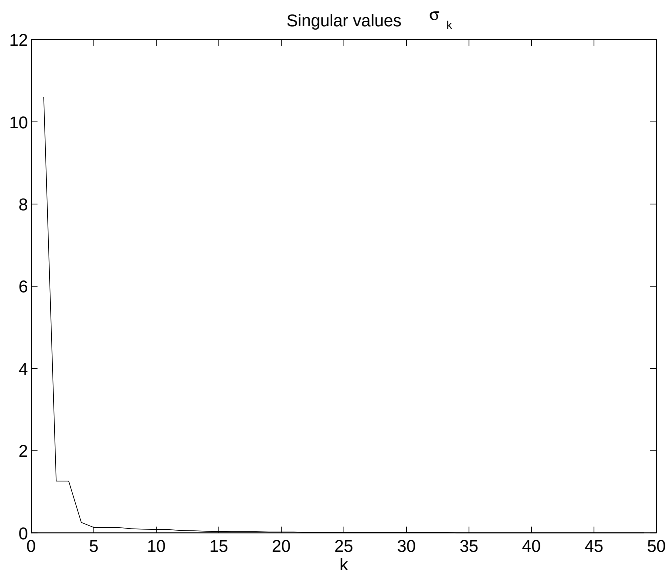


Figure 4.3: Singular values, σ_k obtained in decomposition for lambda = 365 nm, NA = 0.5, and $\sigma = 0.6$

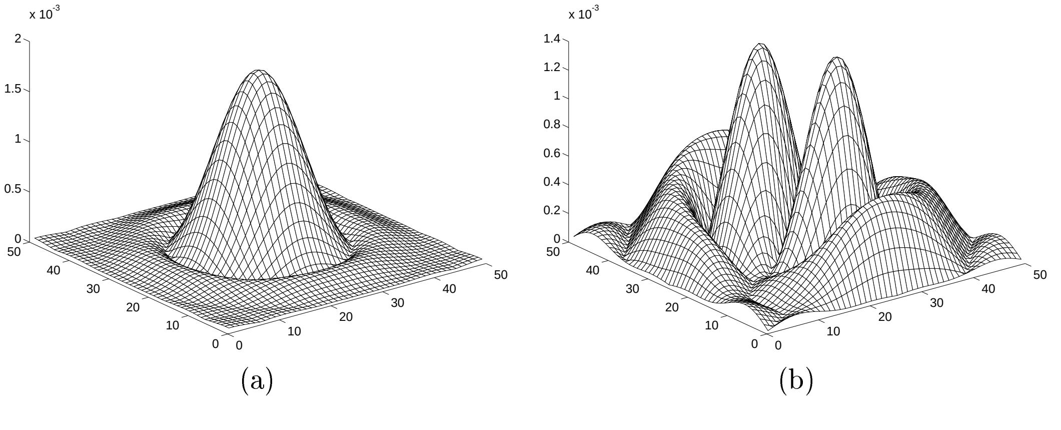


Figure 4.4: (a) $\phi_1(x,y)$ (b)$\phi_2(x,y)$ for lambda = 365 nm, NA = 0.5, and $\sigma = 0.6$

The error plot in Figure 4.5 shows the *relative error* for each of the sample points, where the relative error is defined as the error normalized to the correct SPLAT value:

$$e = \frac{|x - \hat{x}|}{x}$$

The plot indicates a relative error bounded by 1.5% for the chosen sample points.

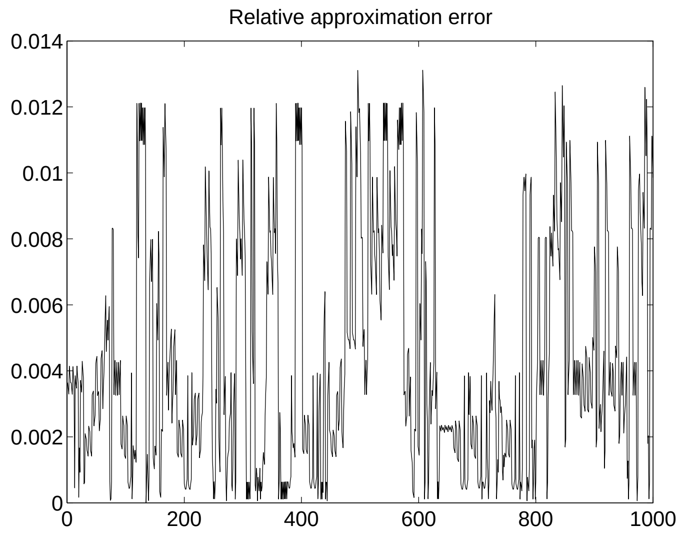


Figure 4.5: Relative approximation error for chosen sample points

Chapter 5

Process models

The topic of this Chapter is the use of empirically determined fast process simulation models in OPC. The simulation models used in OPC must be "tunable" to a given process to ensure validity and accurate OPC results. Model-based OPC relies heavily on having accurate models to compensate for non-idealities in pattern transfer. An effective way of ensuring modeling accuracy is to choose a physically motivated functional form for a model, and then determine the model parameters by experiment. On the other hand, a goal which at times conflicts with thorough modeling accuracy is the need for extremely fast models. A practical OPC solution must balance these conflicting requirements.

To summarize, the process model requirements are:

- Accuracy
- High speed
- Empirically determined or tunable

5.1 Modeling philosophy

The modeling philosophy is that distinct physical processes which introduce significant systematic error should be characterized with distinct models. Purely random variations in a process are not modeled, but viewed as noise.

Black box modeling The most abstract way to view the entire pattern transfer process is as a giant "black box" encompassing all process steps. For lithography, the input is a layout design, and the output is wafer structures. This is depicted in Figure 5.1. In this approach, the model can be determined by measuring the output for given inputs and fitting this to model parameters.

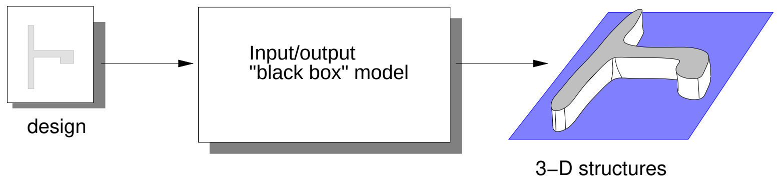


Figure 5.1: Black box model of pattern transfer.

Informed modeling Existing knowledge that we have about the lithography process allows us to refine the model considerably, and therefore perform much more accurate modeling. Since we know there is diffraction optics operating inside the black box, it makes sense to insert that into our model. Likewise, it makes sense to model the resist development and the etch since we *know* that these physical processes are occuring in lithography. A "fleshed-out" version of the black box model is depicted in Figure 5.2. This model has a number of sub-systems which model various distinct physical processes. The diagram is by no means complete. Any phenomena which can be modeling distinctly can be inserted into the simulation model.

Our proposed OPC system architecture from Chapter 2 allows future model improvements as may be needed. This is possible because new models can be "plugged in" to the existing edge movement and optimization engine.

Observable data As seen in Figure 5.2, any experimentally observable data can be used in the modeling. Practically speaking, observable data is anything that can be experimentally measured, including:

- mask CD
- resist CD
- wafer CD

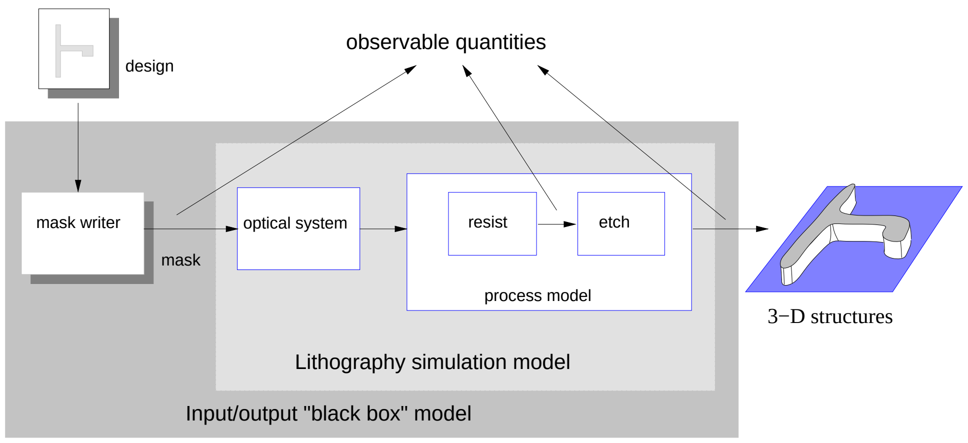


Figure 5.2: Filled-in model, with physical processes clearly demarcated.

5.2 Aerial image models

The aerial image is modeled accurately by the Hopkins equations described in Chapter 3. The use of the Hopkins model provides a very general way of computing scalar aerial imaging.

5.3 Defocus model

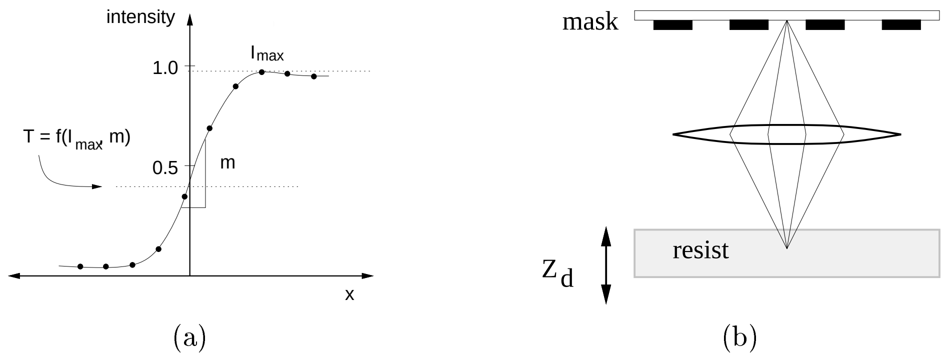


Figure 5.3: (a) 1-D cutline across line transition (b) resist thickness

The 1-D intensity cutline as depicted in Figure 5.3(a) is used to determine the edge placement. The intensity cutline is averaged over some z-direction thickness as depicted in Figure 5.3(b). The averaging technique is designed to capture information about the average exposure energy in resist as described by Garafalo *et al.* [15]. In their work, they show that averaging the image intensity is equivalent to averaging the TCC function from the Hopkins model. The averaged TCC function is given by:

$$TCC_{av} = \frac{1}{Z_d}\int_{z=0}^{Z_d} TCC(z)\,dz \tag{5.1}$$

In this way, a portion of the resist modeling can be absorbed into the imaging model. Conceptually this is equivalent to shrinking the resist itself to a single plane which receives the averaged exposure.

The model requires a single parameter, the resist thickness, Z_d, which is known. Therefore, no experimental tuning is required. Another benefit of this model is that only a single aerial image is computed, rather than the aerial image at several different planes. Furthermore, the averaged TCC function can be represented using the SOCS decomposition to allow the fast aerial image computation techniques previously described.

5.4 VTR Resist Model

We use a physically motivated, simplified resist model with fast aerial image simulation as a basis for a fast lithography simulation system. The resist model is not an attempt at a general model such as the models of Dill[11] or Kim[21, 41], but rather it is a highly specific model designed to match a given lithography process. The spirit of this model is more in line with other fast, simplified resist models.

The Brunner model[3, 4] uses image intensity and image log-slope of the to calculate the resist development path for a given dose. As with the VTR model, this model depends highly on the maximum local intensity and the image slope.

The Mack model[18, 29] also uses a simplified development model to predict the resist profile. In this approach, the resist development is "segmented" into a vertical development component and a horizontal development component. In this technique, the exposure and the resist contrast parameter are important variables.

Garafalo *et al* [15] describe a technique for modeling to match with experimental data which uses a Gaussian convolution with the aerial image followed by a constant threshold. This model accounts for diffusion effects and has been shown to model experimental data with good accuracy.

Our model is called the variable threshold resist (VTR) model because it uses an data-dependent threshold to determine at which normalized light intensity the printed wafer edge will appear. The model depends on empirical data in the form of linewidth measurements. The models reflect an inherent view of the final structures as 2-D top-down images which can be characterized in terms of linewidth and CD measurements.

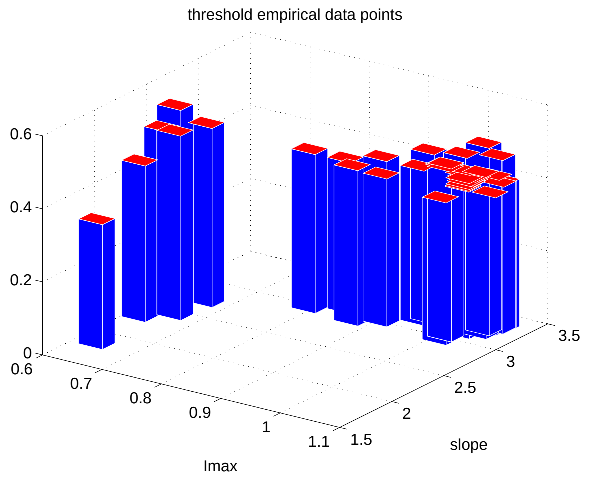


Figure 5.4: Empirical threshold values determined from linewidth measurements.

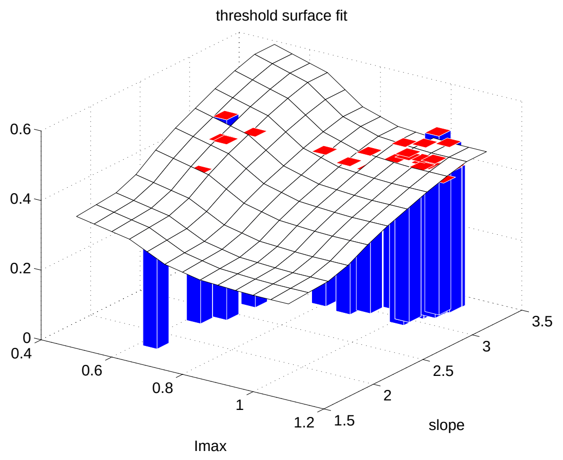


Figure 5.5: Surface fit described by VTR model superimposed on empirical data.

We assume that the nonlinear process which maps impinging light energy to the edge placements and CDs of resulting structure is a thresholding type model. To give the model dynamics, it is allowed to be an image-dependent threshold. The assumption formulates the threshold as a function of the maximum value of the intensity cutline and its slope. In particular we make the assumption that the threshold, T, can be expressed as:

$$\begin{aligned} T &= f(I_{max}, m) && (5.2) \\ &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} a_{i,j} I_{max}^{i} m^{j} && (5.3) \end{aligned}$$

where I_{max} is the peak value of the intensity cutline and m is the maximum slope of the intensity cutline. The variables in the model are the parameters $a_{i,j}$ which are fit using empirical data.

5.4.1 Empirical determination of model parameters

To build a empirical model, first experimental data must be gathered about the process in question. This data takes the form of CD measurements of the printed linewidths versus the desired linewidths. The data is then used to fit a model. The diagram in Figure 5.6 shows the basic steps involved in obtaining empirical data for model building.

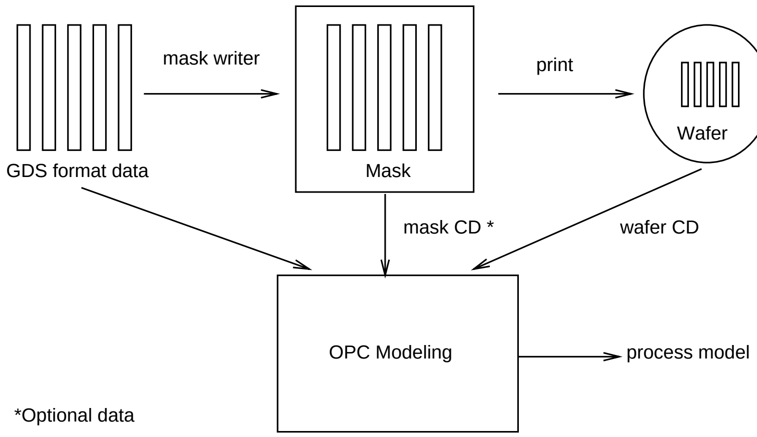


Figure 5.6: Basic OPC test pattern data flow.

The empirical data needed to fit the VTR parameters should include measurements of specific types of 1-D and 2-D structures which capture the maximum amount of information about the process. A minimal set of such structures can be called the basis geometries.

Basis geometries in the test pattern

Printing and measuring CDs on the basis set of geometries provides enough information to characterize a process. Using the process model, OPC can be performed on any geometries. The basis geometries have symmetric structure so that accurate measurement and modeling is possible.

A description of the different types of structures in the basis geometries is given in Appendix B.

5.4.2 Example 1

To model an i-line process, we use empirical data in the form of linewidth measurements. The process parameters are given below:

Summary	
lambda	0.365
NA	0.50
sigma	0.60
resist thickness	1.1 μm

Specifically, linewidths of 3D structures after resist development and etch are first plotted as a function of maximum image intensity and slope. An empirical plot of T versus I_{max} and m is shown in Figure 5.4 for 39 measurement points. The empirical data is fit to the model of equation 5.4. In fitting the model with a 3 $\times$2 polynomial surface, we obtain a complete characterization of the surface as shown in Figure 5.5.

The accuracy of the overall process simulation is measured by comparing empirical linewidth measurements to those predicted by simulation. For a particular i-line process, we have done this comparison. The plots in Figures 5.7 and 5.8 show the empirical edge placement error (EPE), which is determined *for symmetric structures* using the equation

$$EPE = \frac{1}{2}(\text{target linewidth - measured linewidth})$$

compared to the simulated EPE. In Figure 5.7, we see the accuracy over a range of empirical EPE values and in Figure 5.8, the information is condensed into a histogram showing (simulated EPE - empirical EPE). From the histogram, we see that the model predicts the EPE with a 10.4 nm standard deviation for 39 empirical data points.

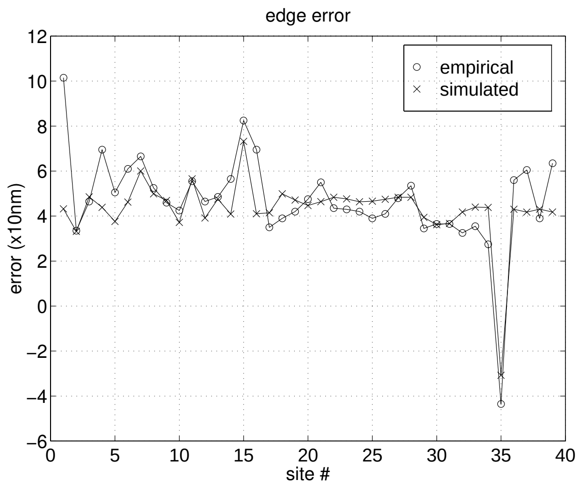


Figure 5.7: Measured and simulated edge placement errors in 10 nm units

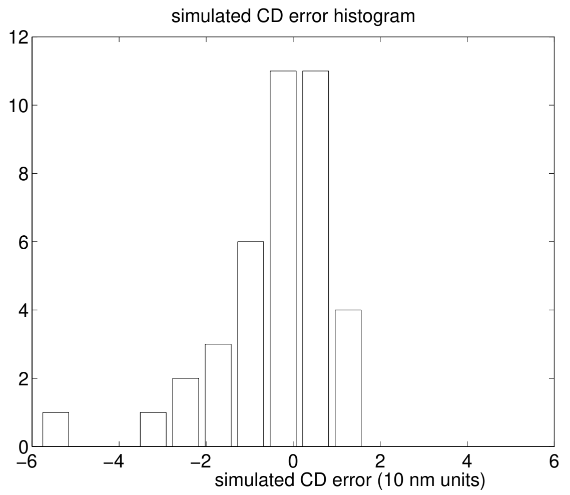


Figure 5.8: Difference between simulated and measured edge placement errors in 10 nm units

5.4.3 Example 2

Figure 5.9 shows an example of empirical linewidth measurements modeled by the VTR model and by a simple constant threshold resist (CTR) model for an i-line case. The uniformity plot is taken from Intel data published by Borodovsky[2]. The linewidth measurements were obtained from variable pitch line/space patterns with the pitch ranging from dense to isolated. A total of 22 resist measurements were used in creating this VTR model.

Summary	
lambda	0.365
NA	0.57
sigma	0.60
resist thickness	(use best focus) μm

For this example, the model in equation 5.4 has $N = 2, M = 2$ so that four $a_{i,j}$ parameters must be determined. The model is observed to fit the data at both dense and isolated pitches. In contrast, the constant threshold model fails for dense lines.

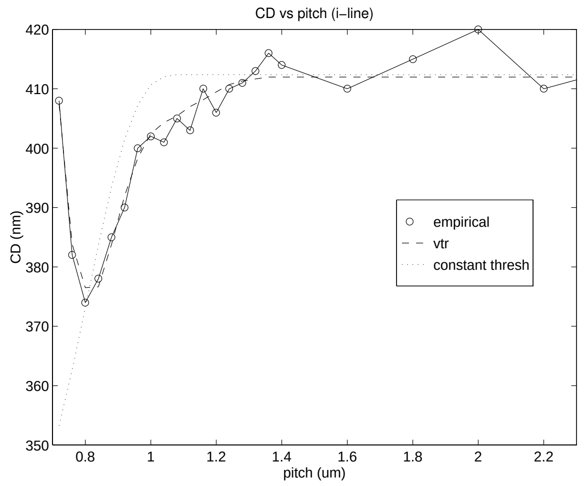


Figure 5.9: VTR model and constant threshold model fit to empirical CD data for i-line example.

5.4.4 Example 3

Figure 5.10 corresponds to 0.30 μm line space pattern using DUV exposure with conditions as follows:

Summary	
lambda	0.248
NA	0.45
sigma	0.60
resist thickness	(use best focus) μm

In the modeling, 13 data points were used to generate the VTR model. In this case, $N = 2, M = 2$ for the modeling fit. Again, the VTR model fits the data well for the range from isolated to dense lines.

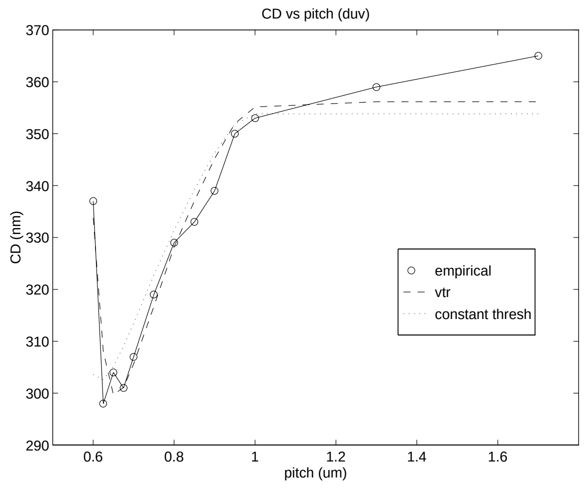


Figure 5.10: VTR model and constant threshold model fit to empirical CD data for DUV example.

5.5 Simulation performance

The goal of developing our simulation models has been to achieve high speed simulation suitable for OPC. Putting the imaging and resist systems together produces a fast process simulation system which takes a mask as input and yields a simulated top-down SEM view of the final structures. Here is an example of the overall process simulation. The tests were run on an HP 700 series workstation. Simulated features resulting from an i-line process in the 48 $\times$27 μm^2 wafer area shown in Figure 5.11 were simulated in 12.5 seconds with our simulation system by combining sparse intensity sampling with the resist model. The vast majority of this time is spent doing intensity calculation and a small fraction is spent on the resist calculation.

5.6 Conclusions

We have proposed a particular structure for OPC which we call *simulation feedback optimization* which achieves accuracy by monitoring edge placements during OPC by the use of simulation. To propel the OPC system into practical use for large designs, we have developed high speed sparse aerial image intensity calculation and simplified resist models. The intensity calculation is performed by applying a SOCS to Hopkins imaging model. Lookup tables implement the convolution operation for general mask geometry, resulting in a fast, highly accurate imaging tool. Image point calculation speeds up to 11,000 points/sec and perturbation speeds of 51,000 points/sec were demonstrated on an HP700 workstation. The physically based resist model we use can be fit to empirical measurements. Together, the simulation systems comprise a lithography process simulator which can be used to produce "simulated SEMs", or simulated etched structure contours as seen from a top view. For a particular set of empirical data, we achieved linewidth prediction accuracy of 10 nm from our overall simulation system. Integrating the simulation tools back into the OPC system allows correction of a 48×27 μm^2 area at 96 seconds/iteration using 6-10 iterations with expected accuracy of wafer features resulting from OPC on the order of the simulation model accuracy.

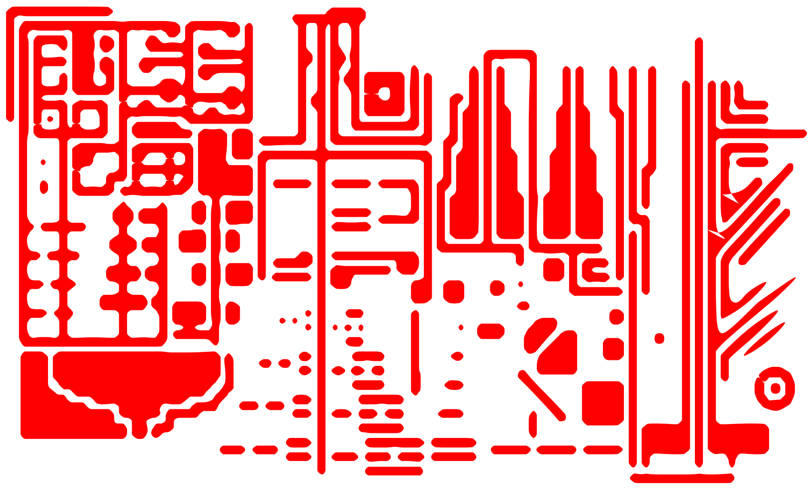


Figure 5.11: Simulated printed features on wafer (Mask image reproduced with permission of SEMATECH)

Chapter 6

Simulation Results Using OPC

In this chapter, the performance of the OPC algorithm described in Chapter 2 is evaluated by performing OPC on trial masks. The qualities under inspection are effectiveness of the OPC algorithm at focus and defocus, speed of the OPC algorithm, mask data expansion after OPC, and mask manufacturability measured in terms of complexity of corrections. The speed and data expansion criteria can be measured quantitatively. However, the effectiveness of OPC and the resulting mask complexity are more qualitative. We will define quantitative OPC effectiveness criteria and use them throughout the performance analysis. Finally, investigating the algorithm stability leads to conclusions about limits of algorithm performance and valid parameter range settings. In conclusion, we determine that OPC can be thought to "align" process windows for various types of structures, resulting in an larger process window overall. This gain comes despite the fact that the individual process window for a certain type of structure may not be improved.

6.1 Quantifying Mask Performance

Measuring the performance of a mask across an entire 2-D mask is not an easy task. For example, exposure-defocus tree data is useful for graphing performance of a single cutline, but in the 2-D case a single cutline is not sufficient to judge overall performance. Perhaps observing the worst-case performance of all cutlines on the mask is a reasonable criteria. However, there are always some areas of the mask which will not produce ideal pattern transfer such as rounding at corners. Despite these difficulties with cutlines measurements on 2-D masks, this method remains the only practical choice for experiments. In this section, the aerial images will be studied after OPC of the patterns using the "naive" resist model of a simple thresholder.

The edge placement error (EPE) and the aerial image contour itself will be the criteria for judging OPC in this section.

6.2 Mask Complexity

The number of vertices needed to describe a mask conveys useful information about the complexity and hence manufacturability of a mask. When comparing an optimized mask to the original mask, the data expansion factor (DEF) is therefore a useful metric. This measure is defined simply as:

$$DEF = \frac{\text{post OPC file size}}{\text{pre OPC file size}} \tag{6.1}$$

The algorithm can be tuned to produce various degrees of complexity by trading off decreased complexity for a smaller number of variables and hence lower "correction power". Reducing the correction power can also result in faster run times. Mask complexity can be user-controlled in a number of ways.

- Fragmentation control
- Control over which feature types receive OPC

For example, no interfeature correction may be performed. Then, only line-ends may be corrected.

6.3 Convergence and Stability

In the following discussion, the goal is to determine conditions under which the algorithm will be called "stable" in the sense of converging towards a solution. We will find that stability is ensured by making sure the cost function uses a local averaging of cost.

Stability for the mask optimization problem means that the edge variables ν_i are converging as the the number of iterations, P_{it} is increased. Denote the value of the displacement variable ν_i at the jth iteration by $\nu_i^{(j)}$. Then, the algorithm is stable if $\exists J$ and $\hat{\nu}_i$ values such that

$$\forall j > J \quad |\nu_i^{(j)} - \hat{\nu}_i| < \delta$$

where δ is a small number. This allows for a "noisy" convergence so that the solution can bounce around due to the grid discretization. The value of $\hat{\nu}_i$ is the exact solution which may not be attainable due to gridding.

In the case of divergence, it is essential to be able to ascertain whether it is caused by an unstable algorithm or an ill-conditioned problem which will converge for no algorithm. The mask design problem inherently has a high condition number, and as the desired minimum feature size goes smaller, the condition number increases until the problem is extremely ill-conditioned. Here, the discussion is confined to cases where convergence is possible, so the issue is algorithm stability.

The stability of the algorithm primarily depends on the size of the cost window. The cost window determines how large an area is in taken into account in the cost function when moving a single edge. Empirically, it is observed that making the cost window too small leads to instability [5]. The instability results in masks which have oscillations along their edges.

6.4 Trial Patterns

The performance of the algorithm is observed for trial patterns with k_1 ranging from 0.41–0.53. The intensity contours of optimized and unoptimized masks are compared at the 0.3 level. Edge placement failures are plotted for the original and OPC masks. Percentages of failures are tabulated in in Table 6.1. Also, the average intensity slope at edges is compared for OPC and original masks in the same table.

			av slope		% failures	
Pattern	Version	k_1	edges	corners	edges	corners
1	orig	0.53	4.4	2.1	34%	65%
1	opc (stable)	"	3.9	2.1	0%	21%
1	opc (unstable)	"	3.2	1.7	1%	14%
2	orig	0.48	2.2	1.4	60%	51%
2	opc	"	2.3	1.3	3%	13%
2	orig (.6 defocus)	"	1.8	1.1	29%	67%
2	opc (.6 defocus)	"	1.8	1.0	10%	55%

Table 6.1: Performance statistics for OPC vs original mask

6.4.1 Pattern 1: Stability Study

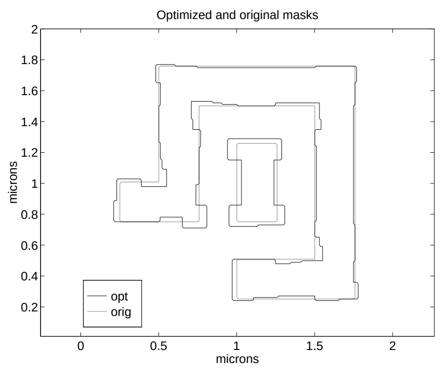
A single mask is tested using the algorithm with different cost window sizes in order to determine how the cost window size affects stability. The study is made by OPC using a large cost window and also a small window. The large cost window results in improvement of the intensity characteristics and reduces the EPE. For unstable operation, the cost window is reduced to include only a single simulation site. This causes oscillations along the edges of the resulting mask as seen in Figure 6.2. These oscillations are seen to degrade contrast performance of the mask which will be shown by a decrease in edge slope.

The trial mask has linewidth of 0.25 μm. It is optimized assuming a DUV stepper is being used with the parameters $\lambda = 0.248$, $NA = 0.53$, and $\sigma = 0.5$, making an effective $k_1 = 0.53$. The mask is snapped into a grid with grid size 10 nm. The step size for edge offset updates is also 10 nm.

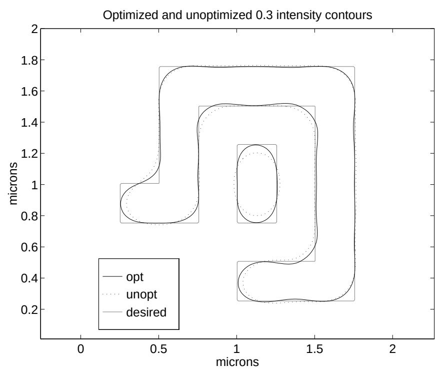
Stable parameters First, pattern 1 is optimized using full cost windows such that any simulation site which is affected by a perturbation will be included in the cost calculation, which is 0.64 μm $\times$0.64 μm in size. Figure 6.1(a) shows the resulting optimized mask overlayed with the original mask.

The 0.3 intensity contours of the unoptimized and optimized masks are shown in Figure 6.1(b). Qualitatively we notice that the optimization corrects for line-end shortening, tightens corners, and produces more accurate edge placements. Figure 6.1(c) and (d) emphasize the gain in edge placement accuracy produced by the correction. The failure plots show the edges whose EPE is greater than the 5% CD margin. Overlayed with the failure points is the intensity contour of the regions which would print. As seen in Table 6.1, OPC resolves all edges failures and reduces the number of corner failures by a factor of 3.

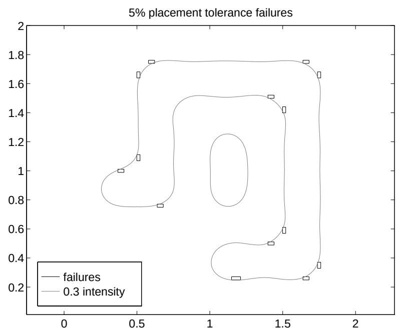
Unstable parameters In the next trial, the pattern 1 is optimized with a very small cost window of 0.02 μm$\times$0.02 μm. This cost window will contain only one simulation site. After 12 iterations, the resulting mask in Figure 6.2(a), to which we refer as the unstable mask exhibits large oscillations. The oscillations only increase in magnitude as the iterations continue. Because of its shape alone the mask is unattractive and will cause manufacturability problems. Surprisingly, however, the 0.3 intensity contour produced by this mask, shown in Figure 6.2(b), appears quite acceptable. From Table 6.1, the edge placement achieved with this mask is as good as with the previous stable mask. However,



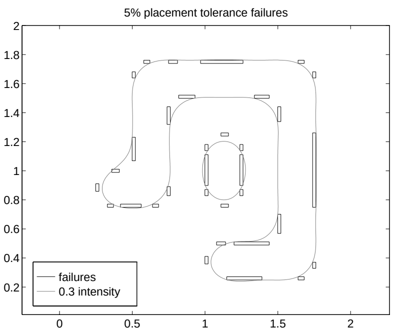
(a)



(b)



(c)



(d)

Figure 6.1: Pattern 1: (a) overlayed original and optimized masks (b) overlayed optimized and original intensities (c) optimized mask edge placement failures (d) original mask edge placement failures

the slope is lower for this mask, as discussed next.

Slope and Contrast Table 6.1 shows that the average slope is 17% lower for the unstable mask compared to the stable mask. This is interpreted as a sacrifice in image contrast and hence a smaller exposure window. Qualitatively speaking, because of these oscillations the edge transitions for this mask are not as abrupt and therefore the mask spreads out the intensity, producing a low-contrast image which passes through the 0.3 contour very accurately. From this example, another generalization which can be made is that, for a given desired intensity contour, an *increase* in the perimeter length of a featue on the mask which does not significantly improve the intensity contour will result in a *lowering* of image contrast. This qualitative generalization is upheld by the data in Table 6.1 for the unstable optimized mask, which has a larger perimeter length than the original mask and a slightly lower average slope. In terms of OPC, this means that a chosen OPC solution should be one which achieves a desired edge placement performance level and has the minimum perimeter length of all masks which achieve that same minimum performance level.

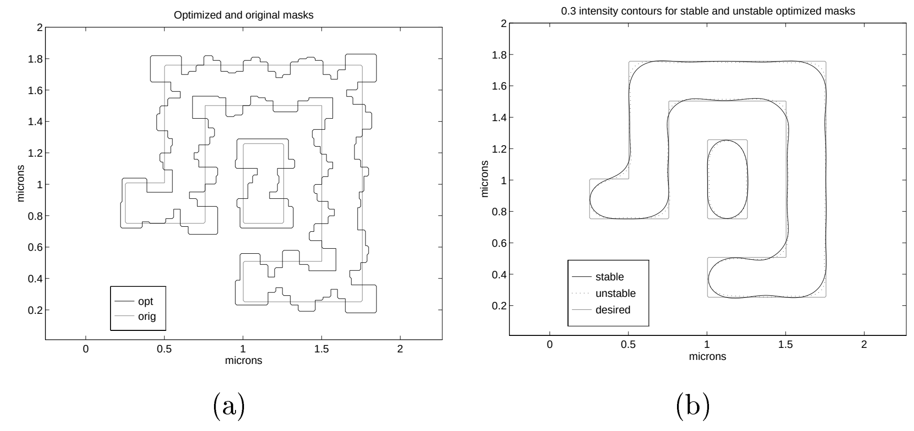


Figure 6.2: Pattern 1: (a) optimized mask with unstable parameters overlayed with original (b) intensity for optimized mask for with unstable parameters

Comparison and stability findings The important conclusion of stability analysis on the test pattern indicates that "single point control" is not sufficient for convergence. In other words the cost window must be large enough to include all simulation sites points from the surrounding neighborhood which are within the range of influence of a given perturbation because moving a single edge affects neighboring simulation sites.

6.4.2 Trial pattern 2

Pattern 2, in Figure 6.3, is another example pattern. It is a 36 $\times$36 μm^2 clear field mask with linewidth 0.35 μm, $k_1 = 0.48$ and 25 nm grid spacing. In this example, $\lambda = 0.365$, $NA = 0.5$, and $\sigma = 0.6$. The OPC converges after 8 iterations. For more analysis we zoom into a section of the mask shown in Figure 6.4, where the OPC mask is overlayed into the original design. At this k_1, the original mask prints, but corner rounding and line-end shortening are observed. The original mask has many edge placement failures as seen in Figure 6.6, totaling 60% of edge perimeter and 51% of corner perimeter including nearly all the line-ends. The OPC mask results in much fewer failures, as seen in Figure 6.5 where only 3% of edges fail, 13% of corners fail and all line-ends are placed accurately. From Table 6.1, the edge slope remains nearly unchanged by optimization.

Defocus Performance For pattern 2, the OPC mask outperforms the original mask for defocus up to 0.6 μm, above which the original mask has better edge placement. Figure 6.7(a) and (b) show the OPC and original mask edge placement failures at the 0.6 μm defocus plane. At 0.6 μm defocus, for the optimized mask 10% of edge fail to be placed accurately and 55% of corners fail for the optimized mask as seen in Table 6.1. For the original mask, 29% of edge fail and 67% of corners fail. The intensity at the 0.3 level is shown in Figure 6.7(c) and (d) for both the masks. At the 0.9 defocus plane, neither mask exhibits accuracy for small features, but as seen in Figures 6.7(e) and (f), the unoptimized mask has slightly more favorable intensity characteristics.

The conclusion of the defocus study is that the OPC mask displays improved performance for a range of defocus values, outside of which the original mask may actually perform better.

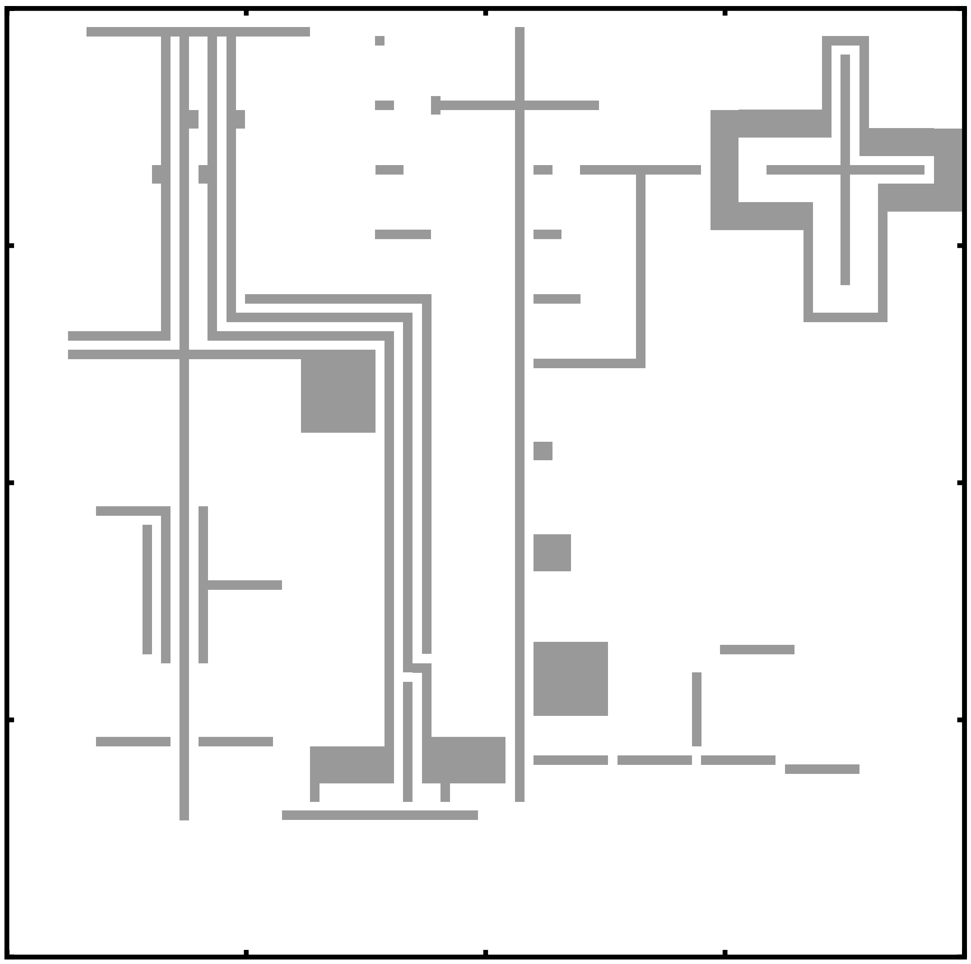


Figure 6.3: Pattern 2 original mask

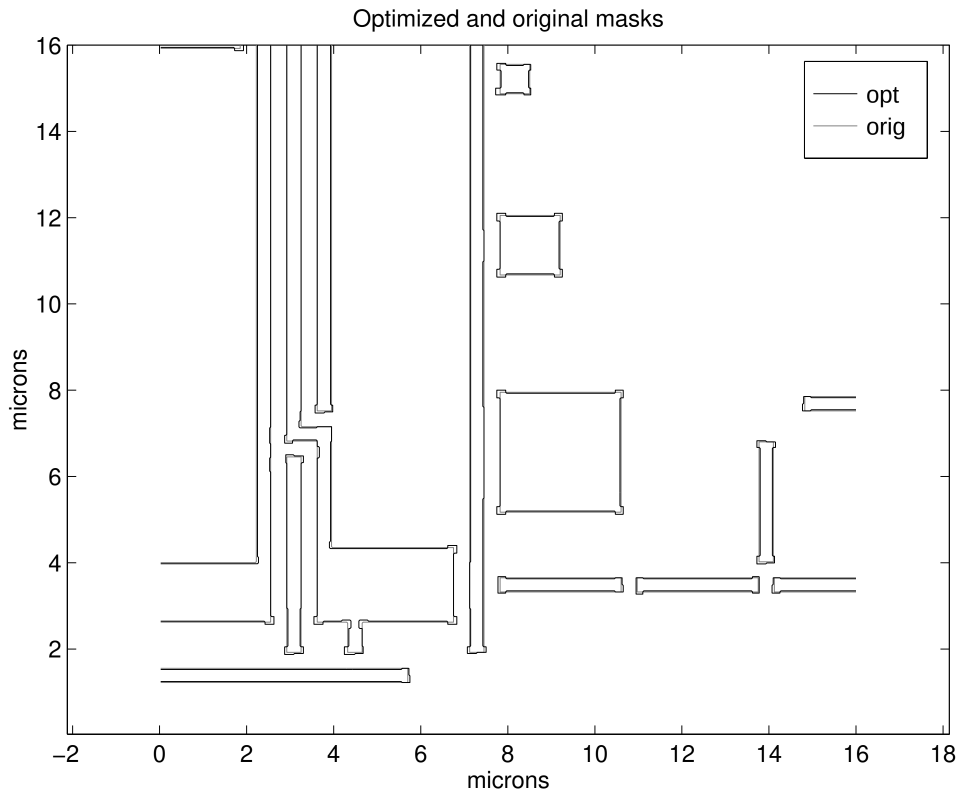


Figure 6.4: Pattern 2

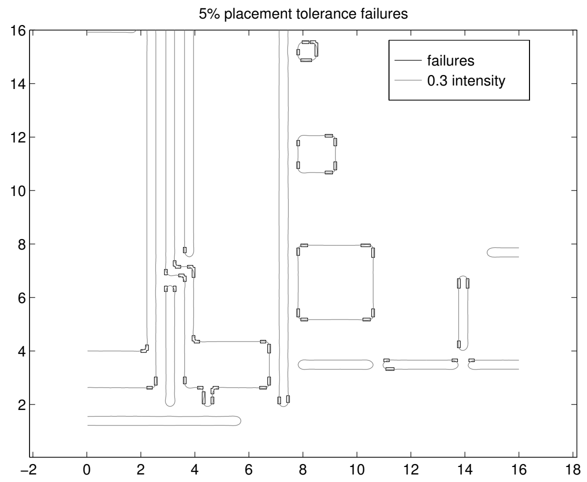


Figure 6.5: Pattern 2 optimized mask placement failures

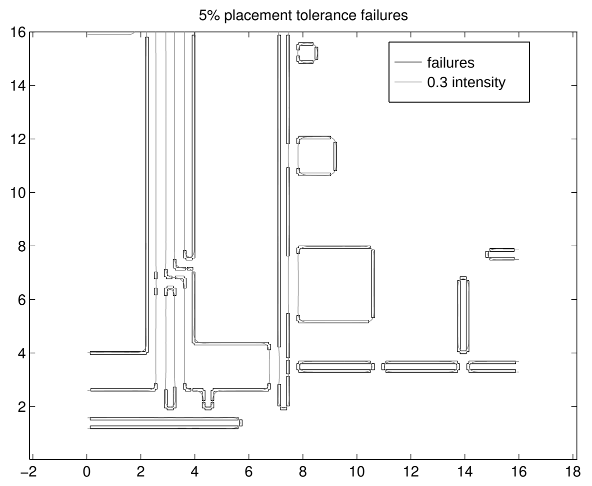


Figure 6.6: Pattern 2 unoptimized mask placement failures

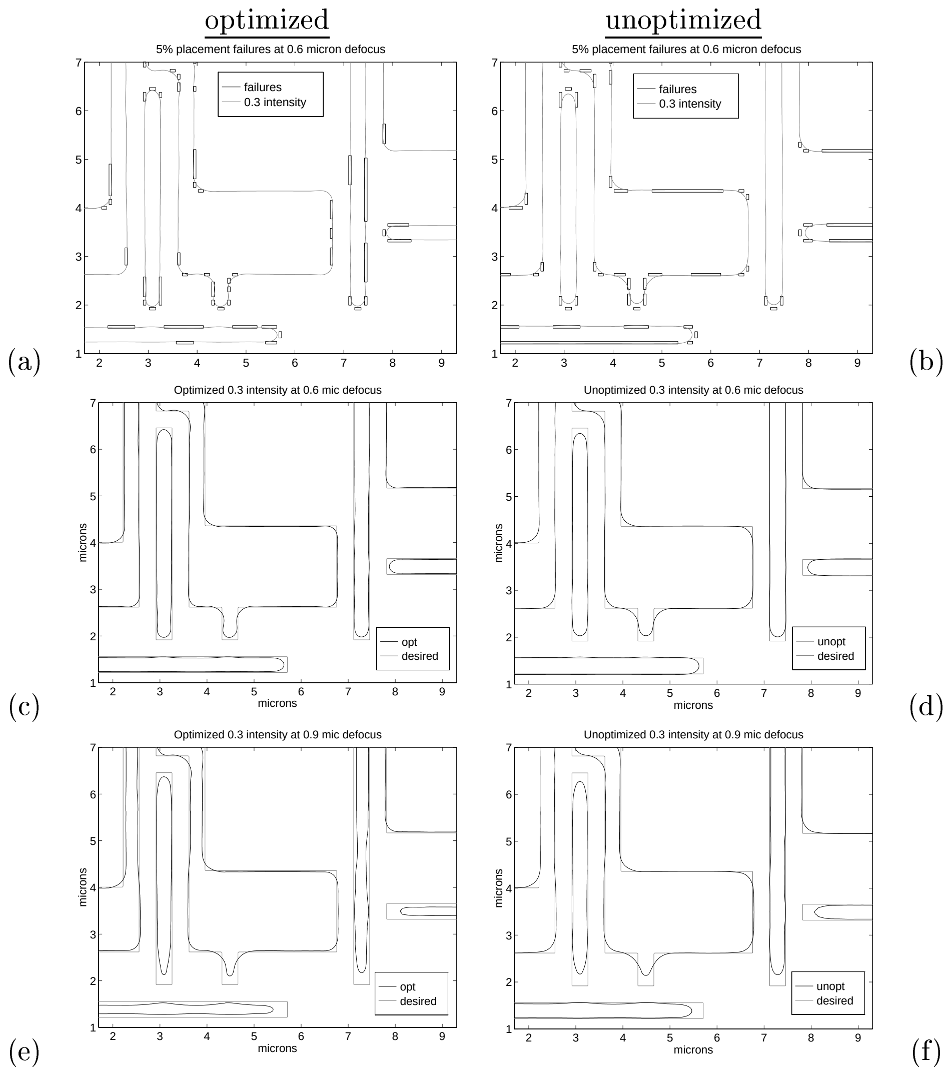


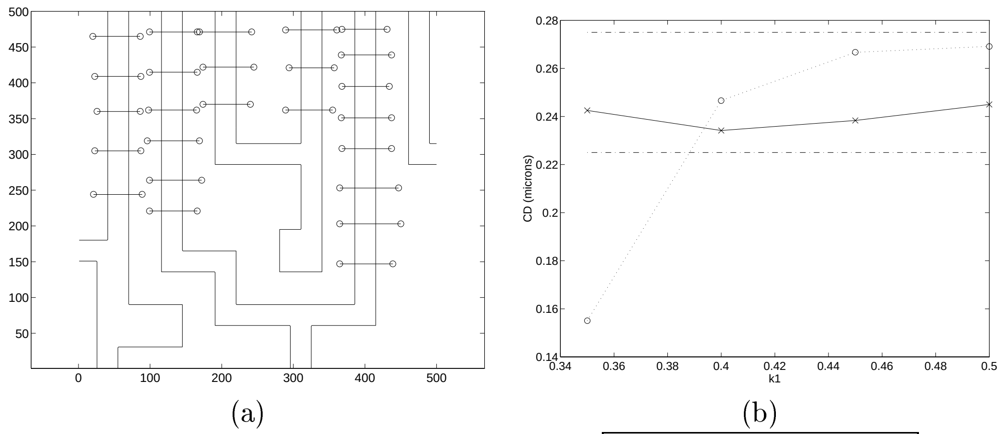
Figure 6.7: Pattern 2 defocus comparison

6.5 Scaling the mask for smaller k_1

In this section we investigate how OPC can affect the achievable k_1. Using OPC, k_1 factors down to 0.4 should be achievable. Normally, k_1 below 0.38 is not considered optically resolvable with binary masks. We will show that OPC can be used to improve contrast even below that threshold for patterns that are not dense.

6.5.1 Scaling k_1 using NA

In this example, $\lambda = 0.248$ and $\sigma = 0.5$. The k_1 factor is varied by varying the numerical aperture from 0.35–0.5 for a 0.25 μm linewidth pattern. In Figure 6.8(b), the average CD for the cutlines in Figure 6.8(a) is plotted versus varying k_1. At the lower k_1 values, the CD is not met for the original mask. The CD is kept within the 10% tolerance down into the $k_1 = 0.35$ range for the OPC mask. The explanation for this "impossible" result, is that the mask geometry is modified by OPC, so the actual k_1 after OPC is larger than that of the original mask.



$\times$ = OPC, $\circ$ = original

Figure 6.8: (a) Cutline locations (b) average CD versus k_1 for cutlines

6.5.2 Scaling down the dimensions

In this section, we start out with pattern 2 and shrink it first to linewidth = 0.3 μm and then to linewidth = 0.25 μm, corresponding to $k_1 = 0.41$ and $k_1 = 0.34$. The light source is $\lambda = 0.365$ μm with $NA = 0.5$ and $\sigma = 0.6$. With these masks, we test the limits of the OPC algorithm correction power. The two OPC masks for these k_1 values are shown with the original masks in Figure 6.9(a) and (b).

In Figure 6.10(a) and (b), the intensity contours for the OPC and original masks at $k_1 = 0.41$ and linewidth 0.3 are shown. As with the previous example, the optimized mask here produces accurate line-end placement, reduces corner rounding and produces more accurate edge placement.

In Figure 6.11(a) and (b) are the intensity contours from the OPC and original masks at linewidth of 0.25 with $k_1 = 0.34$. At this linewidth, the original mask fails completely as seen in Figure 6.11(a). The OPC mask improves the performance as shown Figure 6.11(b). Although the line placement errors are still severe enough to reject this mask, the performance gain over the OPC mask dramatically illustrates the correction ability of the algorithm.

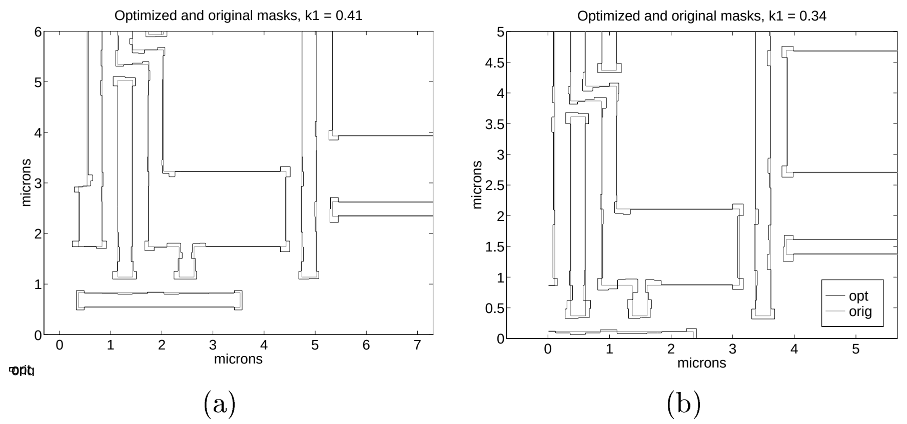


Figure 6.9: (a) Pattern 2 OPC mask where $k_1 = 0.41$ (b) Pattern 2 OPC mask where $k_1 = 0.34$

6.6 Algorithm Speed and Data Expansion

The data in Table 6.6 summarizes some information about algorithm speed for each of the trial patterns. The table lists the optimization time per iteration* for typically around 10 iterations per optimization. These optimization times give a good indication of

*All times are approximate times on an HP 700 Series workstation.

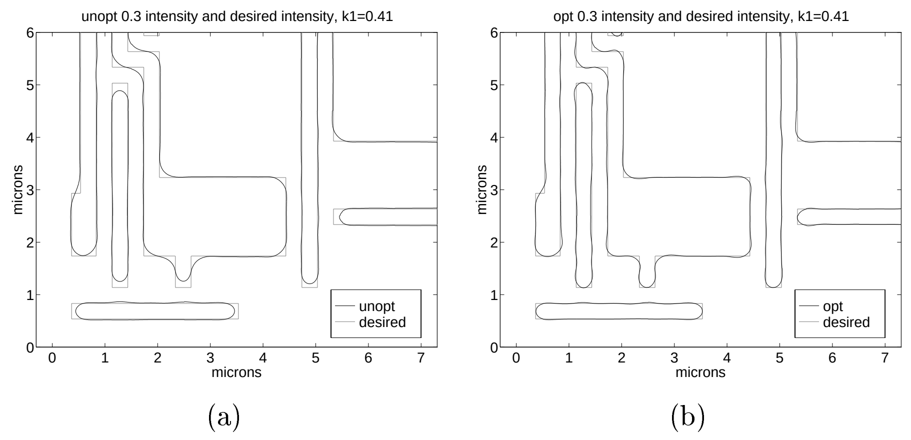


Figure 6.10: At $k_1 = 0.41$ (a) intensity of original mask (b) intensity of OPC mask

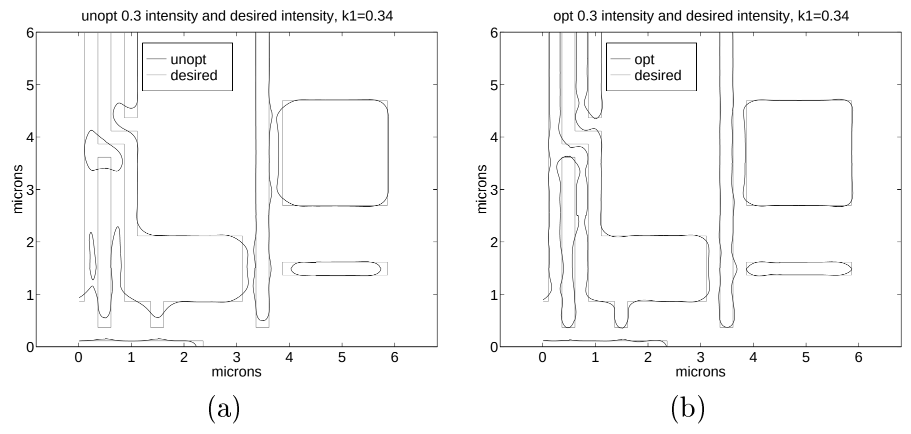


Figure 6.11: At $k_1 = 0.34$ (a) intensity of original mask (b) intensity of OPC mask

the current capabilities of the algorithm.

The column labelled "# of input rects" in the table refers to the number of rectangles in the original mask description, M_r. The number of rectangles in the optimized mask description is given by DEF$\cdot M_r$.

The difference in OPC time for pattern 2 using the "projections with ripples" variable assignment versus the "corners only" variable assignment indicates a speedup factor of 3.1. This results from the reduction in number of variables from 977 to 472, a factor or 2.1. Theoretically from the results in Section 2.8, this should cause a speedup factor of up to $2.1^2 = 4.3$. In practice, the speedup is only 3.1 due to programming overhead.

Pattern	size	# objects			# input	
#	(μm)	edges	corners	time/iter	rects	DEF
1	2×2	25	16	1.1 sec	6	6.0
2	36×36	505	236	8.4 sec	80	4.9
2 (corn only)	36×36	0	236	2.7 sec	80	3.5

Table 6.2: Algorithm speed for trial patterns

Two figures of interest which are not shown in the table are: single point intensity computation speed, and approximate optimization time per mask object. These quantities are found experimentally at this time. The single point intensity computation time ranges from 260–540 μsec* depending on the density of rectangles on the mask and including overhead costs. Experimentally, the average optimization time is around 30–200 msec per mask object per iteration. Using the data here, we can extrapolate the time required for a 1 mm $\times$1mm mask to be around 15–150 hours with a strong dependence on variable object density.

Chapter 7

Experimental Results on OPC

Previously we described how to combine fast aerial image simulation with the VTR model and a closed-loop Optical Proximity Correction (OPC) control system which can compensate for distortion effects at smaller feature sizes. The feedback nature of the OPC system makes it accurate to the precision of the simulation model, in cases where correction is feasible.

Now we present experimental verification of the OPC algorithm and simulation models presented earlier. To model the processes, we use empirical data in the form of linewidth measurements in order to arrive at a VTR model. Then, OPC is performed on test structures using the VTR model.

7.1 Experimental Results

7.1.1 i-line

The i-line experiments were performed with the following parameters:

Summary	
lambda	0.365
NA	0.60
sigma	0.60
resist	JSR, 1.0 μm

The overall structure of the experiment is as follows:

- Print a test pattern

- Take measurements of test pattern
- Create a process model
- OPC more test patterns using the process model
- Print OPC patterns and observe results

A model was fit to the empirical data to make a VTR model. Using the tuned model to perform OPC, 8 iterations were performed with 10 nm step size. An example of the resulting OPC pattern is shown in Figure 7.1, along with the original pattern. The pattern was written out by 3 different mask writer tools, which we will refer to as RWA, RWB and RWC. One wafer was printed with each mask, so that we obtained 3 different wafers. The wafer SEM images for the corrected patterns and the original patterns are shown in Figure 7.2. The improvements to the wafer after OPC are apparent: bridging has been corrected and more accurate widths are present on the wafer.

Another corrected structure, shown in Figure 7.3, is the so-called “1.5D” structure, because it involves the interaction between a very long line and another line. In these images, the benefit of OPC can be seen. For example, Figure 7.3(a) and (b), the bridging is corrected by OPC as shown. As seen, there is considerable improvement from uncorrected designs to corrected designs for all three reticle writers. In particular, the bridging effect in the uncorrected designs is completely removed in the corrected design. Another conclusion to be drawn from Figure 7.2 and Figure 7.3 is that the reticle writers play an important role in the uncorrected wafer SEMs.

7.1.2 DUV

In these set of experiments, a test reticle was again printed in order to calibrate the process under consideration. Approximately 45 post-etch SEM measurements were then used to determine the resist model.

Summary	
lambda	0.248
NA	0.50
sigma	0.60
resist	APEX, 0.89 μm

The VTR threshold surface itself is shown in Figure 7.4. Figure 7.5 shows the goodness of fit of the empirical data to both constant and variable threshold model. The x-axis in this graph corresponds to different sample sites on various structures, and the y-axis is the error between simulated CD and empirical CD. The VTR model provides a better fit to the data than the CTR. The mean and standard deviation of the VTR model are 0.3 and 8.1 *nm* as compared to 1.5 and 17.2 *nm* of the CTR model. After OPC using this model, a plot of CD vs. pitch is shown in Figure 7.6. These wafer results show a uniform CD through pitch when using the OPC mask.

7.2 DUV Defocus effects

Another experiment was performed with DUV OPC correction. Two different experiments were attempted in batch.

Summary	
lambda	0.248
NA	0.42
sigma	0.50
resist	APEX, 0.83 μm

7.2.1 Defocus

The hypothesis which was tested in this experiment is that the defocus thickness is used in the OPC simulation model is important for the accuracy of the OPC corrections.

First, a VTR model, model **mod_9** was created from 42 measurement points. This model was created assuming a 0.9 μm resist thickness. A second model, model **mod_4** was created using the same data, assuming a 0.4 μmresist thickness. The actual resist thickness was 0.83 μm. The assumption is that OPC using **mod_9** will be more accurate than using **mod_4** because the defocus thickness which is used is closest to the actual 0.83 μm thickness.

The wafer results verify the hypothesis, as seen in Figure 7.7. In the figure, the target CD prints uniformly through pitch for **mod_9**, but using **mod_4**, the CD is printing off by 0.1 μm in the dense features. The conclusion we reach is that the defocus thickness simulated during OPC is highly important for accurate OPC.

7.2.2 Pitch

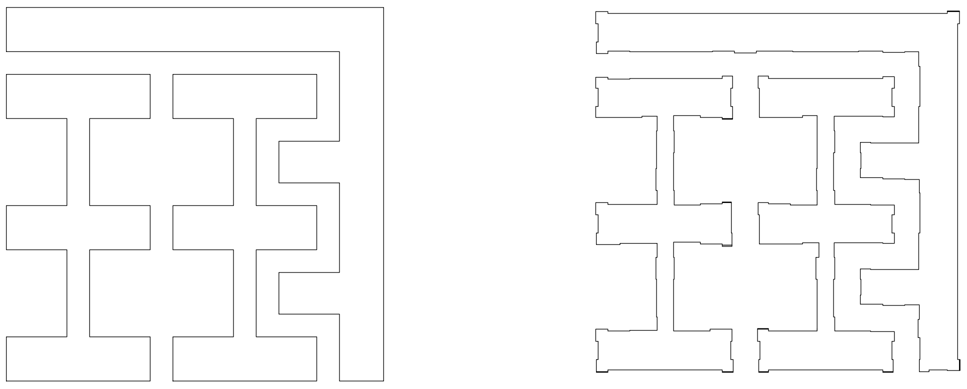
In another experiment, three models were created:

m_all32 use all 42 data points to create the model

m_pitch32 use only 24 data points taken for varying pitch data to create the model

m_le32 use only 18 data points taken from line end data to create the model

The goal in this experiment is to see how accurately different types of structures can be corrected using models taken from only a subset of the available data. The results are seen in Figure 7.8. Again, in this uniformity plot, the target linewidth is 0.26 μm as the pitch is varied from dense to isolated lines. The effect is that OPC improves the uniformity for all models, as compared to the control case. The conclusion is that enough information is available in a subset of all the 42 empirical data points to characterize significant dynamics in the process. Therefore, accurate models can be obtained using whatever data is available, and modeling for different structures is not highly sensitive to the data used to create the model.

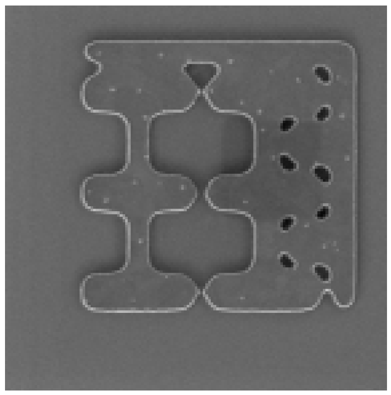


(a) (b)

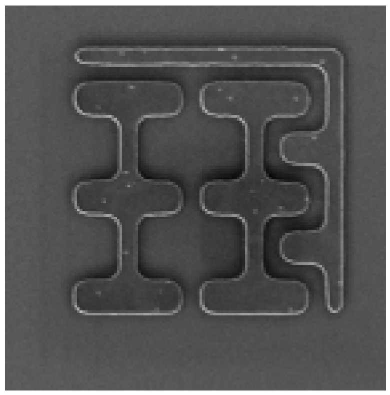
Figure 7.1: (a) Original pattern OPC (b) OPC pattern

7.3 Conclusion

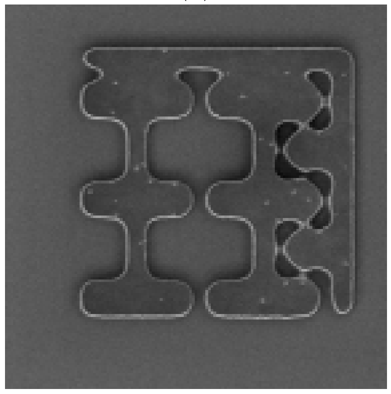
The experimental results on silicon verify the OPC approach previously described. Several conclusions can be made from these experiments. First, OPC can significantly



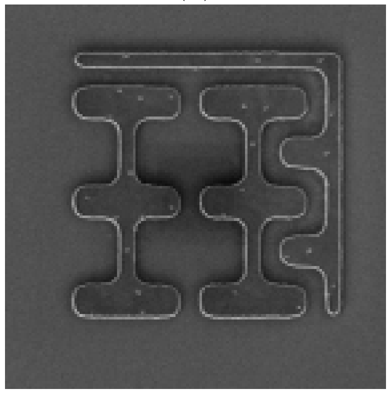
(a)



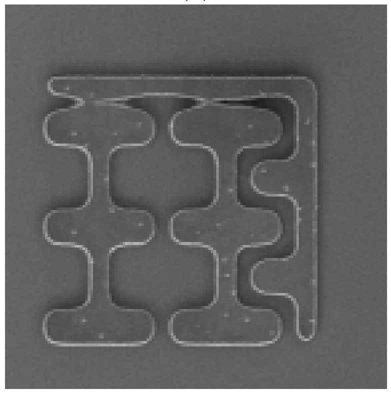
(b)



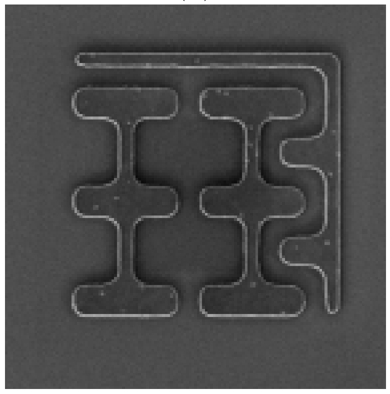
(c)



(d)

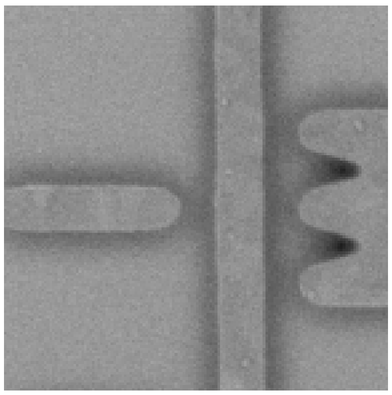


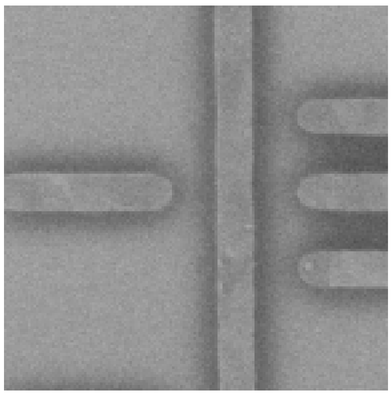
(e)

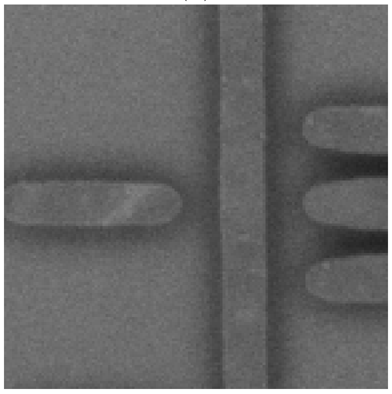


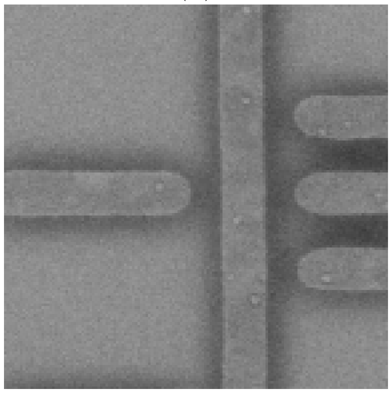
(f)

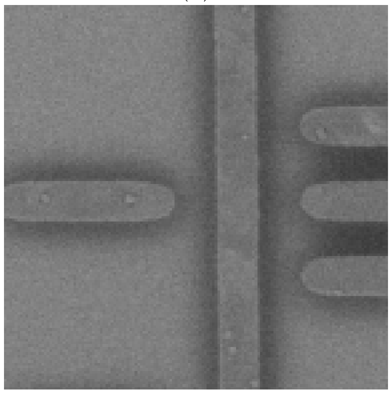
Figure 7.2: (a) Wafer SEM of uncorrected pattern with RWA; (b) Wafer SEM of corrected pattern with RWA; (c) Wafer SEM of uncorrected pattern with RWB; (d) Wafer SEM of corrected pattern with RWB; (e) Wafer SEM of uncorrected pattern with RWC; (f) Wafer SEM of corrected pattern with RWC;


(a)


(b)


(c)


(d)


(e)

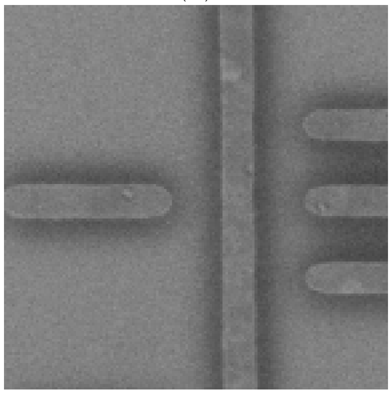

(f)

Figure 7.3: (a) Wafer SEM of uncorrected pattern with RWA; (b) Wafer SEM of corrected pattern with RWA; (c) Wafer SEM of uncorrected pattern with RWB; (d) Wafer SEM of corrected pattern with RWB; (e) Wafer SEM of uncorrected pattern with RWC; (f) Wafer SEM of corrected pattern with RWC;

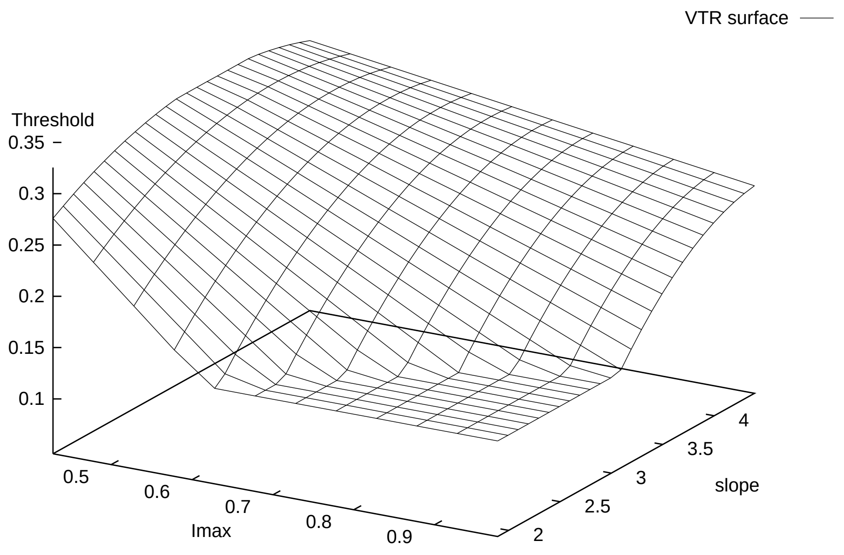


Figure 7.4: VTR surface

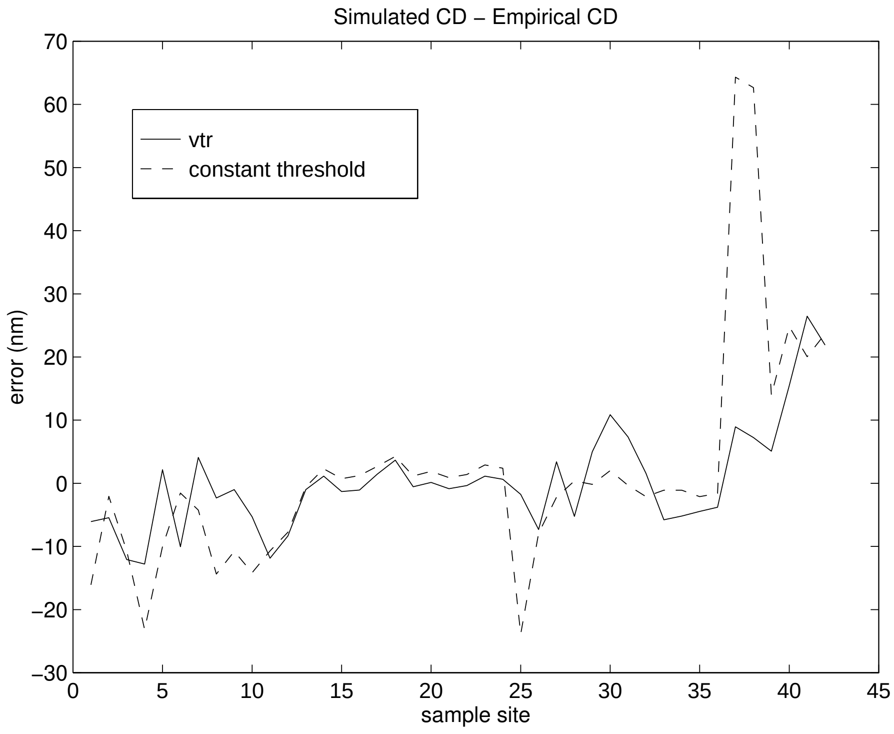


Figure 7.5: VTR and CTR model fit to empirical CD data for DUV process

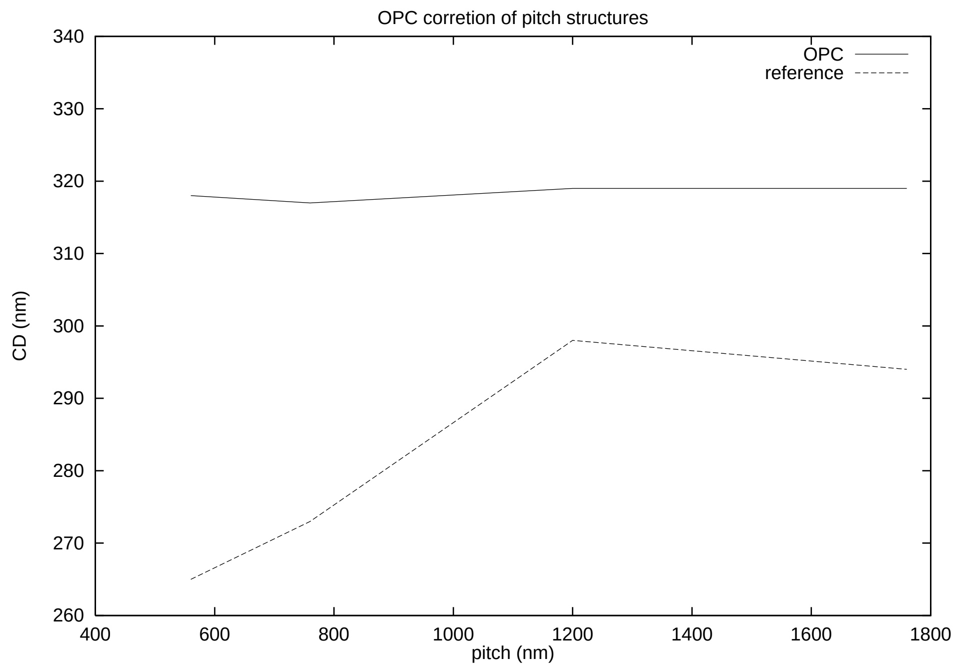


Figure 7.6: VTR and CTR model fit to empirical CD data for DUV process

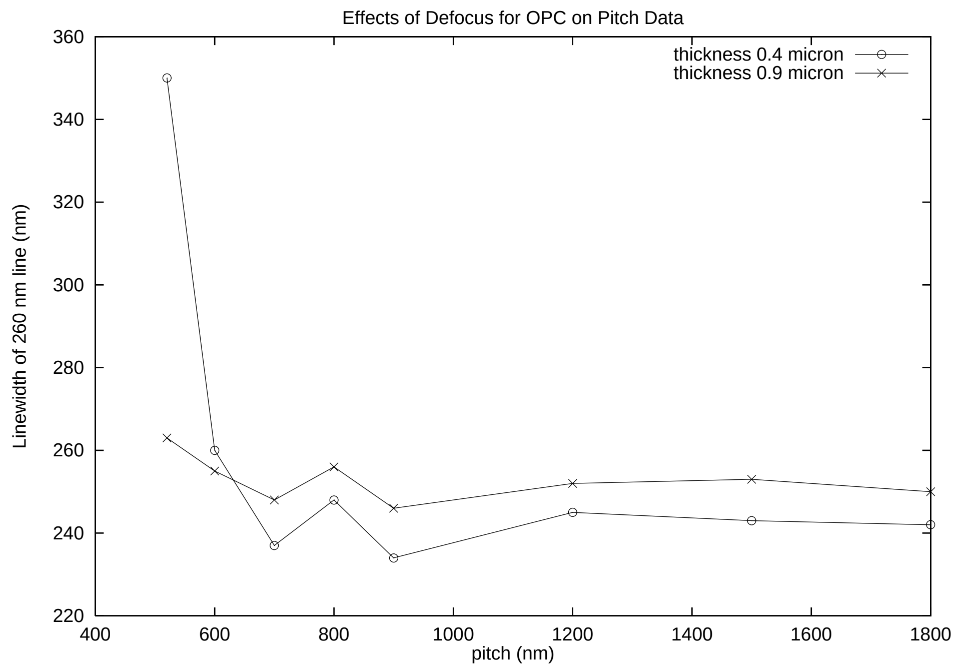


Figure 7.7: Effect of OPC through defocus on 1-D pitch structures.

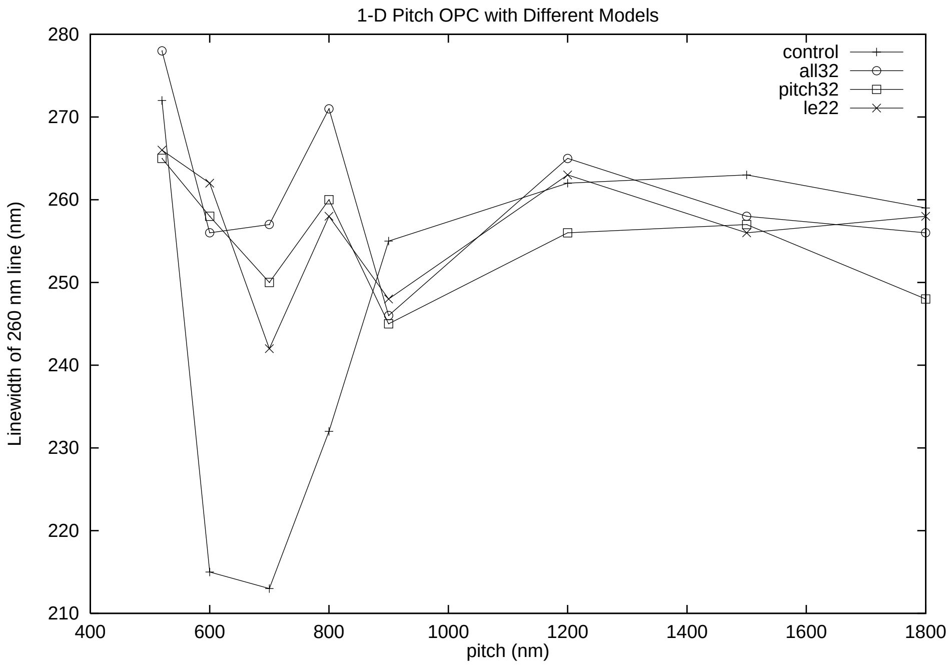


Figure 7.8: OPC of 1-D pitch structures with different models.

improve bridging effects seen in uncorrected patterns. Second, reticle writer effects cannot be ignored and must be taken into account in printing features in addition to OPC. Another conclusion is that resist thickness and defocus effects are important and should be modeled in the OPC simulation. A successful used model of the resist thickness is to average the image over a range of defocus. Future directions for research include taking into account reticle writing imperfections, and cross chip line width variations, and more accurate process modeling techniques.

Chapter 8

Conclusions

The algorithm described in this report performs Optical Proximity Correction using a simulation feedback optimizer structure. The block diagram of the entire system separates the OPC algorithm from the simulation modules which drive it. This makes the system general enough to "plug in" different simulation models as they are needed.

A number of constraints present in mask manufacturing and mask inspection are built into the algorithm so that edges will not be moved in such a way to create highly complicated designs. The overall number of fragmented edges can be controlled in by using "adaptive" fragmentation in which areas near corners will have more fragments than areas which are farther from corners.

The speed of the OPC is increased by using a fast aerial image intensity calculation technique described in this report. The first step to fast aerial image is the use of an eigenvector decomposition of the discrete Hopkins equations which we call Sum-Of-Coherent-Systems, or SOCS for short. The decomposition produces a structure in which aerial image can be calculated using convolutions.

The convolution structure illuminates the possibility of using linearity to speed up calculations. A general technique of fast convolution of an arbitrary convolution kernel with area-wise constant functions is presented. In this technique, the convolution is implemented completely by look-up of edges. The fast aerial image technique outlined here can compute intensity at speeds in excess of 11,000 pts/sec and perturbation update times of 6.3 msec/perturbation*.

A simplified VTR model for resist development and etch is presented, which using

*On a Sun SPARC10 workstation

the maximum intensity and the slope of the image profile to determine the point at which an edge will print. The benefit of this model is that the parameters for the model are determined by empirical data. In the modeling process, a set of "basis geometry" is printed, and then the dimensions measured. Then the measurement data is used to create the model.

Future Work The findings in this work indicate that OPC has every chance of being a valuable aid to reliable pattern transfer at decreasing dimensions. Substantial amounts of work remain before the OPC problem can be considered "solved". Many new additions to the OPC simulation models are possible. The effects of mask writing, resist, post-exposure bake, and loading effects in etch can be modeled more accurately, thus improving the modeling accuracy over what is possible with the variable threshold resist (VTR) model alone. More extensive experimental testing of the modeling accuracy and the OPC algorithm accuracy should also be performed. A range of masks and the optimized mask designs should be fabricated on different types of mask writing equipment.

In the area of mask inspection, it remains to be determined exactly how detailed corrections will be before they cannot be inspected reliably. Also, in mask manufacturing itself, the faithful reproduction of the mask is an area which is even more important for OPC features. An extensive study of defects on OPC structures and how they affect the wafer would also provide valuable information for practical use of OPC in industry.

In the CAD area, work is required in formulating a "design rule checker" (DRC) for OPC-ed structures. Such a DRC may operate by simulating lithography to predict where problem areas may exist even after OPC. A DRC for OPC data would provide a very valuable addition to the bare OPC capabilities.

With the framework and algorithms described in this work, the practical implementation of OPC in industry fab lines becomes a possiblility. Fast simulation models and a iterative OPC feedback structure make this possible.

Appendix A

Geometry algorithms

The polygon data used for mask layouts presents many challenges for OPC. One of these challenges is the need to describe the mask as a set of non-overlapping polygons. The polygon OR operation is used to accomplish this. In this section we present a generalized technique for manipulating polygons which can be used to perform polygon OR, polygon AND, polygon XOR, and many other polygon operations, which we will call "polygon booleans".

A.1 Polygon OR

Problem Statement: Given a set, S_1, of simple polygons defined on R^2, find a set of non-overlapping simple polygons, S_2 which covers the same area as S_1.

Outline Techniques exist to solve this problem, however, to the author's knowledge, existing techniques view the problem as a boolean set operation on R^2. From this point of view, the problem can be reduced to finding out which edges in the initial set of polygons should remain after the boolean OR.

We will show a way to solve the problem by considering indicator functions defined for polygons on R^2. It this view, we take the indicator functions for each polygon

$$1_p(x,y) = \begin{cases} 1 & (x,y) \in p \\ 0 & \text{otherwise} \end{cases}$$

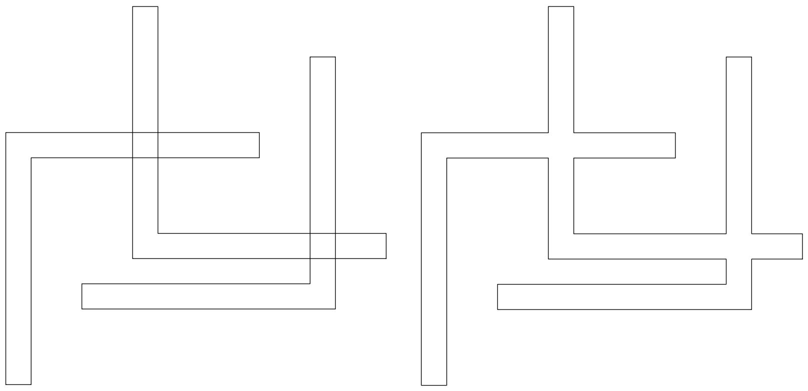


Figure A.1: (a) Overlapping polygons (b) Polygons after the "polygon OR" operation

and sum them

$$F(x,y) = \sum_{k=1}^{N_p} f_k \cdot 1_{p_k} \tag{A.1}$$

Then, if we threshold the result:

$$Y(x,y) = \begin{cases} 1 & \text{if } F(x,y) >= T \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

where, for now, $T = 0.5$. The problem is transformed into finding a set polygons whose indicator functions sum to Y. This is essentially the same as taking a contour plot of Y at the threshold contour level.

A.2 Solution to the 1-D problem

The algorithm can be derived for the 1-D case and then generalized back to 2-D. An interval in R is analogous to a polygon in R^2. An interval can be represented by a pair of two numbers $(p1, p2)$ which are the left and right endpoints of the interval, respectively, as shown in Figure A.2.

A.2.1 1-D "interval OR"

The "interval OR" operation is the analog of the "polygon OR" operation. We apply the same thinking as outline in section A.1. Therefore, for each interval p define the indicator function:

$$1_p(x) = \begin{cases} 1 & p1 <= x <= p2 \\ 0 & \text{otherwise} \end{cases}$$

Then, define the sum of indicators for all intervals to be

$$F(x) = \sum_{p \in S_1} f_p \cdot 1_p$$

where for now, assume all $f_p = 1$. Then, we threshold the result:

$$Y(x) = \begin{cases} 1 & \text{if } F(x) >= 0.5 \\ 0 & \text{otherwise} \end{cases}$$

And the problem is transformed into finding the set of intervals whose indicators sum to Y. An example of this is depicted in Figure A.3.

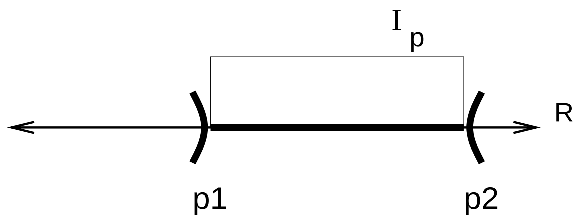


Figure A.2: Interval on R with indicator function defined

Observation 1: The function Y is piecewise constant, and can only change values at the endpoints of intervals in S_1.

Observation 2: There exists a set of intervals, call it S_2, such that Y can be described as the sum of indicator functions for intervals in S_2. Moreover, due to **Observation 1** the desired intervals have endpoints which will be in the set of endpoints of the intervals in S_1.

Observation 3: Assuming all of the intervals are bounded, then the mean value theorem says that there will be an equal number of number of left endpoints and right endpoints in Y. In essense, “What goes up must come down”.

Observation 4: Combining **Observations 1–3**, we can simply search through the set of endpoints in S_1 and evaluate each one to find whether it is a left endpoint, a right endpoint, or not a required point of a interval in S_2.

The **observation 4** above says that we can determine the set S_2, which is equivalent to performing the "interval boolean", by evaluating $2N$ points for N intervals in S_1. As mentioned in **observation 4** above, there are three possible values for a given endpoint:

- left endpoint
- right endpoint
- not an endpoint

To determine the status of a given endpoint, x, the `isOnDrop` function is called:

```
function isOnDrop(x) {
  if ( F(x + eps) >= T) && (F(x - eps) < T)
    return (right endpoint)
  else if  ( F(x + eps) < T) &&  (F(x - eps) >= T)
    return (left endpoint)
  else
    return (not an endpoint)
}
```

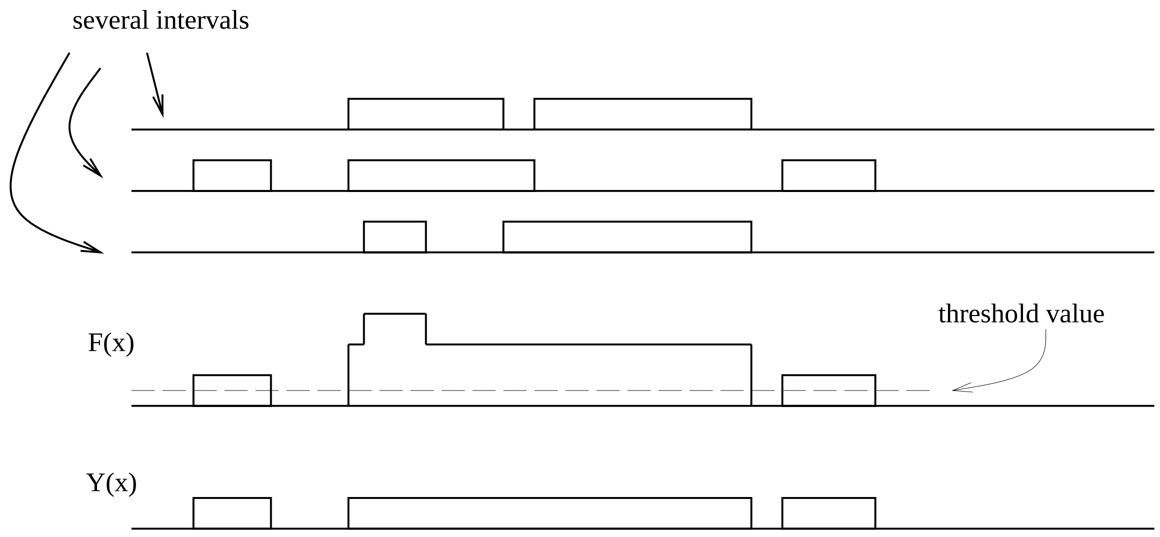


Figure A.3: Several interval which are being OR-ed together. The sum of the interval indicators, $F(x)$, is shown as well as the thresholded version $Y(x)$ whose support regions are the desired "interval OR".

All endpoints which are either right endpoints or left endpoints are saved, along with their `isOnDrop` status. The resulting intervals can then be reconstructed from these endpoints. The intervals form the set S_2.

We note that what we are doing here is nothing more than taking a contour plot of a function defined on R. The insight relating the contour plot to the boolean operation is the main message of this result.

A.2.2 1-D Problem Extended

Using the insight gained for the 1-D "interval OR" problem, we can extend the result to allow us to perform pairwise operations for AND, AND NOT and XOR. The generalization is possible using Equations A.1 and A.2, but using different values for f_p and T.

Booleans	f_1	f_2	T
OR	1	1	0.5
AND	1	1	1.5
AND NOT	1	-1	0.5

Table A.1: Boolean operations using the described algorithm.

The XOR function is built up using the AND NOT operation

$$p_1 \text{ XOR } p_2 = (p_1 \text{ AND NOT } p_2) \text{ or } (p_2 \text{ AND NOT } p_1)$$

An example of each of these is shown in Figure A.4.

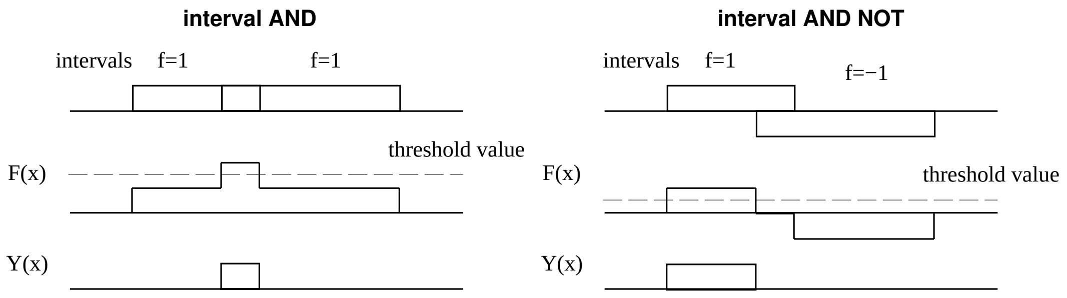


Figure A.4: Other pairwise boolean interval operations.

A.2.3 1-D Generalized to 2-D

The 1-D interval boolean operations are analogous to 2-D polygon booleans. The analog of endpoint in 1-D is polygon edge in 2-D. The `isOnDrop` function in 2-D should return the direction of the polygon edge such that the inside will be to the right of the edge, assuming our polygons are oriented clockwise. After all edges are obtained using `isOnDrop`, the resulting polygons are obtained by connecting the edges together.

Appendix B

Basis geometries in the test pattern

Printing and measuring CDs on the basis set of geometries provides enough information to characterize a process. Once a process model has been generated, OPC can be performed on any geometries. The basis geometries have symmetric structure so that accurate measurement and modeling is possible. The basis structures in Table B.1 will be described in detail in the following pages.

Each entry Table B.1 has a *name* for the structure. The *type* of structure is either 1-D, 1.5-D, or 2-D. The 1-D structures are structures which can be viewed as truncated infinite lines. For example: isolated lines, dense lines, and pitch lines, are all 1-D structures. The 1.5-D structures are where a long line is near other geometry, such as the T junction. The 2-D structures have more complicated geometry involving corners, or line-ends.

The *property* column lists the qualitative property that should be viewed for ideal data of the give *name*. For example, if the measured CD of the isolated lines is plotted versus the target CD, the plot should be linear, as in Figure B.2. For structures with no property listed, the arrangement of data of the test pattern is not sufficient for a meaningful plot, such as a linearity or uniformity plot.

The *measure* column lists the location which to measure the CD for the give structure. A more detailed explanation of the *measure* location is given for each structure in the individual structure descriptions which follow.

Name	Type	Property	Measure
Isolated lines	1-D	linearity	linewidth
Dense lines	1-D	linearity	linewidth
Pitch	1-D	uniformity	linewidth
Double lines	1-D	linearity	space between lines
inverse iso	1-D	linearity	width of space
inverse double	1-D	linearity	width between holes
island	2-D	linearity	width
inverse island	2-D	linearity	width of hole
dense island	2-D	linearity	center island width
line end	2-D	linearity	gap between line ends
dense line end	2-D	linearity	gap between line ends
inverse line end	2-D	linearity	between inverse line ends
T junction	1.5-D	NA	center line width
double T junction	1.5-D	NA	center line width
corners	2-D	linearity	outer gap between corner
dense corners	2-D	linearity	outer gap between corner
bridge	2-D	NA	width of bridge

Table B.1: Basis structures

B.0.4 Isolated lines

Name	Type	Property	Measure
Isolated lines	1-D	linearity	linewidth

The isolate lines basis structures range in width from slightly below the target CD up to above 1 μm wide. A plot of achieved CD versus target CD is called a linearity plot and it a typical calibration technique reveals valuable information about a process. An isolated line structure is shown in Figure B.1. An example of a linearity plot is shown in Figure B.2.

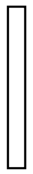


Figure B.1: Isolated line strucuture

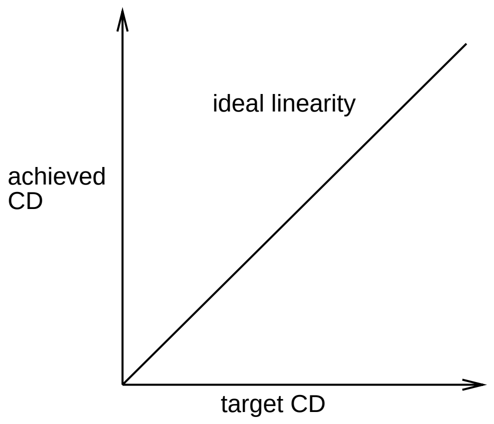


Figure B.2: Example of a linearity curve for isolated lines.

B.0.5 Dense lines

Name	**Type**	**Property**	**Measure**
Dense lines	1-D	linearity	linewidth

The dense lines basis structures is a small grating which has equal line width and space width. The width ranges from just below the target CD up to above 1 μm wide. A dense lines linearity plot can be generated by plotting the achieved CD versus target CD. A dense line structure is shown in Figure B.3.

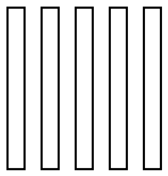


Figure B.3: Dense line strucuture

B.0.6 Pitch lines

Name	Type	Property	Measure
Pitch	1-D	uniformity	linewidth

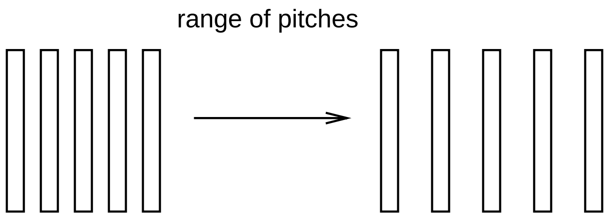


Figure B.4: Pitch strucuture

The pitch lines basis structure is a small grating which has a fixed linewidth but the space separating the lines is variable. The linewidth plus the space is called the pitch, or the period. The linewidth is at or near the target CD, The spacing ranges from just below the target CD up to above 1 μm. A plot of the CD as a function of pitch (period) is called a uniformity plot. An ideal uniformity plot shows a flat characteristic through pitch. An example of a uniformity curve is shown in Figure B.5. A pitch structure is shown in Figure B.4.

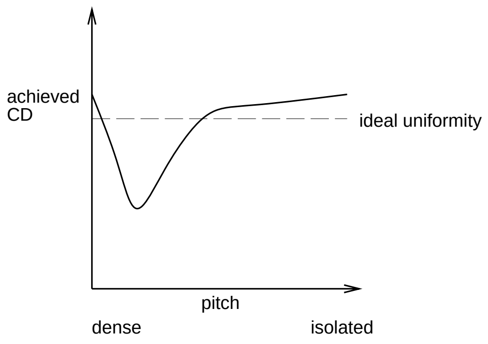


Figure B.5: CD Uniformity through pitch plotted as a function of pitch

B.0.7 Double lines

Name	**Type**	**Property**	**Measure**
Double lines	1-D	linearity	space between lines

The double lines basis structure is just two lines in close proximity. The space between the two lines should be measured. The designed linewidth of each line is equal, which is also equal to the spacing between the lines. The test pattern contains a row of double line structures where the CD is increasing from left to right across the row. A linearity plot of achieved gap versus designed gap can be constructed with this data. Ideally, perfect linearity would be observed. A double line structure is shown in Figure B.6.

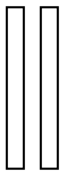


Figure B.6: Double line structure

B.0.8 Inverse isolated lines

Name	Type	Property	Measure
inverse iso	1-D	linearity	width of space

The inverse isolated line is merely an inverse tone isolated line. This is just a gap between lines basis structure is just two lines in close proximity. The space between the two lines should be measured. The designed linewidth of each line is equal, which is also equal to the spacing between the lines. The test pattern contains a row of inverse isolated line structures where the CD is increasing from left to right across the row. A linearity plot can be constructed with this data. Ideally, perfect linearity would be observed. An inverse isolated line structure is shown in Figure B.7.

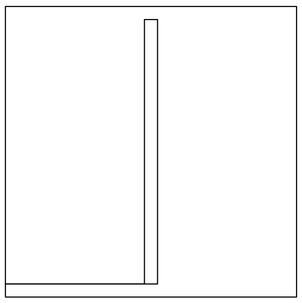


Figure B.7: Inverse isolated line

B.0.9 Inverse double lines

Name	Type	Property	Measure
inverse double	1-D	linearity	width between holes

The inverse double line is merely an inverse tone double line. There are two holes in a large chrome piece. The distance separating the two holes is the measurement point. The test pattern contains a row of double line structures where the CD is increasing from left to right across the row. A linearity plot can be constructed with this data. Ideally, perfect linearity would be observed. The structure is shown in Figure B.8.

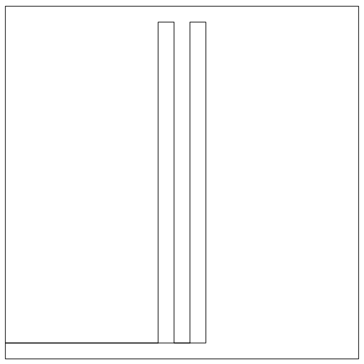


Figure B.8: Inverse double line

B.0.10 Islands

Name	**Type**	**Property**	**Measure**
island	2-D	linearity	width

The island structure is just a small square for which the width is measured. For a row of islands with increasing width, a plot can be constructed. Ideally, perfect linearity would be observed in the plot. The structure is shown in Figure B.9.

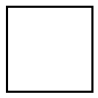


Figure B.9: Island structure

B.0.11 Inverse islands

Name	**Type**	**Property**	**Measure**
inverse island	2-D	linearity	width of hole

The inverse island structure is just a small square opening in a large polygon. The width of the opening should be measured. For a row of inverse islands with increasing width of the hole, a plot can be constructed. Ideally, perfect linearity would be observed in the plot. The structure is shown in Figure B.10.

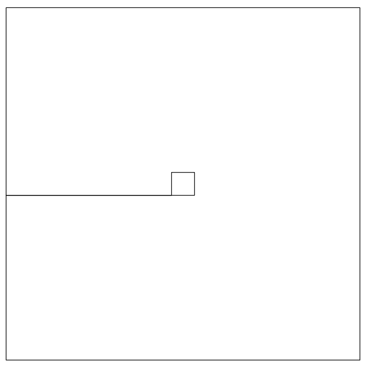


Figure B.10: Inverse island structure

B.0.12 Dense islands

Name	**Type**	**Property**	**Measure**
inverse island	2-D	linearity	center island width

The dense island structure is an array of islands. The width of the center island should be measured. For a row of inverse islands with increasing width of the hole, a plot can be constructed of measured versus target width. Ideally, perfect linearity would be observed in the plot. The structure is shown in Figure B.11.

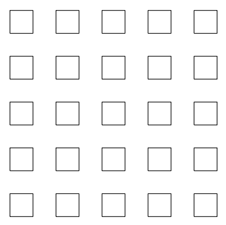


Figure B.11: Dense island structures

B.0.13 Line ends, dense line ends

Name	**Type**	**Property**	**Measure**
line end	2-D	linearity	gap between line ends
dense line end	2-D	linearity	gap between line ends

The line end structure is used to characterize line-end shortening. The minimum distance between the two abutting lines should be measured.

For a row of line ends with increasing separation distance between the lines, a plot can be constructed of measured versus target gap. Ideally, perfect linearity would be observed in the plot. The structure is shown in Figure B.12.

The dense line end structure similar to the line end structure except it contains long lines on either side of the abutting lines. The minimum distance between the two abutting lines should be measured. The structure is shown in Figure B.13.

Figure B.12: Line end structure

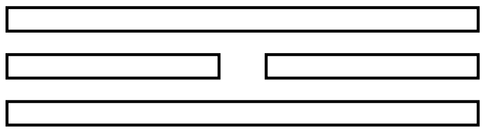


Figure B.13: Dense line end structure

B.0.14 Inverse line ends

Name	Type	Property	Measure
inverse line end	2-D	linearity	between inverse ends

The inverse line end structure is used to characterize 2-D effects. The measurement location is shown in Figure B.14.

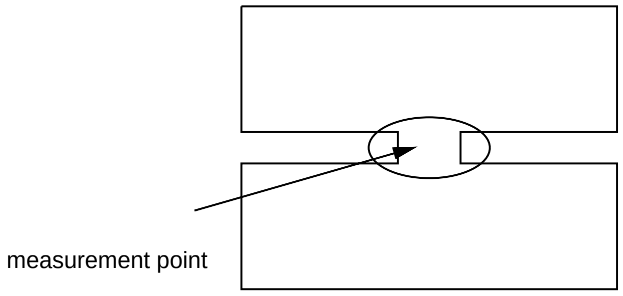


Figure B.14: Inverse line end.

B.0.15 T junction, double T junction

Name	Type	Property	Measure
T junction	1.5-D	general	center line width
double T junction	1.5-D	general	center line width

The T junction contains a single long line in the center, abutting to the center line, there are two lines which are separated by a given distance. The

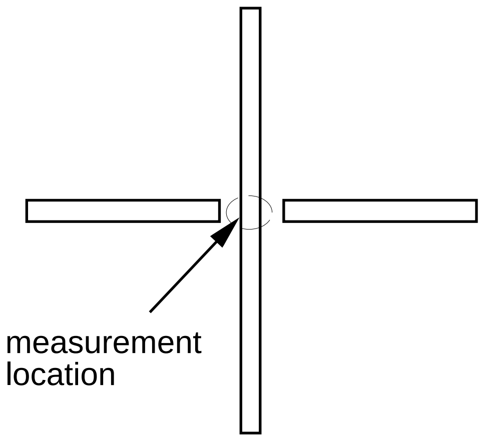


Figure B.15: Line end structure

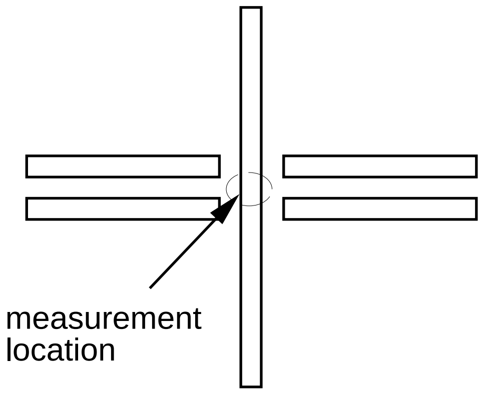


Figure B.16: Dense line end structure

B.0.16 corners, dense corners

Name	Type	Property	Measure
corners	2-D	linearity	outer gap between corner
dense corners	2-D	linearity	outer gap between corner

The corner structures can be used to characterize corner rounding effects.

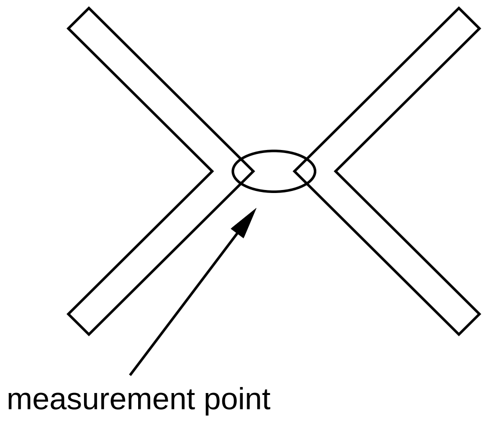


Figure B.17: Corner structure

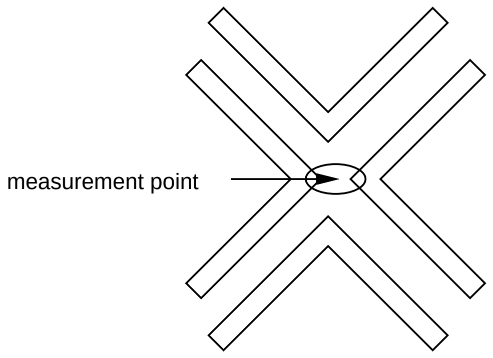


Figure B.18: Dense corner structure

B.0.17 Bridge structures

Name	Type	Property	Measure
bridge	2-D	linearity	width of inside bridge

The bridge structure is used to evaluate 2-D effects. The measurement should be made in the center of the structure, for the width of the "bridge". This is shown in Figure B.19.

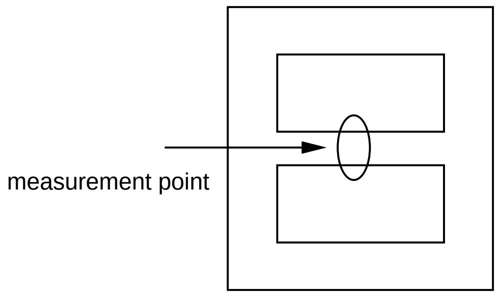


Figure B.19: Bridge structure

Bibliography

[1] M. Born and E. Wolf. *Principles of Optics.* Pergamon Press, London, 1980.

[2] Yan Borodovsky. Impact of local partial coherence variation on exposure tool performance. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2440*, pages pp. 750–770, Santa Clara, 1995.

[3] T. A. Brunner and R. Ferguson. Approximate models for resist processing effects. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2726*, pages 198–207, Santa Clara, 1996.

[4] T. A. Brunner and R. A. Ferguson. Simple models for resist processing effects. *Solid State Technology*, V39 N6:95–103, June 1996.

[5] N. Cobb. Fast mask optimization for optical lithography. Master's thesis, University of California at Berkeley, 1994.

[6] N. Cobb and A. Zakhor. Large area phase-shift mask design. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2197*, pages 348–360, San Jose, 1994.

[7] N. Cobb and A. Zakhor. Fast, low-complexity mask design. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2440*, pages 313–327, Santa Clara, 1995.

[8] N. Cobb and A. Zakhor. Fast sparse aerial image calculation for OPC. In *Proceedings of BACUS Symposium on Photomask Technology Vol 2621*, pages 534–545, Santa Clara, 1995.

[9] N. Cobb and A. Zakhor. A mathematical and CAD framework for proximity correction. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2726*, pages 208–222, Santa Clara, 1996.

[10] B. D. Cook and S. Y. Lee. Fast proximity effect correction: An extension of PYRAMID for thicker resists. *Journal of Vacuum Science Technology B*, 11 No. 6:2762–2767, Dec 1993.

[11] F. H. Dill, A. R. Neureuther, J. A. Tuttle, and E. J. Walker. Modeling projection printing of positive photoresists. *IEEE Transactions on Electron Devices*, ED-22 No. 7:pp.456–464, 1975.

[12] P. D. Flanner. Two-dimensional optical imaging for photolithography simulation. Technical Report Memorandum No. UCB/ERL M86/57, Electronics Research Laboratory, University of California at Berkeley, Jul 1986.

[13] H. Fukuda, T. Terasawa, and S. Okazaki. Spatial filtering for depth of focus and resolution enhancement in optical lithography. *J. Vac. Science Technology B*, 9 No. 6:3113–3116, Dec 1991.

[14] H. Gamo. *Progress in Optics*, volume 3 ed. E. Wolf, chapter 3: Matrix Treatment of Partial Coherence, pages 187–326. North-Holland Publishing Company, 1964.

[15] J. Garafalo, J. DeMarco, J. Bailey, and S. Vaidya. Reduction of asic gate-level line-end shortening by mask compensation. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2440*, Santa Clara, 1996.

[16] K. Harafugi, A. Misaka, N. Nomura, M. Kawamoto, and H. Yamashita. A novel hierarchical approach for proximity effect correction in electron beam lithography. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12 No. 10:1508–1514, Oct 1993.

[17] M. E. Haslam and J. F. McDonald. Transform based proximity corrections: experimental results and comparisons. *Journal of Vacuum Science and Technology, B*, 4 No. 1:168–175, Jan/Feb 1986.

[18] R. Hershel and C. A. Mack. *Lithography for VLSI Electronics–MicroStructure Science* eds. R. K. Watts and N. G. Einspruch, chapter 2: Lumped Parameter Model for Optical Lithography, pages 19–55. Academic Press, New York, 1987.

[19] H.H. Hopkins. On the diffraction theory of optical images. In *Proc. Royal Soc. Series A.*, volume 217 no. 1131, pages 408–432, 1953.

[20] J. C. Jacob, S.Y. Lee, J. A. McMillan, and N. C. MacDonald. Fast proximity effect correction: An extension of pyramid for circuit patterns of arbitrary size. *Journal of Vacuum Science Technology B*, 10 No. 6:3077–3082, Nov/Dec 1992.

[21] D. J. Kim, W. G. Oldham, and A. R. Neureuther. Development of positive photoresist. *IEEE Transactions on Electron Devices*, ED-31 No. 12:pp. 1730–1735, 1975.

[22] S. Y. Lee, J. C. Jacob, C. Chen, J. A. McMillan, and N. C. MacDonald. Proximity effect correction in electon-beam lithography: A hierarchical rule-based scheme-pyramid. *Journal of Vacuum Science Technology B*, 9 No. 6:3048–3052, Nov 1991.

[23] M. D. Levenson, N. S. Viswanathan, and R. A. Simpson. Improving resolution in photolithography with a phase-shifting mask. *IEEE Transactions on Electron Devices*, ED-29 No. 12:pp.1828–1836, 1982.

[24] H. Levinson and W. H. Arnold. *Handbook of Microlithography, Micromachining, and Microfabrication: Volume 1, Microlithography*, chapter 1: Optical Lithography, pages 11–138. SPIE Press, Bellingham, Washington, 1997.

[25] B. J. Lin. Phase-shifting and other challenges in optical mask technology. In *SPIE Symposium on Optical Microlithography*, volume 1496, pages pp.54–76, 1990.

[26] Y. Liu, A. Pfau, and A. Zakhor. Systematic design of phase-shifting masks with extended depth of focus and/or shifted focus plane. *IEEE Transactions on Semicondutor Manufacturing*, V. 6 No. 1:pp.1–21, 1993.

[27] Y. Liu and A. Zakhor. Binary and phase shift mask design for optical lithography. *IEEE Transactions on Semiconductor Manufacturing*, 5 No. 2:pp.138–152, 1992.

[28] Y. Liu and A. Zakhor. Computer aided phase shift mask design with reduced complexity. *Proceedings of SPIE Symposium on Optical Microlithography*, Vol. 1927:pp.477–493, 1993.

[29] Chris A. Mack. Enhanced lumped parameter model for photlithography. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2197*, pages 501–510, San Jose, 1994.

[30] D. Newmark. *Optical Proximity Correction for Resolution Enhancement Technology.* PhD thesis, University of California at Berkeley, 1994.

[31] O. Otto, J. Garafalo, K. Low, C. Yuan, R. Henderson, C. Pierrat, R. Kostelak, S. Vaidya, and P. Vasudev. Automated optical proximity correction-a rules-based approach. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2197*, pages 278–293, San Jose, 1994.

[32] Y. C. Pati, A. A. Ghazanfarian, and R. F. Pease. Exploiting structure in fast aerial image computation for integrated circuit patterns. *IEEE Transactions On Semiconductor Manufacturing*, 10 No. 1:62–74, Feb 1997.

[33] Y. C. Pati and T. Kailath. Phase-shifting masks : Fast automated design and mask requirements. In *SPIE Symposium on Optical Microlithography*, San Jose, 1994.

[34] Y. C. Pati and T. Kailath. Phase-shifting masks for microlithography: Automated design and mask requirements. *Journal of the Optical Society of America A-Optics Image Science and Vision*, 11 No. 9:2438–2452, 1994.

[35] B. Saleh. *Image Recovery – Theory and Application,* ed. Henry Stark, chapter 12: "Image Synthesis: Discovery Instead of Recovery". Academic Press, Orlando, 1987.

[36] B. E. A. Saleh and M. Rabbani. Simulation of partially coherent imagery in the space and frequency domains by modal expansion. *IEEE Transactions on Electron Devices*, ED-29 No. 12:pp.1828–1836, 1982.

[37] R. J. Socha and A. R. Neurether. Role of illumination and thin film layers on the printability of defects. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2440*, Santa Clara, 1995.

[38] J. Stirniman and M. Rieger. Fast proximity correction with zone sampling. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2197*, pages 294–301, Santa Clara, 1994.

[39] J. Stirniman and M. Rieger. Quantifying proximity and related effects in advanced wafer processes. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 2440*, pages 252–260, Santa Clara, 1995.

[40] J. Stirniman and M. Rieger. Spatial-filter models to describe ic lithographic behavior. In *Proceedings of SPIE Symposium on Optical Microlithography Vol 3051*, pages 469–478, Santa Clara, 1997.

[41] K. K. H. Toh. Algorithms for three-dimensional simulation of photoresist development. Memorandum UCB/ERL M90/123, Electronics Research Laboratory, University of California, Berkeley, Dec 14, 1990.

[42] K. K. H. Toh. Two-dimensional images with effects of lens aberrations in optical lithography. Memorandum UCB/ERL M88/30, University of California, Berkeley, May 20, 1988.

[43] University of California at Berkeley. *A User's Guide for SPLATv4.2*, 1993.

[44] Rudolf Murai von Bunau. *Depth of Focus Enhancement in Optical Lithography.* PhD thesis, Stanford Univeristy, 1995.

[45] S. Wittekoek. Optical lithography: Present status and continuation below 0.25 microns. *Microelectronic Engineering*, Vol. 23:pp.43–55, 1994.

[46] E. Wolf. New spectral representation of random sources and of the partially coherent field that they generate. *IEEE Transactions on Electron Devices*, ED-29 No. 12:pp.1828–1836, 1982.