COMPUTER AIDED DESIGN TOOLS FOR PHASE-SHIFT MASKS AND SPATIAL FILTERING

by

David M. Newmark

Memorandum No. UCB/ERL M91/117

18 December 1991

COMPUTER AIDED DESIGN TOOLS FOR PHASE-SHIFT MASKS AND SPATIAL FILTERING

by

David M. Newmark

Memorandum No. UCB/ERL M91/117

18 December 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

COMPUTER AIDED DESIGN TOOLS FOR PHASE-SHIFT MASKS AND SPATIAL FILTERING

by

David M. Newmark

Memorandum No. UCB/ERL M91/117

18 December 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Computer Aided Design Tools for Phase-Shift Masks and Spatial Filtering

David M. Newmark

Department of Electrical Engineering and Computer Sciences,

University of California at Berkeley

ABSTRACT

A computer aided design tool, Mask Analysis System by Computer (MASC), has been developed to automatically generate design graphs of user specified image quality measurements over one-dimensional cut lines and two-dimensional mask areas as a function of both optical system and mask layout variables. MASC is intended to serve as a tool to assist layout designers in determining design rules for phase-shift masks and in examining the printability of mask levels or combinations of mask levels. Several examples of analyses performed using MASC on phase-shift masks are presented. In addition, SPLAT has been extended to include lens filtering effects, and MASC is used to investigate the parameters of a spatial filter introduced by Hitachi which uses two focal points to extend the depth of focus.

The results from our analyses of phase-shift masks indicate that for dark field masks with isolated spaces and Levenson arrays of spaces, a shrink of the mask combined with a bloat of open areas shows promise as a means of designing phase-shift masks if one is willing to sacrifice the area between light regions. Results obtained by examining several typical mask patterns show that an overall shrink of the mask does not require special design rules for each type of feature, but better scaling could result by appropriately developing new rules for different patterns. A probe of phase transitions indicates that a phase transition of 0.6 λ/NA is the optimum length to achieve the highest intensity throughout the phase transition region. A new mask pattern for contact chains is demonstrated to have a high intensity slope for a pitch of 1.2 λ/NA and below. Examination of a phase-shift DRAM layout shows that it is not as sensitive to alignment as might be expected by examination of the mask itself. The linewidth control of the DRAM can be improved by modifying non-critical areas such as the passing Poly line. The relationship between two parameters in the Hitachi spatial filter function is determined to be $\theta=2\pi\beta$, and this relationship is used to show that a 25% improvement in depth of focus requires at least a factor of 5 reduction in the clear field intensity while 100% improvement requires a factor of 10 reduction.

Dec. 18, 1991

Table of Contents

Chapter 1

Introduction

1.1 Overview of Non-Conventional Optical Lithography

Device scaling has placed ever greater demands on optical lithography which necessitates the simulation of aerial images and three-dimensional resist profiles to understand the trade-offs involved in new mask, illumination, and resist techniques. The traditional means of reducing the linewidth has been to buy new steppers which typically operate at lower wavelengths and higher numerical apertures. There are problems with both these approaches. Movement to lower wavelengths requires the development of reliable light sources and lenses at those frequencies. In addition, new photoresist and mask technologies are needed. These problems have renewed interest in non-conventional techniques which can be applied to existing steppers and extend the ultimate limits of optical lithography as well.

One of these techniques arose out of the work of Mark Levenson at IBM. Levenson suggested that a phase-shift mask would cause the spill-over between features to subtract and provide the means to print much smaller features on existing lithography equipment.[1] Another technique, known as spatial filtering, is an even older technology than phase-shift masks; it was used in photography in the early 1950's.[2] Recently, researchers at Hitachi have incorporated spatial filters into their lithography simulation programs. By choosing a special class of filter functions, they were able to increase the depth of focus.[3] Such a technique would be useful in systems with a low depth of focus resulting from the use of an objective lens with a high numerical aperture. A third related idea is the use of modified illumination to emphasize small features.[4]

All of these techniques show promise for extending optical lithography, and decisions must be made regarding which ones to pursue for the next generation of devices. Of course, each method offers many options which have different costs for tooling and mask making. For example, the decision to use phase angles other than 180 degrees in phase-shift masks increases mask costs. The decision is particularly complex because all of these techniques have the common lim-

itation that they do not affect all features in the same manner. While experimentation is viewed as the ultimate means of verification, it is far too slow and expensive to examine every alternative. On the other hand, simulation programs such as SPLAT (Simulation of Projection Lens Aberrations via TCCs) can generate mask pattern images in a few seconds. Coupling these image simulation programs with post processors to extract measures of image quality can provide an effective means of evaluating these technologies.

Based on this need, a number of mask analysis and design systems are beginning to emerge. Initial work by K.K.H. Toh was directed towards the development of SPLAT, a fast and robust optical aerial image simulation program.[5] Further work led to the development of an analysis program which could run SPLAT for several iterations and extract pertinent information from the image cut lines.[6] Industrial versions of SPLAT have been developed leading to products such as DEPICT-2™, TMA's lithography simulation software.[7] In addition, researchers at AT&T and IBM have started developing other automated approaches for assessing technology issues.[8][9]

The work presented in this thesis generalizes these existing approaches by implementing a computer aided design tool to automatically generate design graphs of user specified image quality measurements over two-dimensional mask areas as a function of both optical system and mask layout parameters. The extracted measurements can be made along one-dimensional cutlines through the mask, over the entire mask, or between mask levels. In addition, this thesis reports on the capability to include lens filtering effects in the SPLAT simulation program. This new system is called Mask Analysis System by Computer (MASC). The program is intended to assist layout designers in deciding upon design rules and determining the printability of a particular mask level or combination of mask levels which form a given device.[10][11] Mask patterns from the OCT/VEM/RPC CAD tools can be linked to MASC via the PROSE Technology CAD environment.[12] A graphical user interface and additional data extraction methods can easily be added.

A number of technology issues of current interest can be efficiently addressed with MASC. For example, through a shrink analysis, patterns which commonly occur on masks are used to examine the scalability of various feature types under the same design rules. Phase transitions,

which are vital for clear field mask patterns, are investigated to examine how the transition length is affected by scaling. Finally, a shrink and bloat analysis method is applied to a new contact pattern, Merged Outrigger Alternating Contact String (MOACS), to show its improved characteristics when compared to Levenson contacts.

The critical dimension of the aerial image for a feature, such as a polysilicon line, changes due to both imaging and layout effects as it jogs, for example, around contacts. The associated mask which defines the active area, such as a trench mask, also prints differently than it appears on the layout. As a result, comparisons of the images of the two mask levels under various conditions such as defocus and misalignment must be made to assess the process tolerance of the gate width. Layout designers may also wish to resize non-critical features and introduce phase-shifted non-printing bands to improve the process latitude of critical regions.

Chapter 2 describes the architecture and important features of MASC and relates them to commands issued by the user. The first four sections of Chapter 3 describe some analysis examples using the cut line analysis methods. Layout patterns are presented for determining basic rules for phase-shift masks. Patterns which commonly occur on masks are used to examine the scalability of various feature types under the same design rules. Since phase transitions are vital for clear field mask patterns, 90 and 60-120 degree phase-shift regions are investigated to examine how the transition length is affected by scaling. The printability of contacts is vital to any phase mask technology, so a shrink and bloat concept is applied to Levenson contacts, and a Merged Outrigger Alternating Contact String (MOACS) is shown to combat the defocus problem at the end of contact chains. The final section of the chapter describes a two-dimensional analysis example in which a simplified DRAM layout is analyzed for misalignment tolerance and modified for greater process latitude. Chapter 4 describes the addition of spatial filtering to SPLAT and presents design graphs which show the trade-offs made to obtain the depth of focus increases. For example, when the Hitachi spatial filter is used to print contacts, the depth of focus is increased at the expense of a decrease in the clear field intensity and an increase in the side lobe intensity.

Chapter 2

Mask Analysis System by Computer (MASC)

2.1 Overview of MASC

The purpose of MASC is to automatically generate design graphs of user specified image quality measurements across one-dimensional cut lines and over two-dimensional mask areas as a function of both optical system and mask layout parameters. The design graphs extracted from cut lines can be used to represent trade-offs in image quality, so a user can then apply criteria suited to a specified process for determining new design rules appropriate to phase-shifting. Other design graphs taken over the entire two-dimensional mask area can be utilized to determine the printability of a particular mask level or combination of mask levels which form a device. The implementation of MASC is presented from several points of view. First, an architectural view considers the organization and capabilities of the program. Second, the user view relates the program features to the actual commands used in running MASC. Third, the implementation view explains some programming details. Further specifics on the operation of the program are given in the user guide in Appendix A.

2.2 Architectural View

The block diagram of MASC is in Fig. 2.1 on the following page. This diagram consists of four main blocks, an input block, two output blocks, and a processing block. In the input block, the mask definition, projection printer parameters, and mask operations are specified while in the output blocks, the results are written to disk. The organization of the process block allows for two main paths which results in the single and multivariable iteration schemes indicated on Fig. 2.1. The modules are SPLAT, 1D and 2D data extraction, and the depth of focus calculation. The blocks labeled modification of mask or projection system are part of the single or multivariable integration schemes, so they are explained in that section. SPLAT is an separate optical lithography image simulation program.[5] The data extraction consists of measuring a variety of image

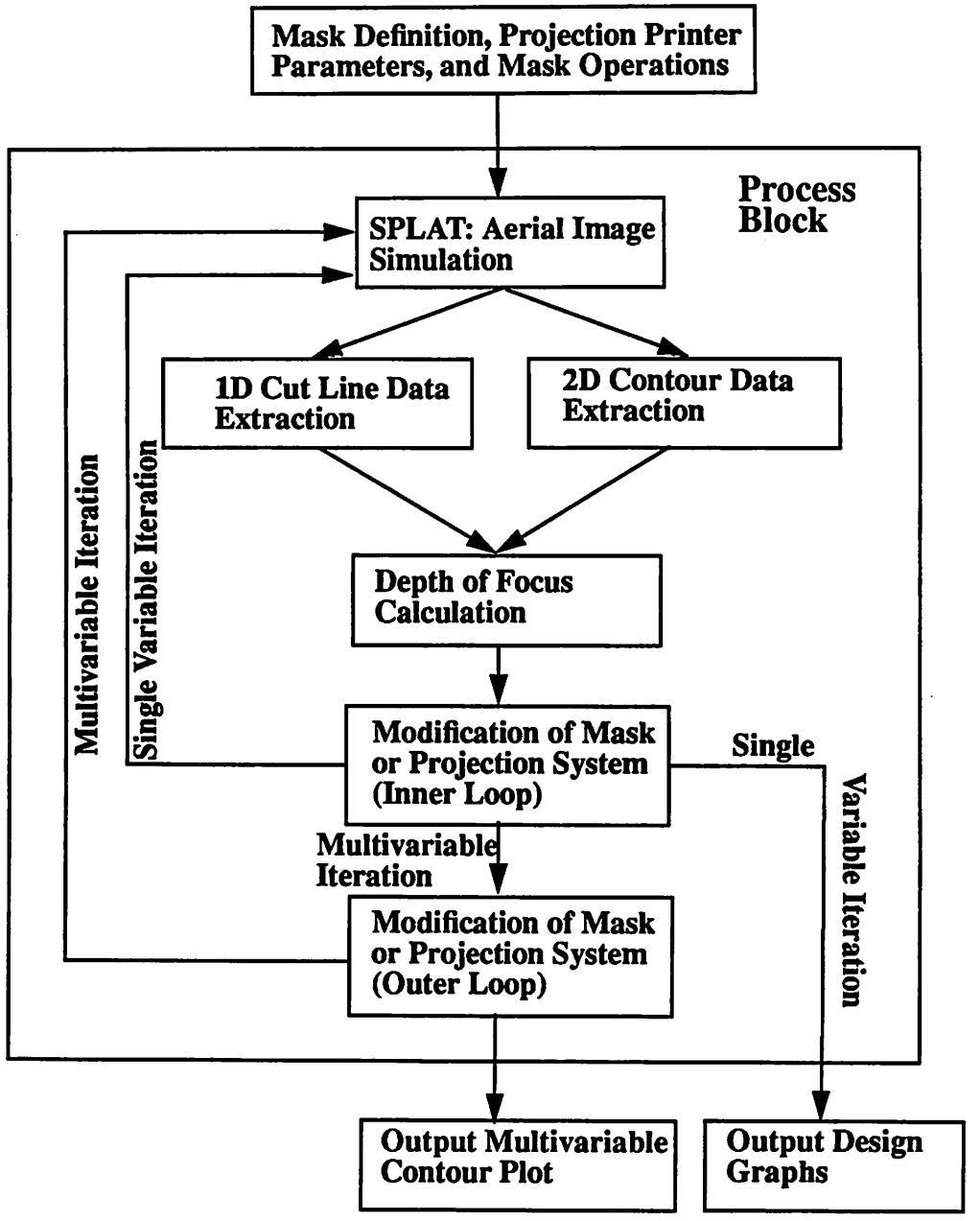


Figure 2.1 Block diagram of MASC. All the processing capabilities are shown in this diagram. In practice, the options are executed only if appropriate commands are issued in the Command File. Two paths through the block diagram indicate the single and multivariable iteration schemes.

characteristics on one-dimensional cut lines and two-dimensional contours. The depth of focus calculation is used to find the depth of focus of an image.

2.2.1 Mask and Projection Printer Variables

In the input block, MASC reads the mask geometry, projection printer parameters, and operations on the mask and printer variables. The projection printer parameters include partial coherence (σ), numerical aperture (NA), wavelength (λ), defocus, and lens aberrations. An operation on one of these parameters consists of simply increasing it. For this reason, all parameters are set up as variables, thus, allowing their values to be changed after each iteration. Similarly, mask operations permit individual boxes to be modified in a variety of ways. Fig. 2.2 has examples of the shrink, bloat, and move operations on a particular mask. These operations arise from the simple modification of one or more numbers which describe the boxes that make up the mask. For instance, a shrink multiplies all numbers describing the mask by a constant shrink factor which results in a global scaling of the mask. A bloat allows open areas to be stretched at the expense of dark regions in order to increase the light intensity. Finally, a move pushes features closer together. These particular operations are mainly useful for studying phase-shifting masks.

2.2.2 Single and Multivariable Iteration Schemes

Before looking at the specifics of each module in the process block, the two iteration schemes, which represent the two major paths through the block diagram, are discussed. The single variable iteration scheme uses only the inner loop on the block diagram of Fig. 2.1. The first step, as discussed above, consists of reading in the arbitrary mask pattern, the projection printer parameters, and the mask or projection system variables which will be modified after each iteration. In the second step, SPLAT, an optical lithography aerial image simulation program, is called to calculate the image intensity due to the pattern and produce the cut line plots through selected areas of the mask and a contour plot of the entire image. In the third step, the 1D data extraction algorithm finds the performance information from the cut lines while the 2D data extraction functions calculate performance criteria from the contour plot. In the fourth step, the mask or projec-

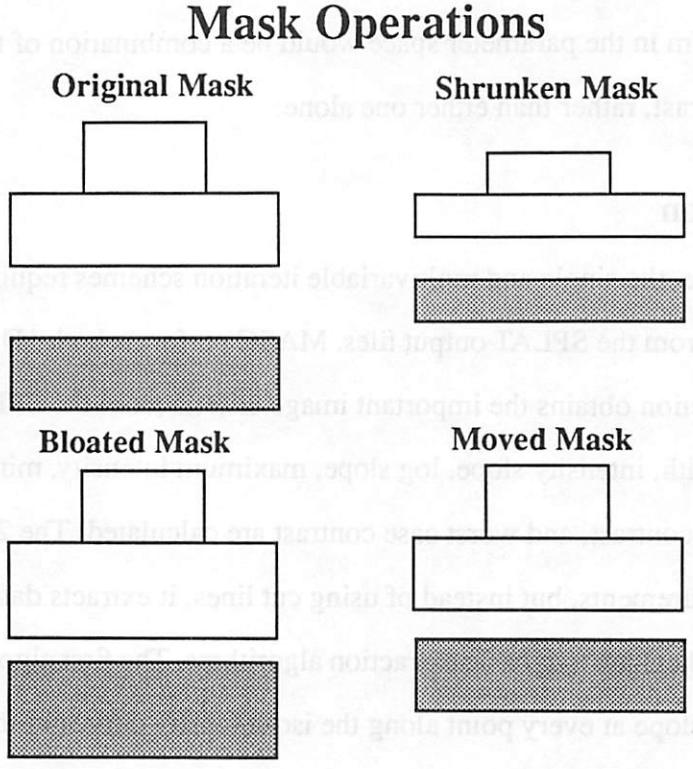


Figure 2.2 The analysis methods of shrinking, bloating and moving.

tion system is modified according to the variables specified in the input. The loop continues by either following the single variable iteration path back to SPLAT, or it ends by writing the results requested by the user.

In the multivariable iteration scheme, both the inner loop and the outer loop are executed; thus, MASC can perform a multivariable analysis in which the parameter space of two variables is examined. During a multivariable iteration, two variables, var1 and var2, are designated as changing. The inner loop consists of the same path discussed for the single iteration scheme with var1 being used as the single variable. However, instead of writing the output design graphs and exiting, the multivariable analysis enters the outer loop in which var1 is reset and var2 is modified. The process continues until the entire parameter space is covered. Another feature of the multivariable analysis is that the extracted image measurements can be combined to yield an objective function. For example, one could plot numerical aperture in the x-direction, wavelength in the y-

direction and have an objective function of 0.5 * depth of focus + 0.5 * contrast in the z-direction. In this way, the maximum in the parameter space would be a combination of two measurements, depth of focus and contrast, rather than either one alone.

2.2.3 Data Extraction

As discussed above, the single and multivariable iteration schemes require measurements of the vital image criteria from the SPLAT output files. MASC performs both 1D and 2D data extraction. The 1D data extraction obtains the important image criteria from the SPLAT cut line plots. The linewidth, spacewidth, intensity slope, log slope, maximum intensity, minimum intensity, user specified intensity, contrast, and worst case contrast are calculated. The 2D data extraction also makes image measurements, but instead of using cut lines, it extracts data from one or more SPLAT contour plots. There are two basic extraction algorithms. The first algorithm calculates the intensity slope and log slope at every point along the iso-intensity curves in the contour image. The second algorithm finds the intersection of the iso-intensity curves of two masks and returns the ones within a user specified linewidth analysis box. Both routines then record the slope along the curves and calculate the minimum, maximum, and average slope. If only two curves are in the data set, the minimum linewidth is also determined.

2.2.3.1 1D Data Extraction

As discussed above, both iteration schemes require measurements of vital image criteria from the SPLAT output files. The 1D data extraction is used to obtain the important image analysis information from the SPLAT cut line plots. Using a specific example, Fig. 2.3 illustrates the important measurements that are taken from the cut line plots. The linewidth and spacewidth are defined as the distance between equi-intensity levels, and the intensity slope is defined as the slope at each equi-intensity point on the cut line. The log slope is then calculated by dividing the intensity slope by the intensity level. These measurements are shown in Fig. 2.3c for the 0.3 intensity level, but they are actually taken at 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6 intensity levels. Fig. 2.3d indicates the other image quality measurements: maximum intensity, minimum intensity, user

1D Data Extraction

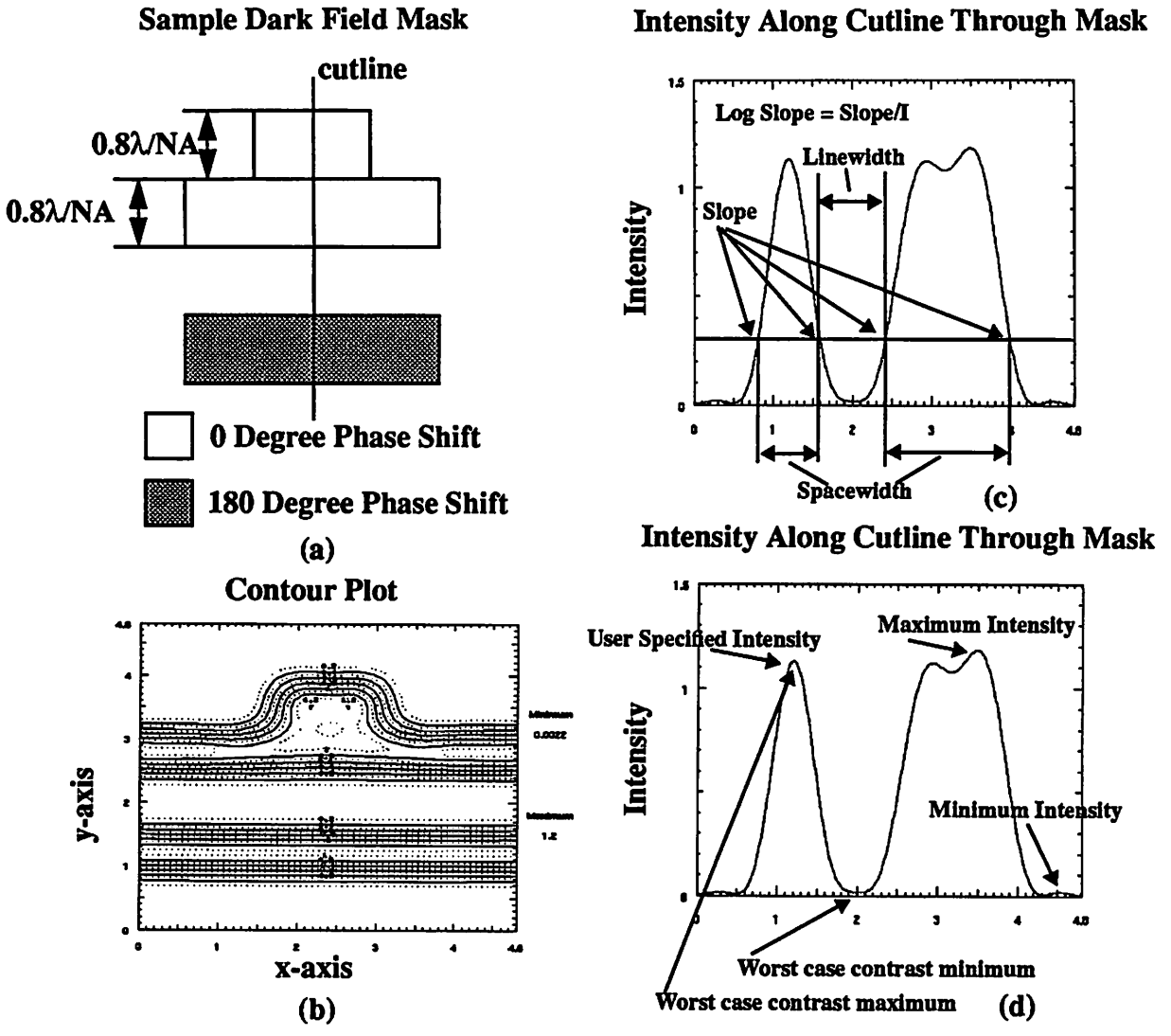


Figure 2.3 Example of MASC 1D data extraction for a dark field mask pattern. A sample dark field mask is shown in (a) while (b) shows the iso-intensity contour plot of the pattern. (c) Illustration of the definition of linewidth, spacewidth, intensity slope, and log slope on a particular feature. These variable are measured at the 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6 intensity levels, but only the line for the 0.3 level is shown. (d) Illustration of the definition of maximum intensity, minimum intensity, user specified intensity, and worst case contrast. Note that the worst case contrast is measured between the user specified intensity and the largest minimum located at most 1.2 λ/NA from the maximum (for 0.8 λ/NA features).

specified intensity, and worst case contrast. The maximum and minimum intensity are the largest and smallest intensity values across the cut lines. These values are used to calculate contrast $\left(\frac{I_{max}-I_{min}}{I_{max}+I_{min}}\right)$. The user specified intensity is an intensity value recorded at a location that was specified by the user. The worst case contrast is defined as the contrast between a user specified intensity and the largest minimum located at most 1.2 λ/NA away (for 0.8 λ/NA spacewidths).

2.2.3.2 2D Data Extraction

The first algorithm for 2D data extraction calculates the slope along the 0.3 iso-intensity curves of the contour image. Fig. 2.4 summarizes the method. The 3D surface data, which makes up the SPLAT image, consists of z-values of the 3D surface arranged on a rectangular grid. Fig. 2.4a illustrates this format, where the dots represent arbitrary z-values. The intensity slope is calculated at every z-value in the image by using a 5-point operator. The five points consist of the point where the slope is desired and its four nearest neighbors. As shown in Fig. 2.4a, these five points comprise 4 triangles. Since a triangle uniquely defines a plane, the unit normal to each of these triangles can be calculated. Once the unit normal is found, the gradient vector can be determined and its magnitude gives the maximum slope of the triangle. By averaging the slopes of the four triangles, the intensity slope at the central point is estimated. After applying the 5-point operator to all points in the image, the Contour program is used to find all the curves which comprise the 0.3 intensity level.[13] Fig. 2.4b shows an example of these curves; they were extracted from the Poly layer of the DRAM layout above. In Fig. 2.4c, a 2D interpolation scheme is used to estimate the slope at all points on the curves. P is the location at which the slope must be estimated, and C1 through C4 are the nearest points in the rectangular grid of Fig. 2.4a. The sum of the distances from P to C1 through C4 is given in Equation 1 where d is the total distance and d1 through d4 are the distances from P to C1 through C4 respectively. Equation 2 is the weighted average or

$$d = d1 + d2 + d3 + d4 \qquad \text{(EQ 1)}$$

2D Slope Extraction

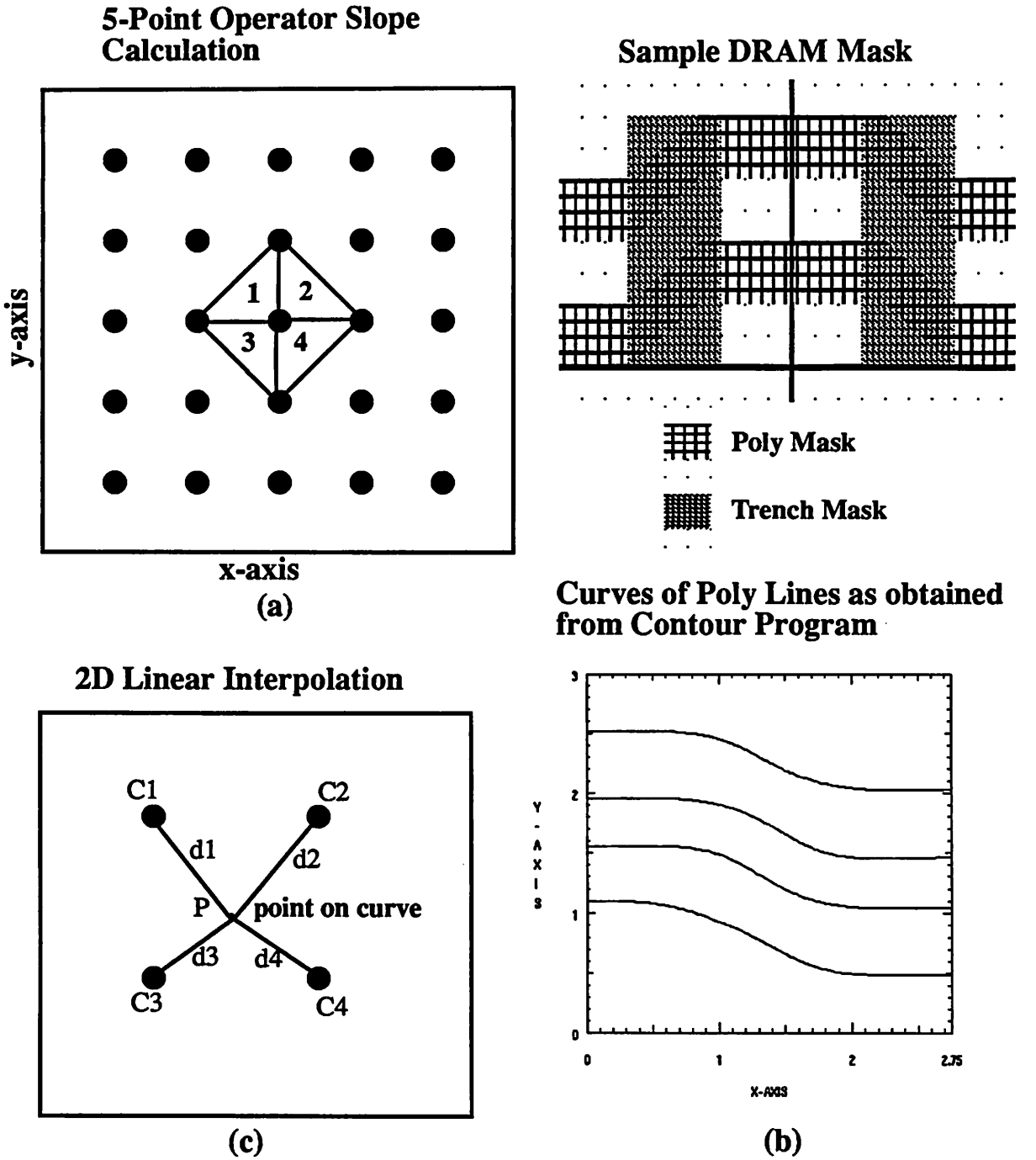


Figure 2.4 (a) Illustration of 5-point operator. The dots represent the z-values of the 3D image arranged in a rectangular grid. The slope at the central point is found by averaging the slopes of each of the four triangles. (b) The contour program is used to extract the iso-intensity curves from the SPLAT contour image. (c) A 2D linear interpolation is used to find an estimate of the slope along the curves in (b).

2D linear interpolation of the slope. The subscript s denotes the intensity slope at the point. Once

$$P_s = \frac{d1}{d}C1_s + \frac{d2}{d}C2_s + \frac{d3}{d}C3_s + \frac{d4}{d}C4_s \qquad \text{(EQ 2)}$$

this information is known, all the intensity slope values along the curve as well as the maximum, minimum, and average slopes are recorded. The log slope is calculated from the slope measurements by dividing by the intensity.

The second algorithm finds the intersection of the iso-intensity curves of two masks and returns the ones within a user specified linewidth analysis box. The algorithm utilizes a method of determining whether a point is inside or outside a polygon. This method is illustrated in Fig. 2.5a. The idea is to draw a segment from the point in the polygon to infinity. If this segment intersects an odd number of edges on the polygon, then the point is inside the polygon. If there are an even number of intersections, the point is outside. This method is applied to the intersection of the two mask layers shown in Fig. 2.5b. The algorithm finds the sections of the Poly curves which are outside the trench but inside the linewidth analysis box. The curves resulting from this intersection are shown in Fig. 2.5c. Using the slope calculation above, the slope along the extracted curves is recorded along with the maximum, minimum, and average slope. In addition, if only two curves are extracted, the minimum distance between them is calculated.

2.2.4 Depth of Focus Calculation

After the data extraction, MASC can execute a depth of focus calculation. However, finding the depth of focus can significantly prolong the run time, so it is calculated only if the user requests the information. The depth of focus is found using such variables as intensity slope at the 0.3 intensity level, contrast, or maximum intensity. Defocus typically degrades all of these variables, and the depth of focus is defined as the defocus level at which the given variable is degraded by a specified amount. In MASC, the depth of focus is determined using an algorithm described in the pseudo-code in Table 1. SPLAT is initially run to determine the base value for the particular variable which will be used for the defocus calculation. This is called "base_variable"

Intersection Algorithm

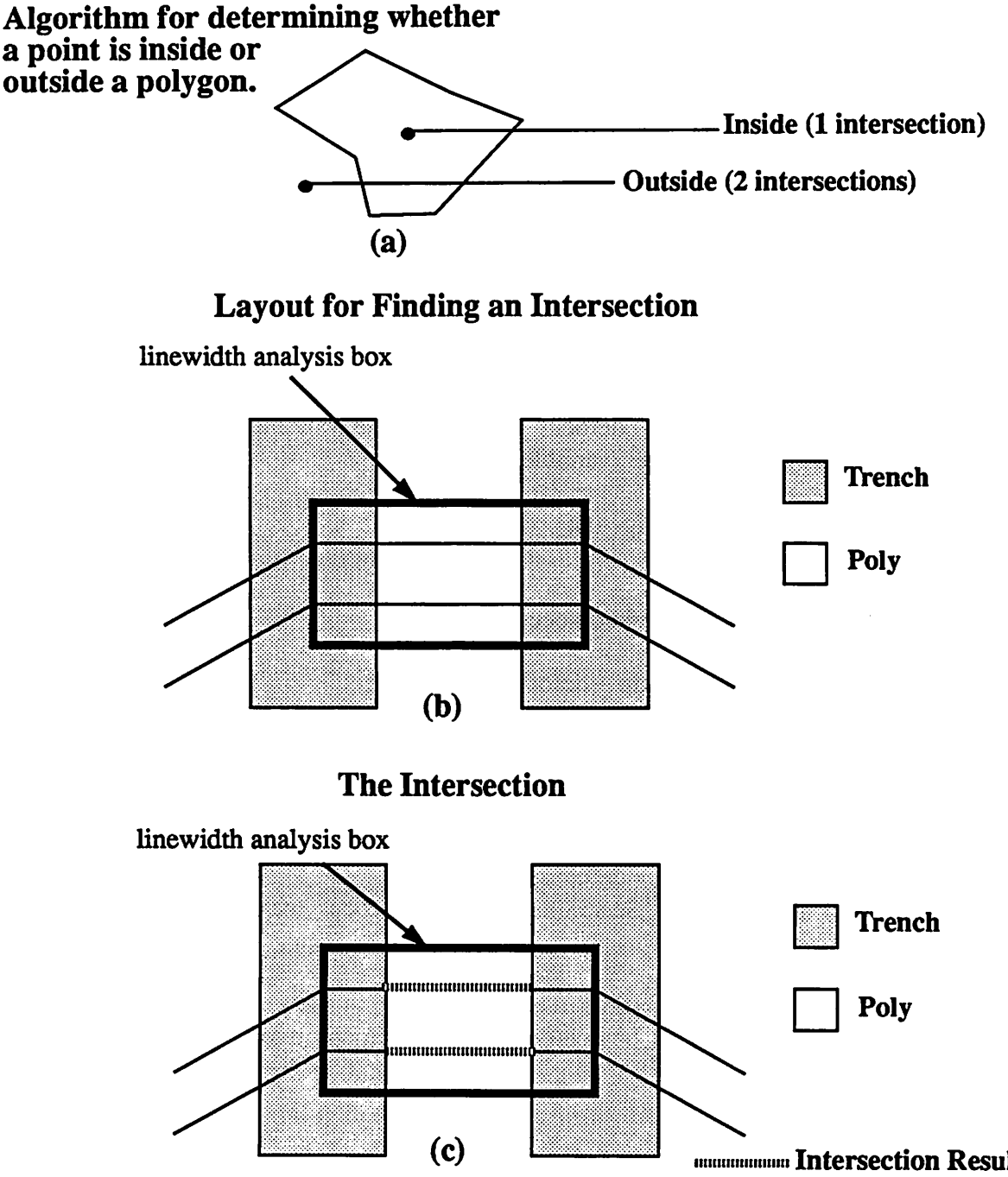


Figure 2.5 Outline for the extraction of curves from the intersection of the Poly and Trench masks. (a) Illustrates how to determine whether a point is inside or outside a polygon. (b) This layout is used to demonstrate the intersection technique. (c) Shows the curves resulting from the intersection.

in the pseudo-code. The change in a variable with increasing defocus can now be referenced to this value to determine its degradation. The user specifies the maximum acceptable degradation; this variable is called "change" in the code. After the setup run, the defocus is incremented until the degradation in the variable is greater than "change". Then, using the bounds on the defocus as determined above, a binary search is performed to find the depth of focus more precisely.

Table 1: Pseudo-Code for Depth of Focus Calculation

```
Depth_of_Focus();
{
    Run_Splat (base_variable);
    defocus = 0;
    defocus_variable = base_variable;
    while ((base_variable - defocus_variable) / base_variable) < change do {
        Run_Splat (defocus_variable, defocus++);
    }
    Binary_Search(0.0,defocus);
}

Binary_Search(defocus_min,defocus_max);
{
    defocus = (defocus_min + defocus_max) / 2.0;
    Run_Splat (defocus_variable);                    /* SPLAT uses defocus when run */
    if ((base_variable - defocus_variable) / base_variable) = change +- error
        return defocus
    else {
        if ((base_variable - defocus_variable) / base_variable) > change {
            defocus_max = defocus;
            return Binary_Search(defocus_min,defocus_max);
        }
        else {
            defocus_min = defocus;
            return Binary_Search(defocus_min,defocus_max);
        }
    }
}
```

2.3 User View

The user must organize three basic units: the tool, the mask geometry, and the design task, to specify the execution of the options discussed above. First, the printer variables are listed in the tool unit. Second, the mask geometry and the modification of its parameters are specified in the mask geometry unit. Third, the single and multivariable iteration scheme, the data extraction, and the depth of focus calculation are entered in the design task unit. The actual file inputs use a command file to specify the tool and the design tasks while a separate geometry file contains the mask geometry. Specific examples and all commands are listed in Appendix A.

The tool is specified by issuing commands which set up the projection printer variables and indicate how to modify them after each iteration. The commands are similar to those used by SPLAT, such as **lambda** and **na**. A recent addition to SPLAT allows the use of a Hitachi spatial filter (see Appendix B); the variables associated with this function are indicated using the command, **hitachi_spf**.

The mask geometry is established in the geometry file. Using the **area, B,** and **T** commands, the user can indicate the area of the mask and the location and size of boxes and triangles. In addition, several parameters denote how the mask is changed after each iteration. For example, the edges of every box can be moved to achieve a shrink or bloat. To shrink the entire mask by a given factor, the **shrink_factor** command is included in the command file. The numbers describing the mask are nominally in CIF units. The value of 1 CIF unit is determined by the **cif_scale** command. All numbers associated with the geometry are divided by the scale factor before being sent to SPLAT. Thus, if the scale is 50, 1 CIF unit corresponds to 0.02 um.

To accomplish a design task, additional commands are included in the command file. These commands can be organized into three parts. In the first part, the iteration method is indicated. By default, MASC uses the single variable iteration scheme. The **iterations** command specifies the number of times to execute the inner loop. However, if a multivariable iteration is desired, the **multi_variable** command is used. The multivariable analysis automatically executes the outer loop the same number of times as the inner loop. The second part details the data extraction

options. The 1D cut line measurements are made by default, but cut lines through the image must be listed by using the **cutline** command. Any number of cutlines can be entered. The **contour_analysis** and **intersection** commands are used to instruct MASC to extract the 2D image measurements. In addition, a depth of focus calculation can be requested by utilizing the **depth_of_focus** command. The third part consists of the plotting commands. Plotting any 1D or 2D measurement for a single variable analysis can be accomplished using the **plot** command. Any number of **plot** commands can be defined. For the multivariable analysis, the **weight_function** command is used. This command allows formulation of an objective function for the contour plot. Any extracted variable from the 1D or 2D measurements may be used in the **weight_function** command. By including several of these statements in the command file, an objective function of many measurements can be produced. Again, the details of these commands are located in Appendix A.

The actual indication of which variable to modify during a design task is implicit for a single variable iteration and explicit for the multivariable iteration. In practice, this means that for a single variable iteration, all variables are modified, but only those variables which have nonzero entries for increases are actually changed. However, for a multivariable iteration, the variable that is modified is indicated in the **multi_variable** command. This structure requires the user to exercise care in specifying how parameters change since multiple parameters can be modified during each iteration of the single variable analysis. This may lead to confusing results.

Once the tool, mask geometry, and design task are specified in the command and geometry files, MASC can be run. If the command file is saved in "command" and the geometry file is saved in "geometry", the analysis is executed by typing, "masc command geometry". The parser will send its interpretation of the input commands to standard output and then begin the iteration loops. Once the SPLAT runs are complete, the measurements made by MASC are written to disk. For a single variable analysis, any measurement requested by **plot** is written to a separate file. If several cut lines are specified, then the information is saved for every one. These files are saved in a format suitable for plotting by Drawplot.[13] The results from a multivariable iteration are writ-

ten to a single file using the objective function specified by the **weight_function** commands. This file is plotted with Contour.[13]

2.4 Implementation View

The code for MASC implements the blocks of Fig. 2.1 in modules which are called from within a main process loop. The nine basic modules are: 1) the parser, 2) run SPLAT, 3) 1D data extraction algorithms, 4) 2D algorithm - slope, 5) 2D algorithm - intersection, 6) depth of focus calculation, 7) mask and projection printer modification, 8) single variable plots, and 9) multivariable plots. The communication of the tasks and measurements between these modules takes place through external variables. Consequently, additional image analysis code should follow this convention. The new code can be created as a separate module or incorporated into the existing modules. However, the plotting and parser modules must be modified to incorporate the additional commands necessary to implement new functions or measurements.

After the parser initializes the data structures with the tool, mask geometry, and design task information, the main process loop calls modules two through seven for the number of iterations specified in the command file. The run SPLAT module first writes a SPLAT input file to the directory, and then it makes a system call to SPLAT. The SPLAT output is written to the directory for post-processing by the extraction and depth of focus modules. The mask and projection printer parameters are modified in module seven. When the iterations are complete, modules eight and nine output the results.

MASC consists of 5000+ lines of C code written for the unix operating system. In addition to the program itself, SPLAT Version 4.0 and Contour Version 3.0 are required for its operation. The single variable iteration results are written to files that can be plotted with Drawplot Version 2.2 while the multivariable iteration results are written to a file that is plotted with Contour. Tentative interfaces exist to PROSE through Caltech Intermediate Form (CIF).[12] This interface will be made more robust in the future.

Chapter 3

Analysis Examples

By combining the iteration schemes, the 1D or 2D data extraction, and the mask or projection printer variables, which were discussed in Chapter 2, it is possible to create a wide variety of analysis methods through which MASC can contribute to the understanding of layout images. The examples which follow concentrate on the issues of phase-shift mask design. In sections 3.1 to 3.4, the analysis methods combine the single variable iteration scheme, 1D data extraction, and mask and projection printer variables. In particular, the Levenson type of phase-shift mask appears to be very promising, so finding basic design rules for its application to irregular mask patterns is the focus of the first four examples. First, the shrink and bloat concept for printing both isolated and arrayed features is considered. Second, a variety of dark field mask patterns that would appear on typical VLSI masks are analyzed. Third, phase transitions, vital for clear field masks, are examined. Fourth, a new style for producing contact chains is introduced and compared to other methods. In the final example, section 3.5, the analysis method combines the multivariable iteration scheme and the 2D data extraction options to analyze a phase-shifted DRAM for enhanced printability. Together these studies give an overall idea of how MASC works as well as indicating some general means for applying phase-shifting to masks.

3.1 The Shrink and Bloat

A possible method for designing phase-shift masks is to shrink the mask and then bloat the open areas. The shrink and bloat idea takes advantage of the intrinsic value of phase-shifting, which is its ability to resolve objects spaced closely together if they are of opposite phase. Fig. 3.1 shows the mask chosen to study the effect. It contains an isolated space and an array of spaces. Both must print acceptably on a given mask since random logic chips require isolated spaces and closely spaces features. Fig. 3.1 also gives an example of what it means to apply the shrink and bloat operation to a mask. Essentially, we take the original mask, shrink it by a given factor as

shown in (b) and then bloat all the clear regions as in (c). This bloat step effectively increases the light intensity through the clear areas at the expense of eliminating some of the dark region between the light areas. In theory, allowing increased light through the clear areas results in improved printability since peak intensity can be maintained even when features are close together.

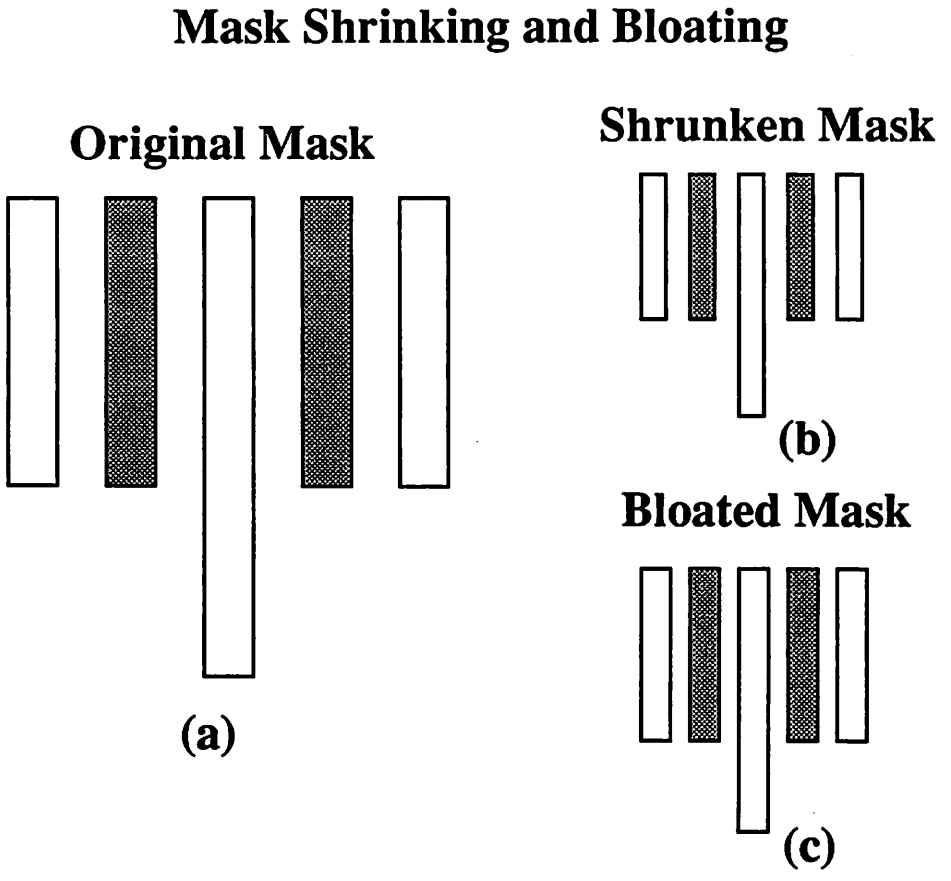


Figure 3.1 Mask pattern for studying isolated spaces and arrays of spaces. (a) The original mask consists of an isolated space as well as arrays of spaces. (b) The shrunken version of the original mask. (c) The same mask with all the clear areas bloated to increase the light intensity.

In order to further demonstrate the concept of shrinking and bloating, we made a design graph of an isolated space in order to pick a minimum spacewidth that can be tolerated by the process. Assuming 80% of clear field intensity is necessary for printing, we choose a minimum spacewidth of 0.56 λ/NA for σ equals 0.5 and 0.53 λ/NA for σ equals 0.3 as shown in Fig. 3.2 on the next page. For these open areas widths, we then produce design graphs to see how the charac-

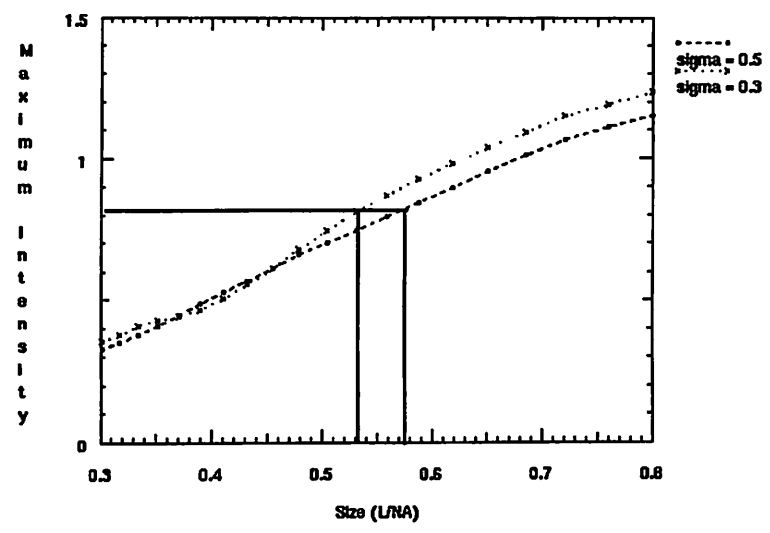


Figure 3.2 Design graph of the maximum intensity through an open area on the mask of varying width.

teristics of an isolated space change when the space is brought closer to other open areas. Fig. 3.3 and 3.4 exhibit the resulting design graphs for the cases where σ equals 0.5 and 0.3 respectively. Both of these design graphs show that even for pitches of only 0.8 λ/NA, which corresponds to a 0.56 or 0.53 λ/NA open area and a 0.24 or 0.27 λ/NA dark region on the mask, the intensity and contrast are as good as those for the isolated case. In effect, the resulting mask could be obtained by shrinking it overall by one-half and then bloating the light areas by 0.16 or 0.13 λ/NA.

The problem, of course, is that the light to dark area ratio varies considerably when one compares the arrayed case to the isolated case. For example, the light region varies 15% over a pitch of 1.6 λ/NA to 0.8 λ/NA, but the dark area variation is tremendous. Depending on whether positive or negative resist is used, these variations have a different significance. For example, in negative resist, the line of resist variation would be only 15%, and this might be acceptable for many areas of the chip. However, there would be a large variation in the space between lines which could cause problems. For the case of positive resist, the difficulty is reversed. The space between the line of resist has only a 15% variation, but the linewidth variation is large. If the line-

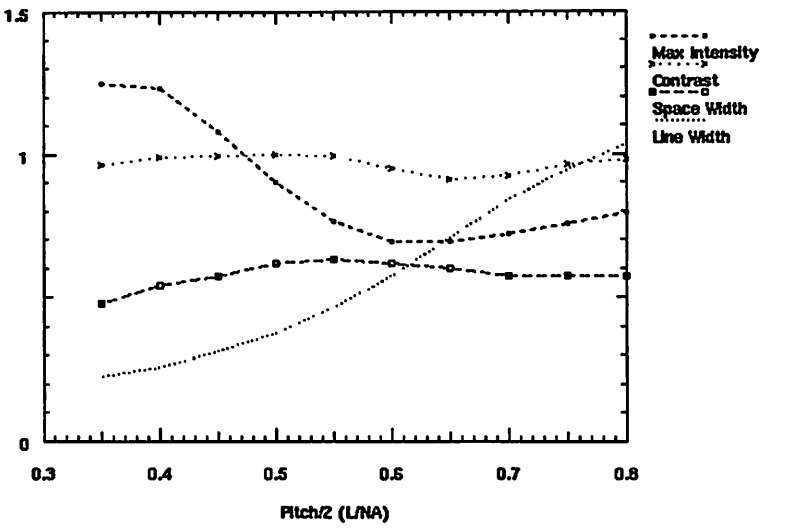


Figure 3.3 Intensity, contrast, space and linewidth for an open area width of 0.56 λ/NA when placed in an array of similar sized spaces. The pitch has been changed resulting in higher packing density for the spaces.

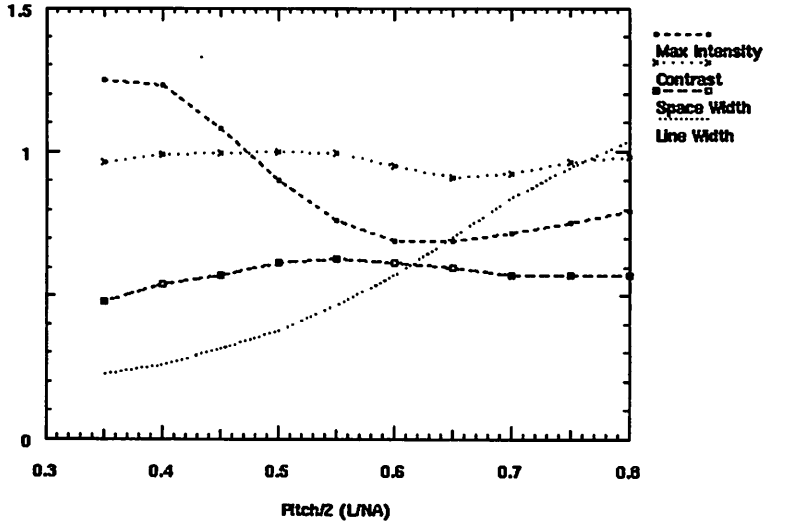


Figure 3.4 Design graph of intensity, contrast, space and linewidth for a spacewidth of 0.53 λ/NA.

width variation is important in certain areas of a chip, those regions would have to be designed more carefully and the shrink and bloat method must be applied more judiciously. However, for the majority of cases, the most important criterion for production is the need for continuous lines, which is not a problem.

3.2 Irregular Dark Field Mask Patterns

In addition to shrinking and bloating, we examine several types of typical patterns found on VLSI masks. Fig. 3.5 shows the masks simulated to determine how different patterns respond to scaling. The cutline through the masks indicate the four cases: "space", "contact", "elbow", and "end". Fig. 3.6 is the design graph for this analysis; it is a plot of the minimum intensity and worst case contrast for a minimum spacewidth varied from 0.8 λ/NA to 0.3 λ/NA. The line with open squares shows the "space" mask without phase-shifting as a reference. The lines with shaded and dark squares are simulations from the "contact" and "spaces" masks with phase-shifting while the lines with open and shaded diamonds are results from the simulation of the "end" and "elbow" masks. As one can easily see from the design graph, the other simulated masks never behave worse than the "spaces" case. If we take the spaces mask as the worst case reference, this means that for these irregular patterns, phase-shifting can be applied without developing new design rules for each layout.

There are also interesting variations of the basic design rules used to pack certain features tighter with phase-shifting. One such case is the foreshortening of an end near a phase-shifted feature. Fig. 3.7 shows the foreshortening for the normal and phase-shift cases. The normal case shows more shortening than the phase-shift case. One can determine the minimum size for the feature if the amount of linewidth variation that can be tolerated by the process is known. For example, for 20% line variation, the normal case can be shrunk to 0.5 λ/NA while the phase-shift mask can be 0.43 λ/NA

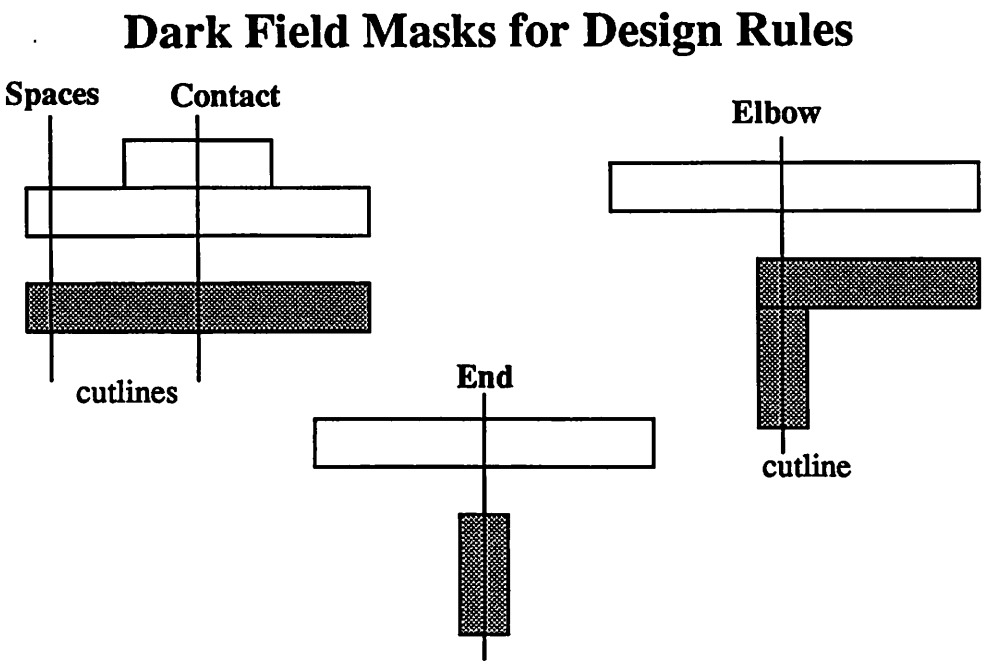


Figure 3.5 These dark field masks are used as examples for determining scaling and design rules. The four mask types represented here are: spaces, contact, elbow, and end.

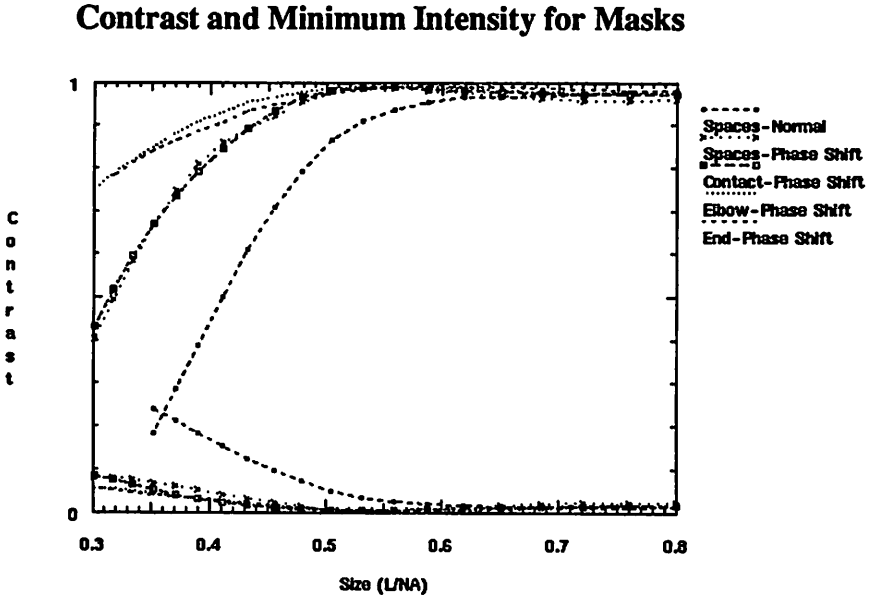


Figure 3.6 Design graph of contrast and minimum intensity for the dark field masks in Figure 3.5. The measurements were made through the cut lines indicated on that figure.

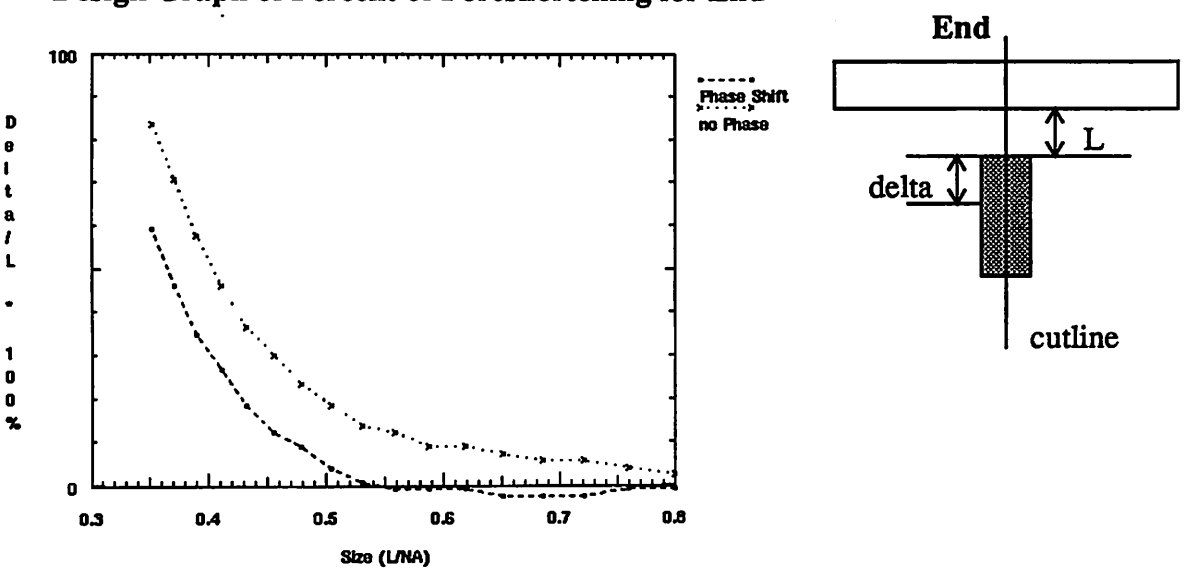


Figure 3.7 Design graph for the end mask. The effect shown here is the end foreshortening caused by shrinking the mask. The percentage of foreshortening is calculated by finding delta, the amount of foreshortening, and dividing by the current spacewidth.

3.3 Phase Transitions

A more difficult family of patterns occurs on clear field masks where phase transitions must be considered. In order to see how phase transitions behave during scaling, we look at 90 degree and 60-120 degree transitions for a variety of open area widths and phase transitions lengths. Fig. 3.8 presents the mask used to analyze the 90 degree phase transition lengths and shows the design graph. Interestingly, an optimum transition length of 0.6 λ/NA produces the least decrease in intensity regardless of the open area width. Fig. 3.9 demonstrates similar results for the 60-120 degree phase transition. Note, however, that each phase section requires a transition length of 0.6 λ/NA to achieve the minimum loss of intensity; this leads to a total transition length of 1.2 λ/NA for the 60-120 case.

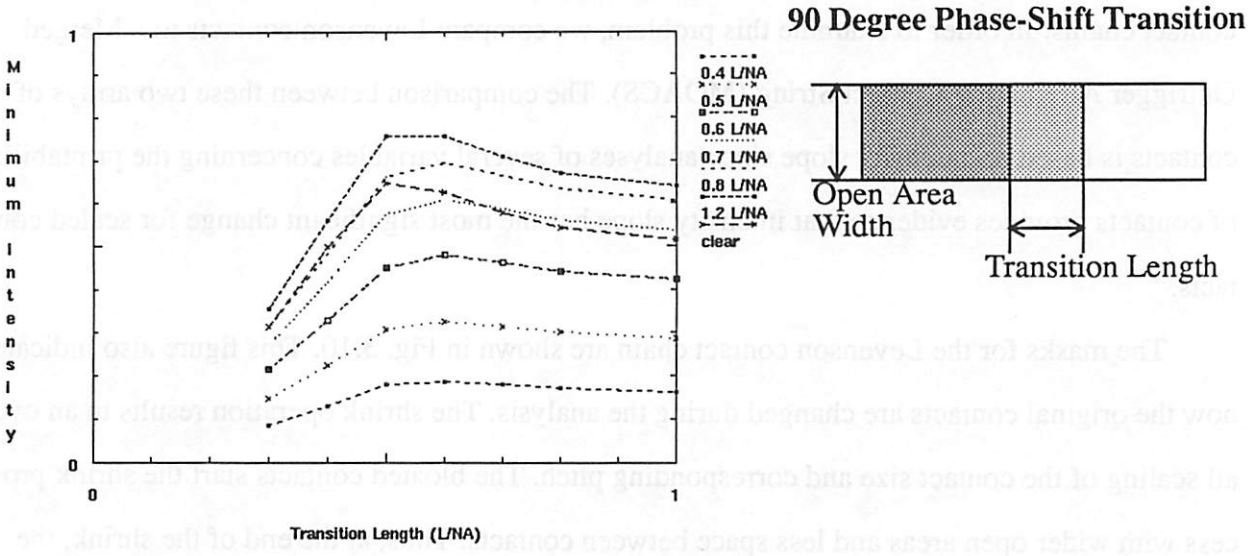


Figure 3.8 Design graph of minimum intensity in an open area resulting from a 90 degree phase transition. The minimum intensity is calculated for a variety of open area widths where the transition length is the length of the 90 degree section. At approximately 0.6 λ/NA, the minimum intensity is greatest.

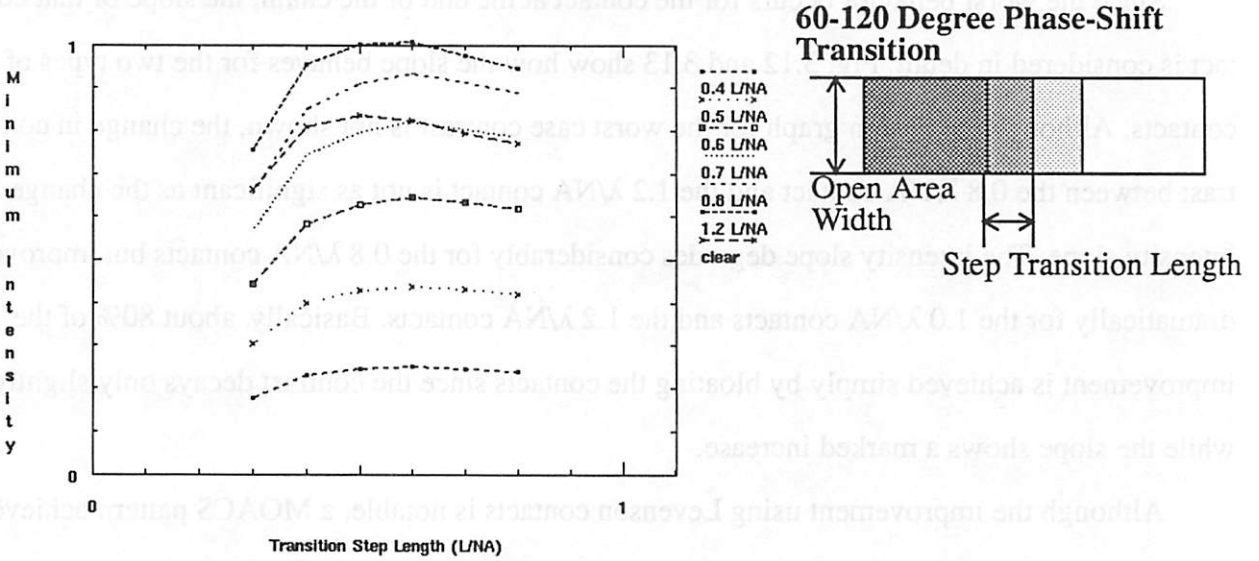


Figure 3.9 Design graph of minimum intensity in an open area resulting from a 60-120 degree phase transition. The minimum intensity is calculated for a variety of open area widths. At a step transition length of 0.6 λ/NA, the minimum intensity is at its greatest value. This corresponds to a full transition length of 1.2 λ/NA.

3.4 Arrayed Contacts

Printing arrayed contacts with phase-shifting can be rather difficult, especially at the end of contact chains. In order to examine this problem, we compare Levenson contacts to a Merged Outrigger Alternating Contact String (MOACS). The comparison between these two arrays of contacts is based on intensity slope since analyses of several variables concerning the printability of contacts provides evidence that intensity slope has the most significant change for scaled contacts.

The masks for the Levenson contact chain are shown in Fig. 3.10. This figure also indicates how the original contacts are changed during the analysis. The shrink operation results in an overall scaling of the contact size and corresponding pitch. The bloated contacts start the shrink process with wider open areas and less space between contacts. Thus, at the end of the shrink, the contacts are larger than before, but they are spaced at the same pitch. The actual study uses three shrink and bloat cases. The initial contact size was 0.8 λ/NA, 1.0 λ/NA, and 1.2 λ/NA. Fig. 3.11 indicates a similar set of masks for a MOACS. In this pattern, non-printing phase bands are added around the contact array.

Since the worst behavior occurs for the contact at the end of the chain, the slope of that contact is considered in detail. Fig. 3.12 and 3.13 show how the slope behaves for the two types of contacts. Although the design graph for the worst case contrast is not shown, the change in contrast between the 0.8 λ/NA contact and the 1.2 λ/NA contact is not as significant as the change in intensity slope. The intensity slope degrades considerably for the 0.8 λ/NA contacts but improves dramatically for the 1.0 λ/NA contacts and the 1.2 λ/NA contacts. Basically, about 80% of the improvement is achieved simply by bloating the contacts since the contrast decays only slightly while the slope shows a marked increase.

Although the improvement using Levenson contacts is notable, a MOACS pattern achieves better defocus tolerance. Fig. 3.14 shows the results of a defocus analysis of Levenson contacts for different σ. The slope of the 0.8 λ/NA isolated contact is shown for reference. The results demonstrate that defocus of 1.5 R.U. degrades the intensity slope for the Levenson case. How-

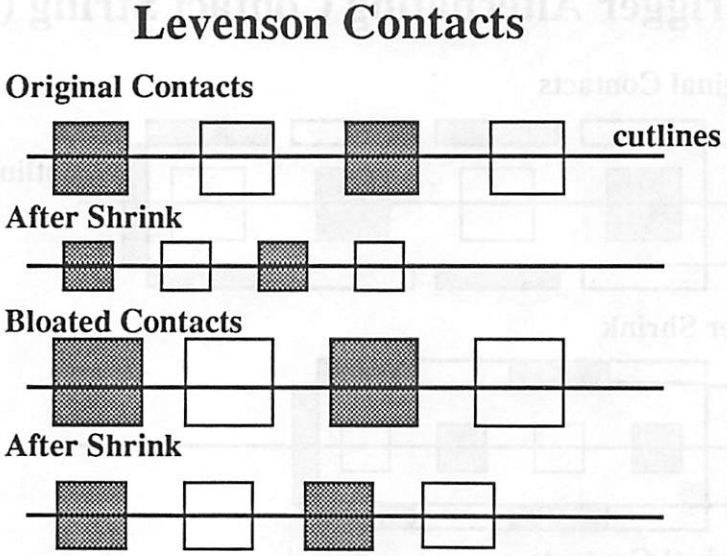


Figure 3.10 Mask pattern for Levenson contacts. The two cases shown here are the original and scaled contacts and bloated and scaled ones. The actual design graphs are generated using two different bloated contact sizes.

ever, the intensity slope of a MOACS, as shown in Fig. 3.15, can be as large as the intensity slope of the isolated contact even with defocus. Thus, the MOACS method shows improved intensity slope for defocus. It also has better contrast than the Levenson approach. This indicates that arrayed contacts can easily be printed on a 1.2 λ/NA pitch even with severe defocus.[10]

Merged Outrigger Alternating Contact String (MOACS)

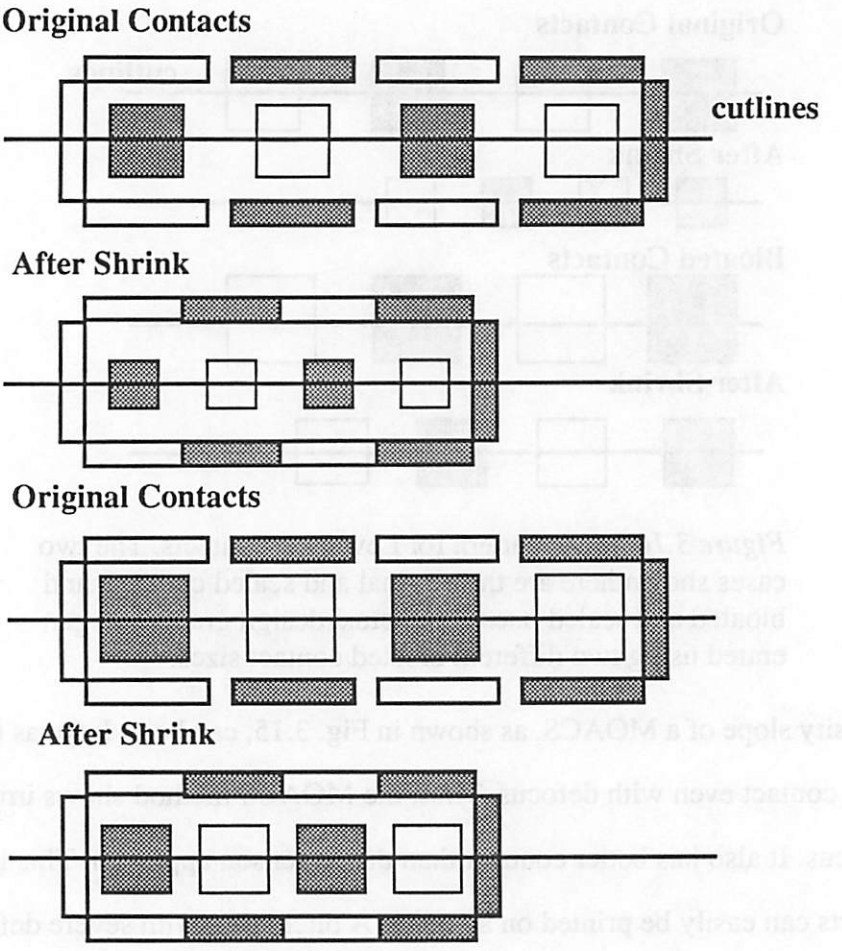


Figure 3.11 Mask pattern for a MOACS. The two cases shown here are the original and scaled contacts and the bloated and scaled ones. The actual design graphs are generated using two different bloated contacts.

End Contact Slope for Levenson Contacts

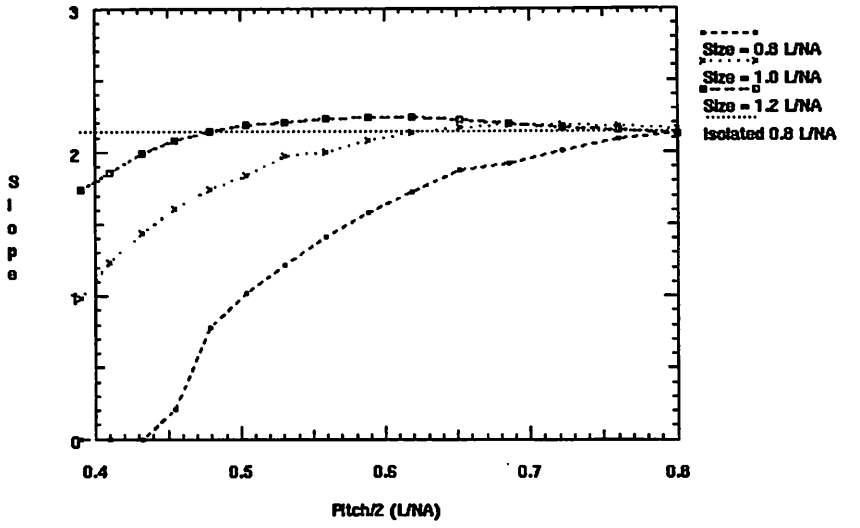


Figure 3.12 Design graph of the intensity slope for the left-most contact in the Levenson contact chain. Two different bloat factors are utilized in the simulations. The intensity slope for an isolated 0.8 λ/NA contact is shown for reference.

End Contact Slope for MOACS Contacts

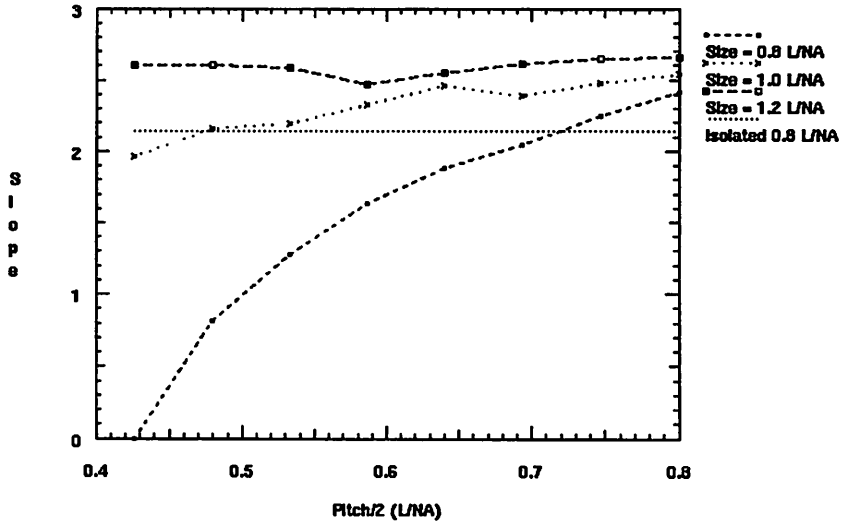


Figure 3.13 Design graph of the intensity slope for the left-most contact in the MOACS. Again the 0.8 λ/NA contact is shown for reference.

End Contact Slope for Levenson Contacts with Size = 1.2 λ/NA

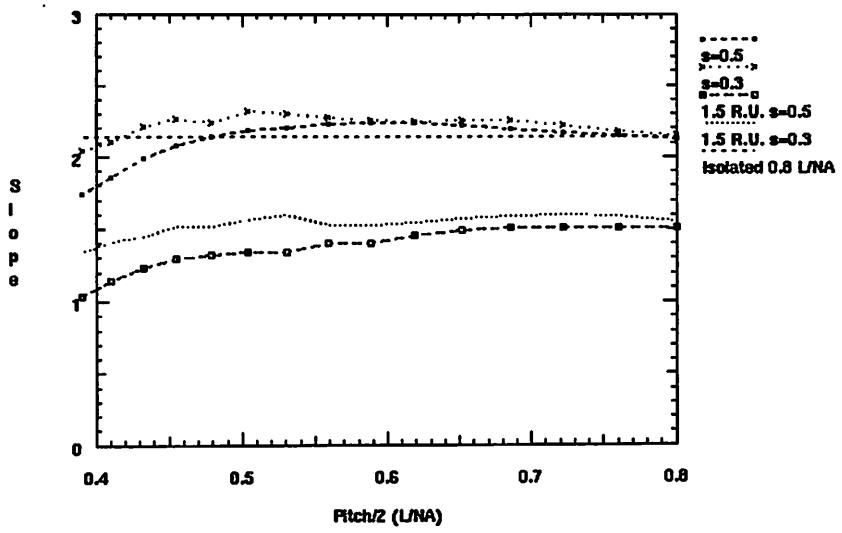


Figure 3.14 Design graph of the intensity slope for the 1.2 λ/NA bloated Levenson case for two value of σ and two defocus values. The slope degrades significantly with defocus and σ has little effect on restoring the intensity slope.

End Contact Slope for MOACS, Size = 1.2 λ/NA

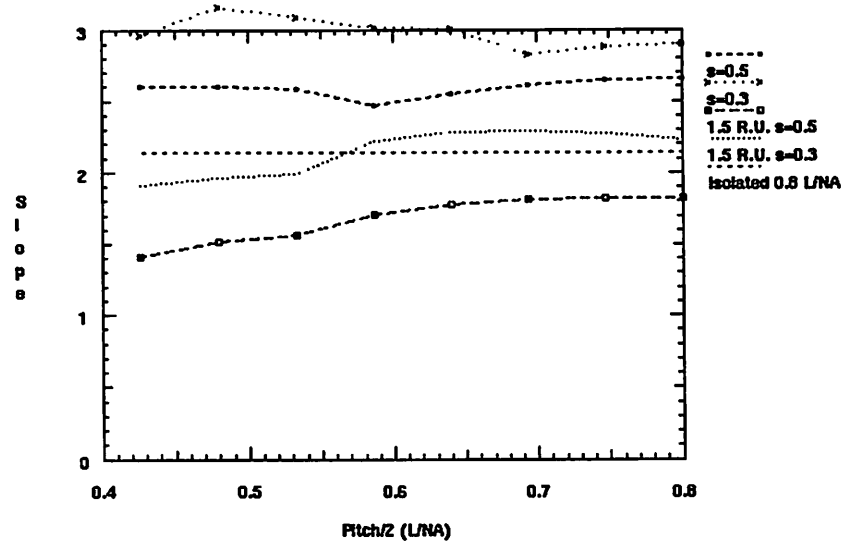


Figure 3.15 Design graph of the intensity slope for the 1.2 λ/NA bloated MOACS for two values of σ and two focus values. The intensity slope is larger than the slope of the isolated 0.8 λ/NA contact when σ is equal to 0.3.

3.5 DRAM

Due to their periodic structure, DRAMs may benefit enormously from phase-shifting. However, certain issues arise which may cause problems. To investigate these difficulties and illustrate the 2D data extraction capabilities and multivariable iteration schemes available with MASC, the phase-shifted DRAM in Fig. 3.16 is investigated. The first problem with this DRAM mask is that as the trench layer is misaligned with respect to the poly layer, the transistor gate, which is formed by the intersection of the bottom Poly layer with the trench mask, is subject to change in minimum width. Fig. 3.17 shows that the linewidth extracted using the 0.3 intensity contour from the image of the poly and trench levels is less sensitive to misalignment than might be predicted from the overlap of the mask geometries themselves. A second problem inherent in this DRAM layout is that the intensity slope on the lower edge of the Poly gate is not adjacent to a feature of the opposite phase, and, consequently, it may vary more than desired. Thus, a multivariable modification of the Poly mask is examined to improve the linewidth control of the lower Poly line. Since the upper Poly linewidth is not critical in this region, its bottom edge can be bloated as one variable. The space around the contact allows room for a 0 phase non-printing band, and its separation from the gate is taken as another variable. Fig. 3.18. shows that the average slope on the 0.3 contour in the gate region is improved by bloating the passing Poly linewidth by about 0.2 λ/NA and by locating the phase band at a separation of 0.3 λ/NA. The resulting mask is in Fig. 3.19.

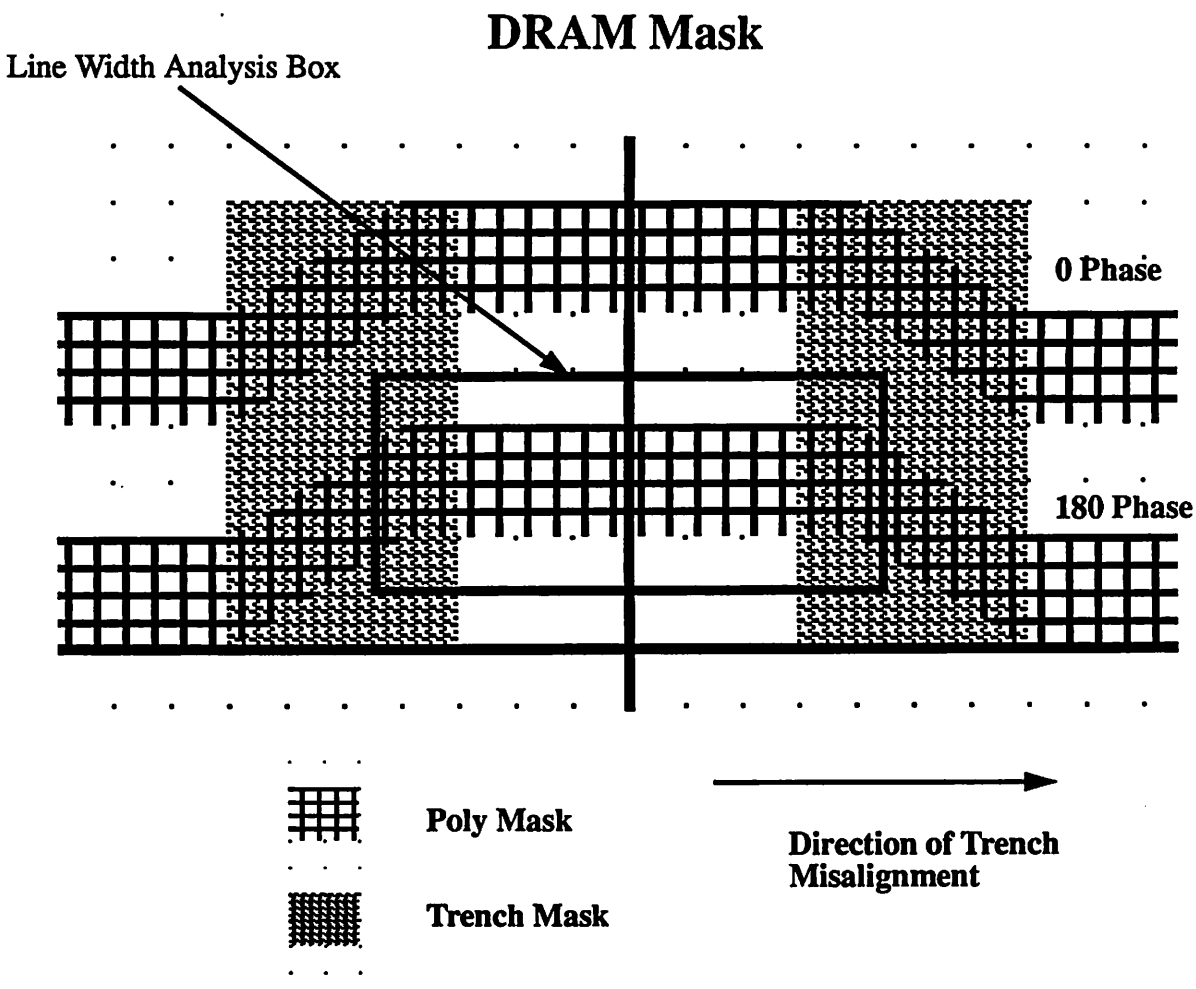


Figure 3.16 Simplified DRAM mask. The bottom poly layer forms the gate for this cell. The linewidth of the poly lines is 0.5 λ/NA. The contact is not shown.

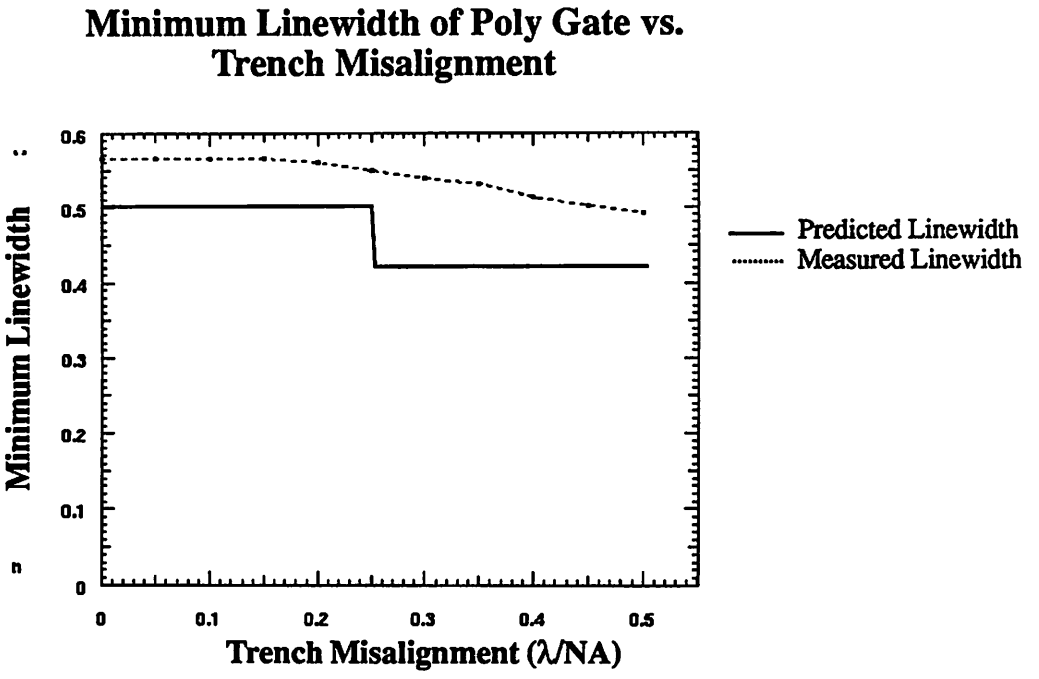


Figure 3.17 Design graph of minimum linewidth of the poly gate as the trench is misaligned to the right. The predicted and measured linewidths are shown.

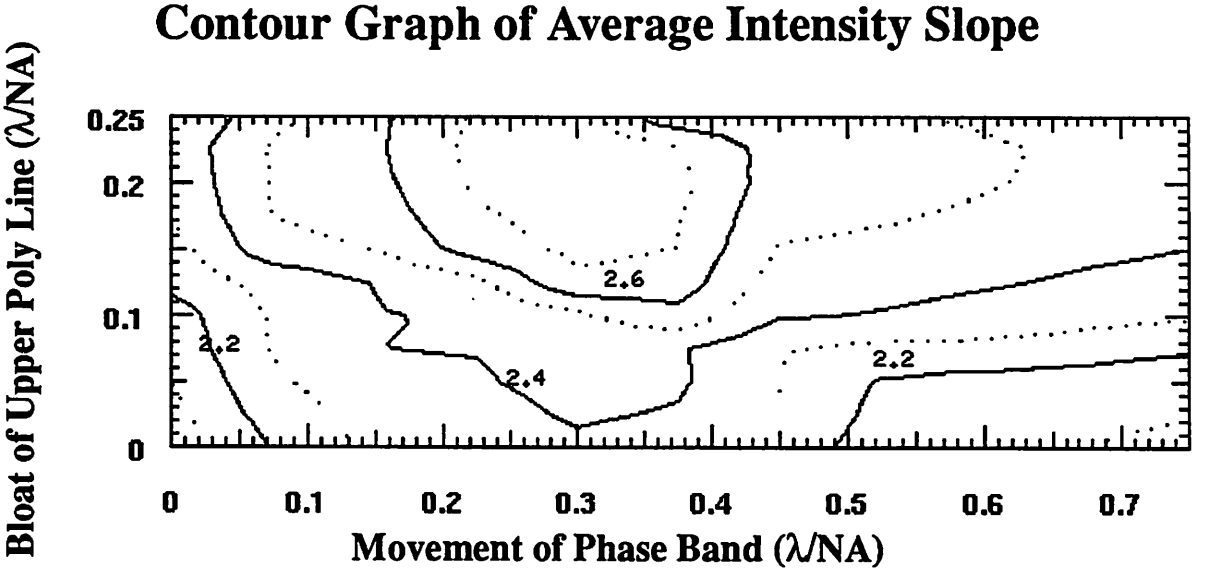


Figure 3.18 Contour graph of average intensity slope on both top and bottom edges of the poly gate as the phase band is moved and the top poly line is bloated.

Modified DRAM Layout

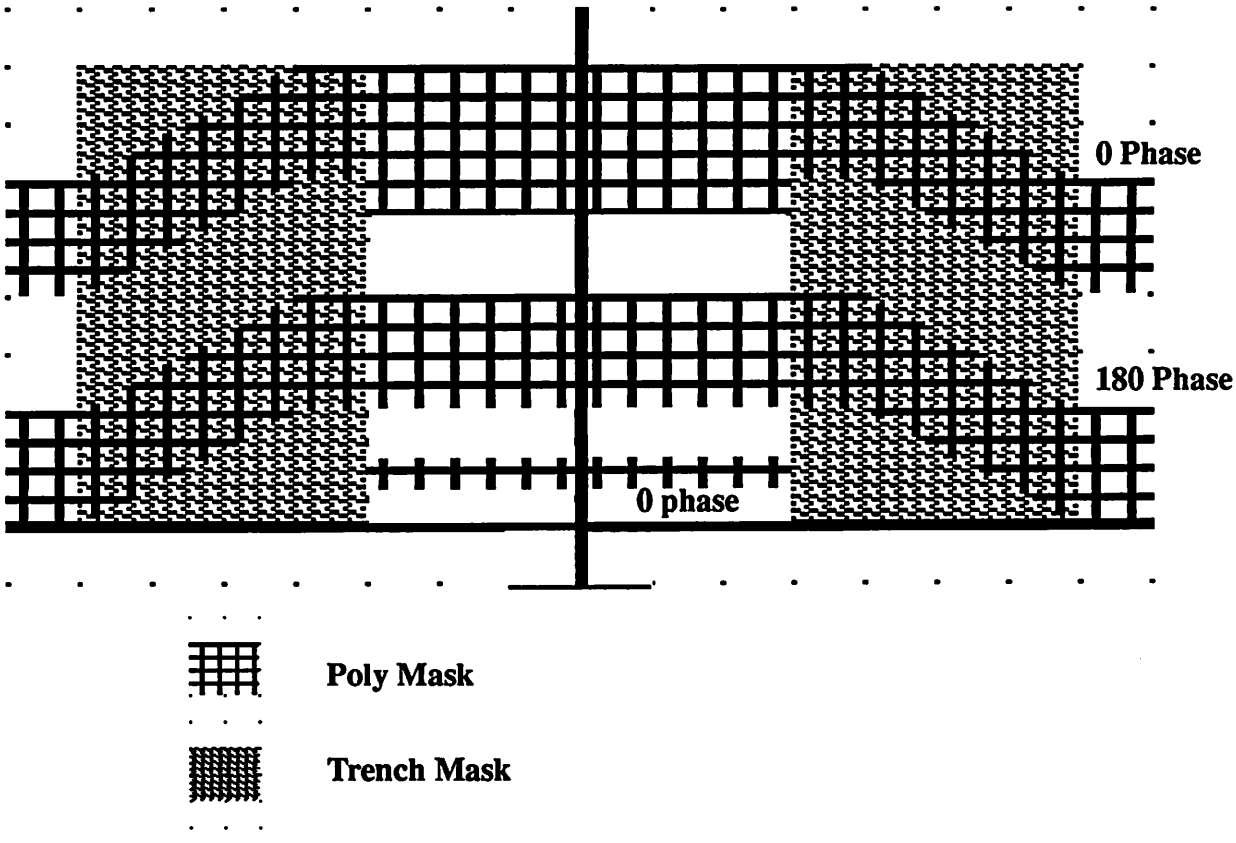


Figure 3.19 Layout of DRAM with top Poly line bloated by 0.2 λ/NA and phase band located 0.3 λ/NA from the edge of the lower Poly line. The non-printing band is 0 phase so it is the opposite phase of the bottom Poly line.

Chapter 4

Spatial Filtering

4.1 Implementation of Spatial Filtering in SPLAT

In order to implement spatial filtering at the lens pupil, it is necessary to delve into the fundamental equations used to implement SPLAT (Simulation of Projection Lens Aberrations via TCCs). The basic concept behind SPLAT arises from Hopkin's theory of partially coherent imaging. Hopkin's theory uses transmission cross-coefficients (TCCs) to calculate the image intensity. Physically, the TCCs model the interaction of the diffracted orders due to the partially coherent nature of the illumination source. According to Hopkin's theory, the four-fold integral of Equation 1 specifies the intensity of light in the image plane; the Fourier transform of the mask transmittance is given by F(f,g) where f and g are spatial frequencies. The equation sums the contributions from every pair of spatial frequencies (f',g'), (f",g") of the object. The resulting intensity, I(x,y), is the intensity in the image plane with coordinates (x,y).[5][14]

$$I(x,y) = \iiiint TCC(f',g',f'',g'')\,F(f',g')\,F^{\circ}(f'',g'')\,e^{-2\pi i[(f'-f'')x+(g'-g'')y]}\,df'dg'df''dg'' \qquad (EQ\ 1)$$

The key to evaluating this 4-fold integral is to evaluate the TCC. The TCC is defined in Equation 2 below. The $J_0(f,g)$ function is the mutual intensity of the light incident on the object while K(f,g) is the pupil function of the lens.

$$TCC(f',g',f'',g'') = \iint J_0(f,g)\,K(f+f',g+g')\,K^{\circ}(f+f'',g+g'')\,dfdg \qquad (EQ\ 2)$$

The TCC has a simple graphical interpretation; for every pair of spatial frequencies (f',g') and (f",g"), the TCC is the area of the overlapping circles shown in Fig. 4.1. $J_0(f,g)$ is the cone of illumination, and for kohler illumination, it is a circle whose radius is proportional to the partial coherence. Typically, the pupil function is a clear circular aperture. However, to model aberrations in the lens, a phase factor can be included in the transmission function. In addition, to model spa-

tial filtering at the pupil, a transmission function can also be added. Equation 3 is the new form for a pupil function which models both these effects. T(f,g) models the spatial filter while the phase factor in the exponent models the lens aberrations.

$$K(f,g) = T(f,g)\, e^{-i\frac{2\pi}{\lambda}\Phi(f,g)} \qquad (EQ\ 3)$$

When K(f,g) is introduced into Equation 2, we obtain an integral that is a function of the mutual intensity, the spatial filter, and the aberrations. If these unknowns are specified properly, the integral can be evaluated numerically. SPLAT uses a two dimensional adaptive quadrature method.[5]

$$TCC(f',g',f'',g'') = J_0 \iint T(f+f',g+g')\, e^{-i\frac{2\pi}{\lambda}\Phi(f+f',g+g')}\, T^{*}(f+f'',g+g'')\, e^{i\frac{2\pi}{\lambda}\Phi(f+f',g+g')}\, df\,dg \qquad (EQ\ 4)$$

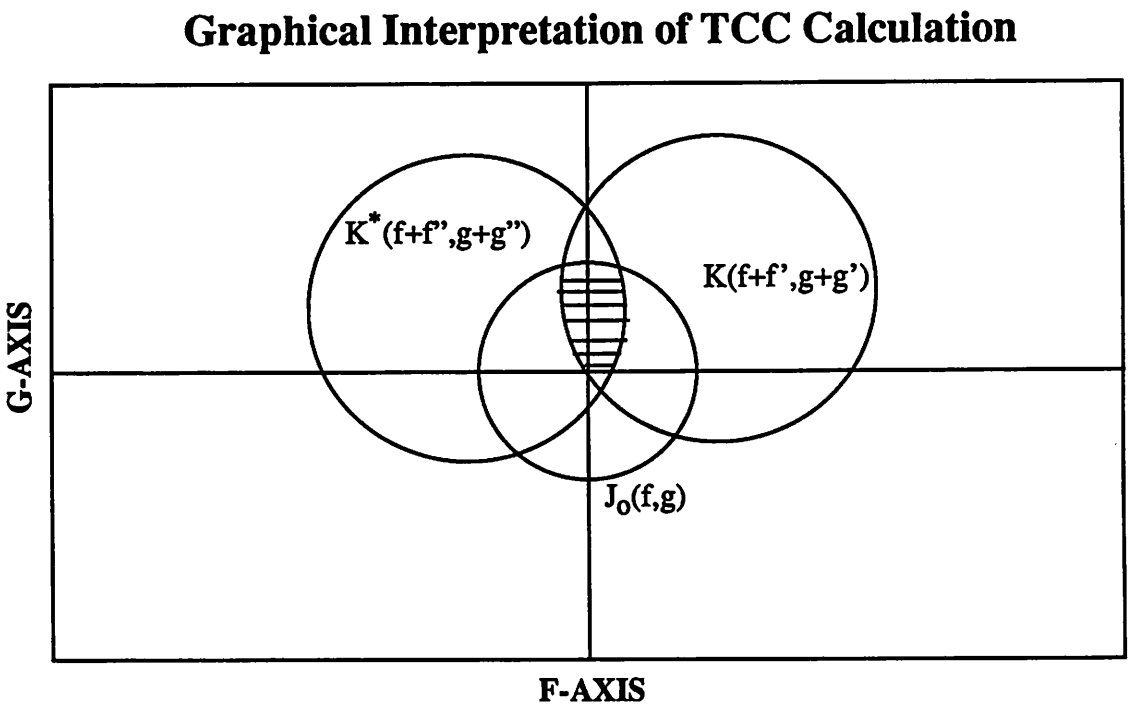


Figure 4.1 The contribution to the TCC integral for every pair of spatial frequencies can be determined by integrating over the area of the three overlapping circles shown here.[5]

4.2 The Hitachi Spatial Filter

Although a wide variety of spatial filters can easily be added to SPLAT, the filter suggested by Fukuda, Terasawa, and Okazaki at Hitachi Central Research Laboratory, has been examined because preliminary results show that it offers an exceptional increase in the depth of focus for both contacts and arbitrary features.[3] In addition, through the adjustment of two parameters, the filter offers a certain amount of flexibility. Through simulations the designer can extract values for these parameters in order to trade-off image characteristics to achieve acceptable stepper performance.

The Hitachi Spatial Filter function arises from the superposition of two images along the axis of light propagation. As discussed by Fukuda, the inclusion of Equation 5 in the standard

$$T(r) = \cos\left(2\pi\beta r^2 - \frac{\theta}{2}\right) \qquad \textit{(EQ 5)}$$

equation for the amplitude distribution is mathematically identical to the amplitude distribution resulting from the sum of two images separated in the z-direction by 2β. Thus, this filter function emulates two images; one image has a focus located at $+\beta$ and the other at $-\beta$. The image is remarkably constant throughout the space between the two focal planes, resulting in a large depth of focus.

Obviously, this function has two parameters: β and θ. One would expect that a relationship must exist between β and θ such that a particular value of β has one or more optimum values of θ. In addition, the relationship changes depending on the mask features. For contacts, the key concept is that the positive maximum of the pupil function must be at a radius of about 0.7. This setting maximally transmits the diffracted orders associated with the contact and rejects the other orders. Therefore, by setting the argument of the cosine equal to 0, we obtain Equation 6. Solving

$$2\pi\beta r^2 = \frac{\theta}{2} \qquad \textit{(EQ 6)}$$

for θ as a function of β we next obtain Equation 7 where r is chosen equal to 0.7 as discussed

$$\theta = 2\pi\beta \qquad \textit{(EQ 7)}$$

above. Plots of several filter functions in which this relationship holds are shown in Fig. 4.2.

Using Equation 7, it is possible to plot a variety of design graphs with MASC which demonstrate how the image characteristics change for different values of β. Fig. 4.3 indicates how the intensity of the central lobe and side lobe of a contact change with increasing β. For reference, the figure also shows how the clear field intensity changes with β. Fig. 4.4 is a plot of the inverse of the clear field intensity. This graph, thus, indicates the throughput penalty for utilizing a spatial filter. Of course, the throughput penalty can be eliminated simply by increasing the intensity of the illumination. Finally, Fig. 4.5 is a the plot of the depth of focus versus β. The depth of focus is determined by finding the focus level for which the intensity of the central lobe of the contact is reduced by 10%.

Ideally, one could use the design graphs in Fig. 4.3, 4.4, and 4.5 to choose β such that sufficient depth of focus is obtained while not allowing the side lobe intensity to become too large or the central lobe intensity to become too small. For example, if a depth of focus of 1.0 um is needed, Fig 4.5 can be used to find that β equals 0.52. Then, from Fig 4.3, the intensity of the central lobe is 0.2 while the intensity of the side lobe is 0.02. Also, the clear field intensity is 0.11. From Fig 4.4, the throughput hit is 7.5; thus, to achieve the same throughput after the addition of the spatial filter, the illumination source must be 7.5 times as intense. In general a 25% improvement in focus requires at least a factor of 5 reduction in the clear field intensity, and a 100% improvement requires a factor of 10 reduction.

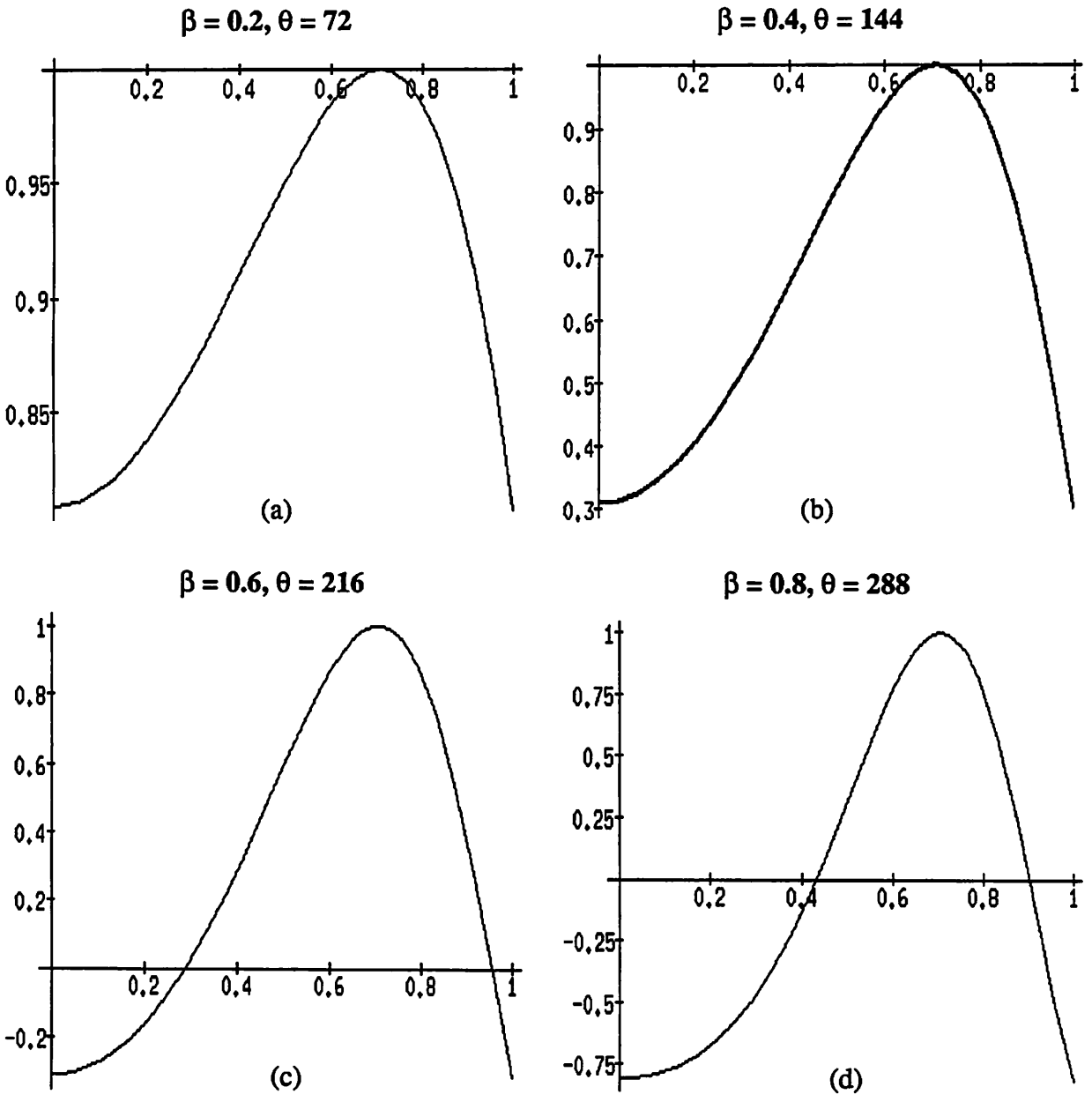


Figure 4.2 Plots of the Hitachi Spatial Function for a variety of β values which obey the relationship in Equation 7. θ is in degrees.

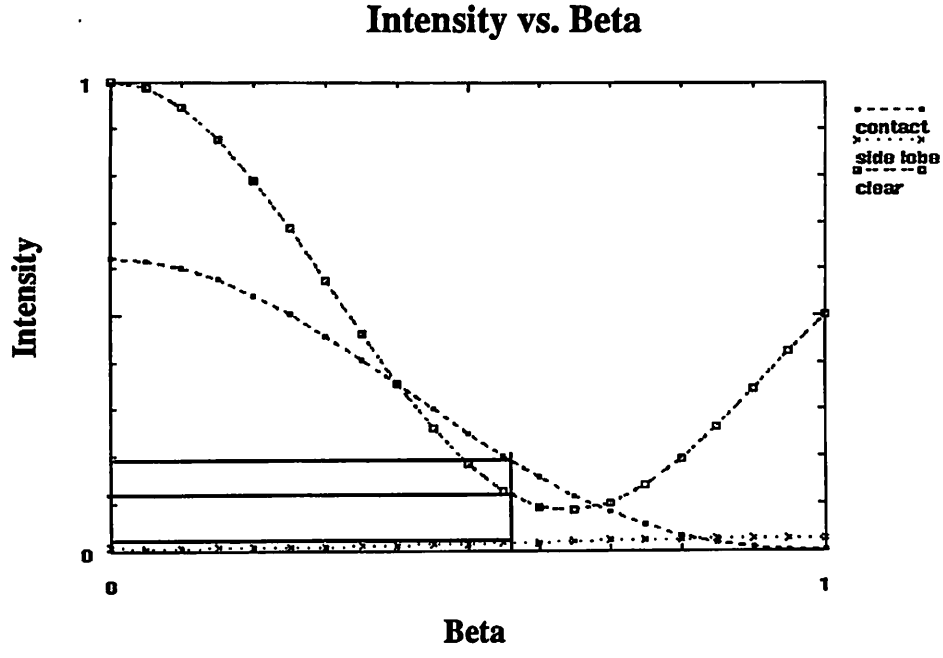


Figure 4.3 Design graph of the central contact lobe and side lobe intensities of a contact and the clear field intensity versus β.

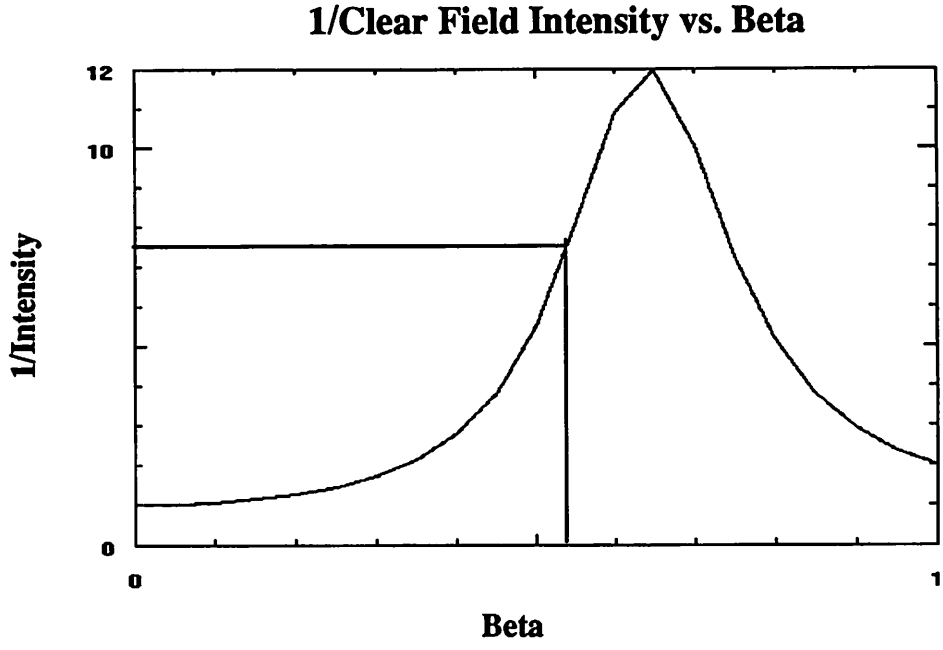


Figure 4.4 Design graph of the inverse of the clear field intensity versus β. The intent of the graph is to show the throughput hit resulting from the use of a spatial filter.

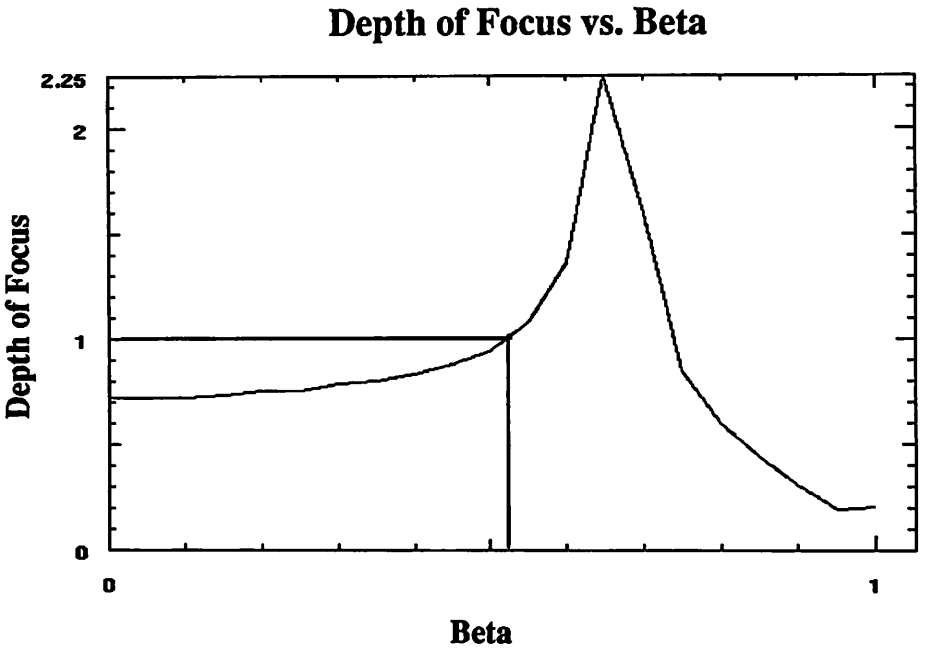


Figure 4.5 Depth of focus versus β. The depth of focus is determined by finding the focus level for which the intensity of the central lobe of the contact is reduced by 10%.

Conclusion

A computer aided design tool, Mask Analysis System by Computer (MASC), has been developed to automatically generate design graphs of user specified image quality measurements over one-dimensional cut lines and two-dimensional mask areas as a function of both optical system and mask layout variables. MASC is intended to serve as a tool to assist layout designers in determining design rules for phase-shift masks and in examining the printability of mask levels or combinations of mask levels. In addition, SPLAT has been extended to include lens filtering effects, and MASC is used to investigate the effects of spatial filters on mask features.

By developing a flexible tool in which single and multivariable iteration methods can be used with 1D or 2D data extraction and mask or projection system variables, a wide variety of analyses can be performed. We utilized MASC to investigate phase-shift masks and spatial filters. The single variable iteration method, 1D data extraction, and mask and projection system variables allowed us to examine a variety of features to determine design rules for phase-shifting. The multivariable analysis and 2D data extraction were useful for examining the misalignment problem and linewidth control of a DRAM. In addition, we examined how parameters of the Hitachi spatial filter affect contacts.

More specifically, in investigating technology issues, we demonstrate that isolated features and arrays of features can be printed on the same mask by correct application of shrink and bloat operations. Also, typical mask features can be scaled using phase-shifting without applying special design rules to each particular pattern. By running the system with a variety of phase transitions, we also determine that a transition length of 0.6 λ/NA is an optimum distance for 90 and 60-120 degree phase transitions. Bloated Levenson contacts can be used to produce closely spaced contact chains, but to achieve the best defocus tolerance for the contacts, we introduce the Merged Outrigger Alternating Contact String (MOACS) with non-printing alternating phase bands. The DRAM layout we studied is not as sensitive to alignment as might be expected by examination of the actual mask, and the linewidth control of the DRAM can be improved by bloating the top Poly line by 0.2 λ/NA and by adding a phase band 0.3 λ/NA from the lower edge

of bottom Poly line. The relationship between the two parameters of the Hitachi spatial filter was postulated to be $\theta = 2\pi\beta$. By using this relation for the filter, we found that to get more than a 25% improvement in the focus requires at least a factor of 5 reduction in the clear field intensity while a 100% improvement requires a factor of 10 reduction.

Although the current version of MASC offers many features, several improvements are needed. A closer tie to the PROSE Technology CAD environment is desirable to facilitate the generation of masks for MASC since PROSE can better address global mask issues. In addition, a graphical user interface connected to PROSE, which provides layout editing facilities, could make MASC much easier to use. Of course, more image quality measurements, which better describe how the image will print in resist, can be added as they are suggested by other researchers. Finally, SPLAT can be extended to include other non-conventional lithography techniques such as modifications of the cone of illumination.

Acknowledgments

First and foremost, I will thank Professor Andrew R. Neureuther for his valuable insights into the work discussed here. Second, I would like to thank Prof. Oldham for being my second reader and critical reviewer of my work. Also, thanks to John Nistlor of AMD for many interesting discussions about phase-shifting.

This work was supported by Sematech under grant 90-MC-500.

References

[1] M.D. Levenson, N.S. Viswanathan, R.A. Simpson, "Improving Resolution in Photolithography with a Phase-Shifting Mask," *IEEE Transactions on Electron Devices,* Vol. Ed-29, No. 12, pp. 1812-1846, December 1982.

[2] J.W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill Publishing Company, New York, pp. 141-149, 1968.

[3] H. Fukuda, T. Terasawa, S. Okazaki, "Spatial Filtering for Depth of Focus Resolution Enhancement in Optical Lithography," *EIPB,* 1991.

[4] S. N. Shiraishi, S. Hirukawa, Y. Takeuchi, N. Magome, "New imaging technique for 64-M DRAM," *SPIE: Optical/Laser Microlithography V,* March 1992.

[5] K.K.H. Toh, "Two Dimensional Images with Effects of Lens Aberrations in Optical Lithography," *M.S. Thesis, University of California, Berkeley,* May 1988.

[6] K.K.H. Toh, "Design Methodology for Dark-Field Phase-Shifted Masks," *Proceedings of SPIE: Optical/Laser Microlithography IV,* pp. 402-413, March 1991.

[7] Technology Modeling Associates, Inc., Palo Alto, CA.

[8] J.G. Garofalo, R.L. Kostelak, "Phase-Shifting Structures for Isolated Features," *Proceedings of SPIE: Optical/Laser Microlithography IV,* pp. 151-166, March 1991.

[9] C.A. Spence, D.C. Cole, B.A. Peck, "Using multiple focal planes to enhance the depth of focus," *SPIE: Optical/Laser Microlithography V,* March 1992.

[10] D.M. Newmark, A.R. Neureuther, "Phase-Shifting Mask Design Tool," *BACUS Photomask Technology,* 1991.

[11] D.M. Newmark, A.R. Neureuther, "Computer aided design tool for phase-shift mask design," *SPIE: Optical/Laser Microlithography V,* March 1992.

[12] A.S. Wong, D.M. Newmark, J.B. Rolfson, R.J. Whiting, "Investigating Phase-Shifting Mask Layout Issues Using a CAD Toolkit," *Intl. Electron Devices Meeting (IEDM)*, pp. 705-708, Dec. 1991.

[13] K.K.H. Toh, "Algorithms for Three-Dimensional Simulation of Photoresist Development," *Ph.D. Thesis, University of California, Berkeley,* Dec. 1990.

[14] M. Born and E. Wolf, *Principles of Optics, Sixth Edition,* Pergamon Press, London, 1980. Chapters 5, 9, and 10.

Appendix A

MASC User's Guide

SYNOPSIS

masc [command file] [geometry file]

DESCRIPTION

MASC (Mask Analysis System by Computer) is a C program which has been developed to automatically generate design graphs of user specified image quality measurements over one-dimensional cut lines and two-dimensional mask areas as a function of both optical system and mask layout variables. It is intended to assist layout designers in determining design rules for phase-shift masks and in examining the printability of mask levels or combinations of mask levels.

MASC executes an analysis based on the commands in the input files and writes the user requested results to disk The input files are expected to be in the current directory and the output are written there as well. The analysis performed by MASC can use a single or multivariable iteration scheme, 1D or 2D data extraction, and modification of a variety of mask and projection system variables. Basically, MASC reads the user input files, runs SPLAT, executes the 1D and 2D data extraction algorithms, and modifies the mask or projection system. The iterations continue for the number of times requested by the user. After the iterations, the program writes the extracted results to disk. The results from the single variable analysis are written to files that can be plotted with **drawplot**. The multivariable results are in files which are plotted by **contour**. The user is encouraged to modify or combine the output files to generate more meaningful graphs.

To execute the above options, the user must specify commands for three basic units: the tool, the mask geometry, and the design tasks. The command file contains the commands used for the tool and the design tasks while the geometry file contains commands for describing the mask geometry. The tool unit commands set up the projection printer variables and indicate how to modify them after each iteration. The mask geometry unit uses the commands, **area**, **B**, and **T** to establish the geometry. These commands are placed in the geometry file. However, the **shrink_factor** and **cif_scale** commands are placed in the command file. The design task unit incorporates the commands necessary to specify a single or multivariable analysis, 1D or 2D data extraction, and plotting. The commands in the following list are organized according to these units, but the user should be aware that the input is free format so the commands can be listed in any order.

COMMANDS

Command lines begin with a keyword, continue with a list of variables, and end with a semicolon ";". Occasionally, separator words appear in a command line to separate lists of variables. These words must be typed in as shown or the program will not run as intended. The commands are in bold type, the variables are in italics, and the separator words are in normal type.

TOOL UNIT

Projection Printer Commands

lambda *wavelength increment*;

lambda sets the *wavelength*, in micrometers, of the light used to illuminate the mask. The second variable, *increment*, sets the amount that the wavelength is increased after each iteration.

na *numerical_aperture increment*;

na sets the *numerical_aperture* of an imaging lens used in the projection system. The second variable, *increment*, sets the amount that the numerical aperture is increased after each iteration.

sigma *partial_coherence increment*;

sigma sets the *partial_coherence* of the imaging system. The second variable, *increment*, sets the amount that the partial coherence is increased after each iteration.

defocus *defocus increment*;

The image can be calculated at a plane other than the plane of best focus. This distance is entered by setting **defocus** to *defocus*. The defocus increase after each iteration is specified by *increment*.

xpyp *xpos ypos*;

This statement specifies the coordinates of the object relative to the lens axis. The coordinates (xpos,ypos) determine how the object is situated (perpendicular, parallel, etc.) relative to the lens axis. For example, (xpos,ypos) = (1,0) means that the mask is situated parallel to the lens axis. This distinction is important for the proper simulation of aberrations.

coma *coma increment*;

coma specifies the amount of *coma* in the lens system and *increment* specifies the amount to increase the coma after each iteration.

astigmatism *astigmatism increment*;

astigmatism specifies the amount of *astigmatism* in the lens system and *increment* specifies the amount to increase the astigmatism after each iteration.

spherical *spherical_aberration increment*;

spherical specifies the amount of *spherical_aberration*] in the lens system and *increment* specifies the amount to increase the spherical aberration after each iteration.

curvature *curvature increment*;

curvature specifies the amount of *curvature* in the lens system and *increment* specifies the amount to increase the curvature after each iteration.

distortion *distortion increment*;

distortion specifies the amount of *distortion* in the lens system and *increment* specifies the amount to increase the distortion after each iteration.

hitachi_spf b*eta beta_inc theta theta_inc*;

hitachi_spf specifies a hitachi filter of the form:

$$\cos\left(2\pi\beta r^2 - \frac{\theta}{2}\right)$$

beta (β) is the distance between the planes of best focus and *theta* (θ) is the phase difference between those planes.

MASK GEOMETRY UNIT

Geometry File Commands

area *length height* bottom_left *bot_x bot_y*;

area allows the specification of the *length* and *height* of the mask which will be simulated. If the mask is not normalized to bottom left coordinates of (0,0), then the user can specify *bot_x, bot_y* to properly normalize the mask.

B *length height center_x center_y* phase *phase increment* move *x_move y_move* b_s *amount side1 side2 side3 side4*;

B specifies a CIF box along with some additional variables that specifies how the boxes are modified after each iteration. The separator word "phase" allows the specification of the *phase* of each box and the *increment* in the phase. The next separator word, "move", specifies the box movement in the x and y direction through the variables *x_move* and *y_move*. Finally, "b_s" performs a bloat or shrink operation on each side of a given box. *Side1, side2, side3, side4* indicate the side or sides that will be bloated or shrunk by *amount* after each iteration. *Amount* can be any integer while *side1, side2, side3, side4* is specified by putting a number other than 0 for the value. For example, if *amount* = -1 and *side1*= 1 while *side2* = *side3* = *side4* = 0, then after each iteration the left hand side of the box is move left by 1 CIF unit. Note that the sides are specified as shown in Fig. A.1.

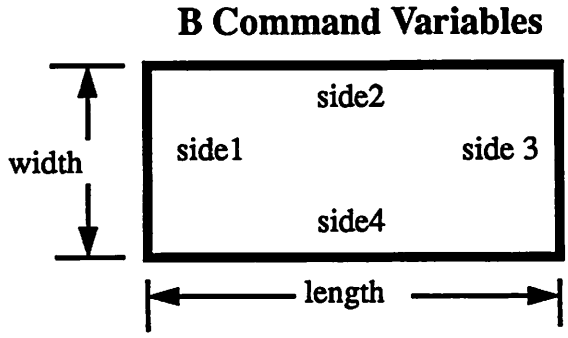


Figure A.1. Variables for the **B** command in the Geometry File.

T *x1 y1 x2 y2 x3 y3* phase *phase;*

T specifies a SPLAT triangle. This command allows the specification of triangles in MASC, but this version does not allow any modification of their parameters.

Command File Mask Geometry Commands

shrink_factor *shrink*;

shrink_factor specifies the *shrink* that will be applied to the mask after each iteration. In other words, all parameters of the mask are multiplied by *shrink* after each iteration. By default, *shrink* is set to 1.0.

cif_scale *scale_factor*;

cif_scale specifies the *scale_factor* which will be used to translate the coordinates in the geometry file into values for SPLAT boxes. Usually, the coordinates of the geometry file are in CIF units so this command scales the CIF units into microns for SPLAT.

DESIGN TASK UNIT

Iteration Schemes

iterations *number_of_runs*;

iterations specifies the number of times to execute the main program loop. Data from each iteration is saved as specified by the **plot** command. After each iteration of the main program loop, the variables above are increased by their increment. Note that the **B** command in the geometry file specifies how the mask is modified after each iteration. Also, default operation modifies all variables. It is the users responsibility to carefully enter each *increment* for the variables.

multi_variable *var1 param1 var2 param2*;

multi_variable allows the user to vary two parameters during a single MASC run. The results are plotted in the contour format. *Var1* is the first variable while *var2* is the other variable. These variables can be any command which is modified by an increment after each iteration. Table 2 lists all keywords which can be used for *var1* and *var2*. For exam-

ple, one could vary the numerical aperture and partial coherence while extracting the contrast of the image. The output plot is a contour plot in which the x-axis shows the changing numerical aperture and the y-axis shows the changing partial coherence. The z-values of the contour file would be the contrast. Table 2 shows the variables which may be used in the **multi_variable** command.

Table 1: Keywords Available for the Multivariable Analysis

sigma	na	lambda	defocus	coma
spherical	astigmatism	curvature	distortion	beta
theta	B			

Currently, the only keyword which requires *param1* or *param2* is B. For this keyword, the purpose of the parameter is to identify which box is being modified during the desired analysis. The user simply enters the box number, counting from the last box, of the box being modified. For all other words, *param1* and *param2* should be set to 0.

1D Data Extraction

cutline *xi yi xf yf* move *x_move y_move* intensity *i1 i1_inc i2 i2_inc i3 i3_inc i4 i4_inc i5 i5_inc*;

cutline specifies how the cut lines will be drawn through the image. This command has several options which allow flexibility in specifying how the cut lines are used. All the variables specified in this command are in CIF units. *xi yi xf yf* are the parameters which determine the coordinates of the cut line through the image. If the separator word "move" follows the cut line coordinates, *x_move y_move* specify how the cut line should be moved after each iteration. If the keyword "intensity" follows the move variables, the user can specify five locations where the intensity is recorded after each iteration: *i1, i2, i3, i4, i5*. These locations can be modified after each iteration by using non-zero entries for *i1_inc, i2_inc, i3_inc, i4_inc, i5_inc*.

min_space *minimum_spacewidth*;

min_space specifies the *minimum_spacewidth*. This command is used to properly calculate the contrast between features. See new_contrast in the **plot** command description.

2D Data Extraction

contour_analysis *bottom_left.x bottom_left.y x_length y_length*;

contour_analysis tells MASC that an analysis of the intensity slope along a contour is desired. Effectively, this command directs MASC to tell SPLAT to write a contour data file which is then analyzed to determine the slope at every point on the contour image. Then, the slope along the 0.30 intensity threshold of the image is extracted. The results from this operation can be plotted using the **plot** command. The results which may be plotted are: 1) the slope along the slice for each iteration, 2) the maximum slope along a given slice, 3) the minimum slope along the slice, and 4) the average slope along the slice.

intersection *reg_bot_left.x reg_bot_left.y x_length y_length*
lwa_bot_left.x lwa_bot_left.y x_length [y_length cif_file

intersection tells MASC to perform an analysis which extracts the intersection resulting from the overlap of two mask layers. The first four numbers represent the region analysis box which is simply the section of the mask for which SPLAT will write the contour data file. The next four numbers indicate the linewidth analysis box which allows one to restrict the intersection to a single line (2 curves). Finally, *cif_file* tells MASC the name of the second mask. The result of this analysis will typically be two curves, which allows the program to calculate a minimum spacewidth for the line. As with the **contour_analysis** command, this command allows one to plot: 1) the slope along the intersection curves, 2) the maximum slope along a given intersection curve, 3) the minimum slope along the intersection curve, 4) the average slope along the curve, and 5) the minimum spacewidth between the curves. Note that 1) plots n curves, where n is the number of iterations, while the other plots are single curves which have n points.

Depth of Focus Calculation

depth_of_focus *var param change accuracy*;

depth_of_focus causes MASC to perform a calculation of the depth of focus after each iteration. For var, the user may enter any keyword from the list of keywords available for cut line plotting. *Param* again is used to indicate the particular instance of the variable on the cut line. *Change* is used to tell MASC how much *var* must change for the image to be considered out of focus. For example, a 0.1 change means that when the variable changes from its original value by 10%, the image is considered out of focus. *Accuracy* defines how close *change* must be to the actual change of the variable. The depth of focus calculation always utilizes the last **cutline** entered in the command file.

Plotting

plot *plot_name parameter* x_begin *x_begin* increment *increment*;

The **plot** command allows the user to specify the plots that should be written to the current directory after all iterations are completed. *Plot_name* is one of the allowed keywords listed in Table 1. MASC records multiple values for spacewidth, linewidth, intensity slope, and log slope, depending upon the number of times these variables appear on a cut line. Thus, the user must specify which value to record. *Parameter* is used to specify this information. To determine the proper value for *parameter*, examine a cut line and count the number of times the given variable would be extracted if the curve is examined from left to right. Use this number for *parameter.* (NOTE: count from 0) After the separator word "x_begin", enter the starting value on the x_axis for the plot. Again, after the separator word "increment", specify the x_axis increment for the plot. Usually, you will want *increment* to reflect the increase in one of the projection system variables or the mask modification values. The *increment* value is ignored if the *shrink* is not 1.0.

Table 2: Plotting Keywords

CUT LINE PLOTTING KEYWORDS AND OUTPUT FILES

keyword	file
slope10	slope10.[j].c[k]
slope20	slope20.[j].c[k]
slope30	slope30.[j].c[k]
slope40	slope40.[j].c[k]
slope50	slope50.[j].c[k]
slope60	slope60.[j].c[k]
logslope10	logslope10.[j].c[k]
logslope20	logslope20.[j].c[k]
logslope30	logslope30.[j].c[k]
logslope40	logslope40.[j].c[k]
logslope50	logslope50.[j].c[k]
logslope60	logslope60.[j].c[k]
spacewidth10	sp10.[j].c[k]
spacewidth20	sp20.[j].c[k]
spacewidth30	sp30.[j].c[k]
spacewidth40	sp40.[j].c[k]
spacewidth50	sp50.[j].c[k]
spacewidth60	sp60.[j].c[k]
linewidth10	lw10.[j].c[k]
linewidth20	lw20.[j].c[k]
linewidth30	lw30.[j].c[k]
linewidth40	lw40.[j].c[k]
linewidth50	lw50.[j].c[k]
linewidth60	lw60.[j].c[k]
local_max	lmax.[j].c[k]
new_contrast	new_cntrst.[j].c[k]
contrast	contrast.c[k]
maxI	maxint.c[k]
minI	minint.c[k]

CONTOUR PLOTTING KEYWORDS AND OUTPUT FILES

keyword	file
depth_of_focus	dof..c0
max_contour_slope	max_cntr_slp.cu[j]
min_contour_slope	min_cntr_slp.cu[j]
average_contour_slope	ave_cntr_slp.cu[j]
slope_along_slice	slp_cntr.cu[j]
minimum_contour_space	min_cntr_sp.cu[j]

The slope, logslope, spacewidth, and linewidth are recorded at the intensity threshold of 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6. The contrast, maxI, and minI are specified for the entire cut line. New_contrast is saved at each user specified intensity location, but the result makes sense only when the user specified intensity is a maximum (or near a maximum) within a given feature. Thus, the slope, logslope, spacewidth, and linewidth require the second variable *parameter*. The local_max and new_contrast keywords also require *parameter* to specify which of the 5 intensity locations to plot. To keep this plethora of information under control, these number are indicated in the plot file name as [j]. The information is automatically recorded for each cut line that was entered and [k] indicates the different cut lines. The contour plotting keywords request plots such as the slope along a contour, the minimum, maximum, and average slope along that contour, and the minimum spacewidth between two curves. These are explained further in the **intersection** and **contour_analysis** commands. The contour plotting keywords are slightly different because the [j] in the filename now indicates the curve number as extracted from the contour program using the -slice option. Please refer to the examples for further explanations of this organizational method.

weight_function *var param weight*;

weight_function is used in conjunction with **multi_variable** to determine the variable that will be plotted in the z-direction. For *var*, the user can choose any of the cut line plot keywords. As before, *param* indicates the particular instance of the variable on the cut line. Finally, *weight* is the multiplying factor for the variable. For example, if an objective function involving two variable is desired, each with equal weight, two **weight_function** commands are entered. The *weight* would be 0.5 for both. Typically, the weights should add up to 1, but that is not required.

AUTHOR

David M. Newmark

SEE ALSO

SPLAT, drawplot, contour

Examples

Example 1

The goal of this example is to show a simple shrink analysis. In the command file, the shrink factor is set to 0.95 which means that all the coordinates are multiplied by 0.95 after each iteration. Since the program is run for 11 iterations, the total shrinkage is 60%. The minimum space on the original mask is 0.8 and this is indicated by min_space. The cut line and geometry information is in CIF units and they are scaled by 50 to obtain the nominal units for SPLAT. Two cut lines are drawn through the mask as shown in Fig. A.2. The variables plotted from these cut lines are the slope30 at two locations, the maximum intensity, and the spacewidth30. The geometry file gives the dimensions of the mask and the information about how to move the boxes on the mask after each iteration. The raw graphs from this run of MASC are shown in Fig. A.3. The standard output for this example is attached. Since this output is similar for the remaining examples, it is not included.

Command File

```
lambda 0.5;
na 0.5;
sigma 0.5;
defocus 0.0;
shrink_factor 0.95;
iterations 11;
min_space 0.8;
cif_scale 50;
cutline 120 0 120 240 move 0 0 intensity 60 0 0 0 0 0 0 0 0 0;
cutline 0 0 0 240 move 0 0 intensity 60 0 0 0 0 0 0 0 0 0;
plot slope30 0 x_begin 0.8;
plot slope30 2 x_begin 0.8;
plot maxI 0 x_begin 0.8;
plot spacewidth30 1 x_begin 0.8;
plot local_max 0 x_begin 0.8;
```

Geometry File

```
area 240 240 bottom_left 0 0;
B 240 40 120 60 phase 180 0 move 0 0 b_s 0 0 0 0 0;
B 80 40 120 180 phase 0 0 move 0 0 b_s 0 0 0 0 0;
B 240 40 120 140 phase 0 0 move 0 0 b_s 0 0 0 0 0;
```

Phase-Shift Mask Used in Example 1

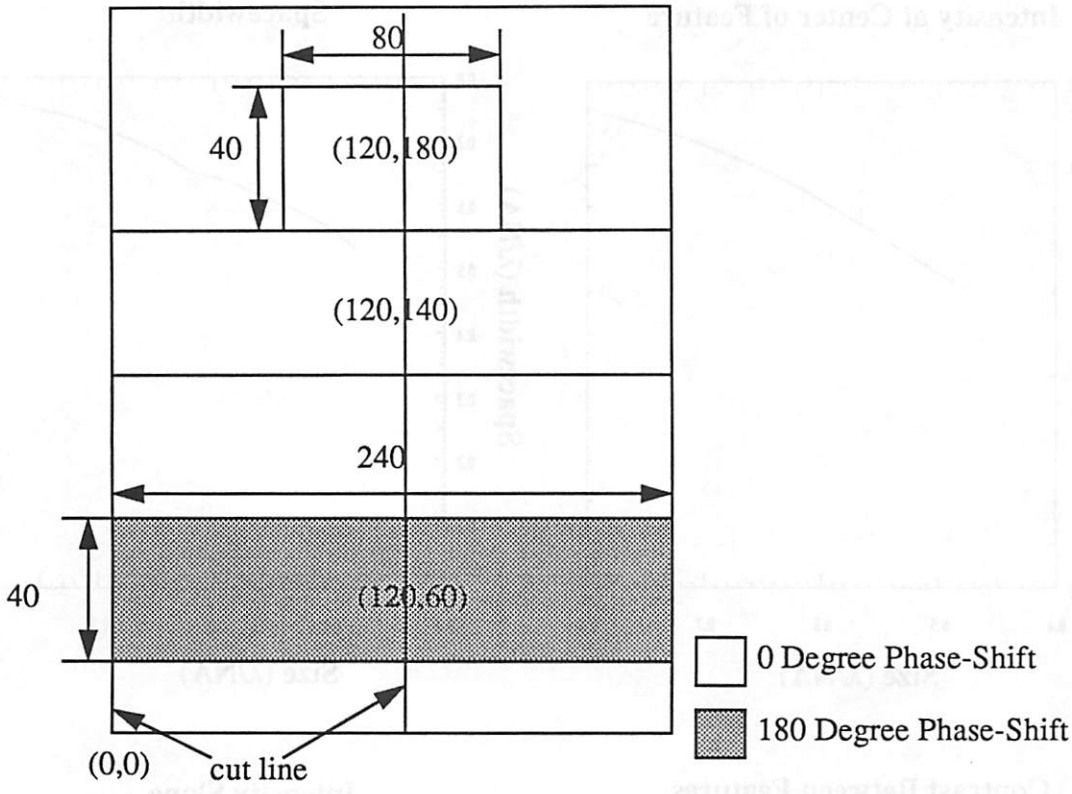


Figure A.2. The actual mask along with the dimensions of each box are shown here. The dimensions shown are the center coordinates and the length and width. In addition, the cut lines are drawn through the mask as indicated in the command file. Note that the mask is mirrored along the x and y axes.

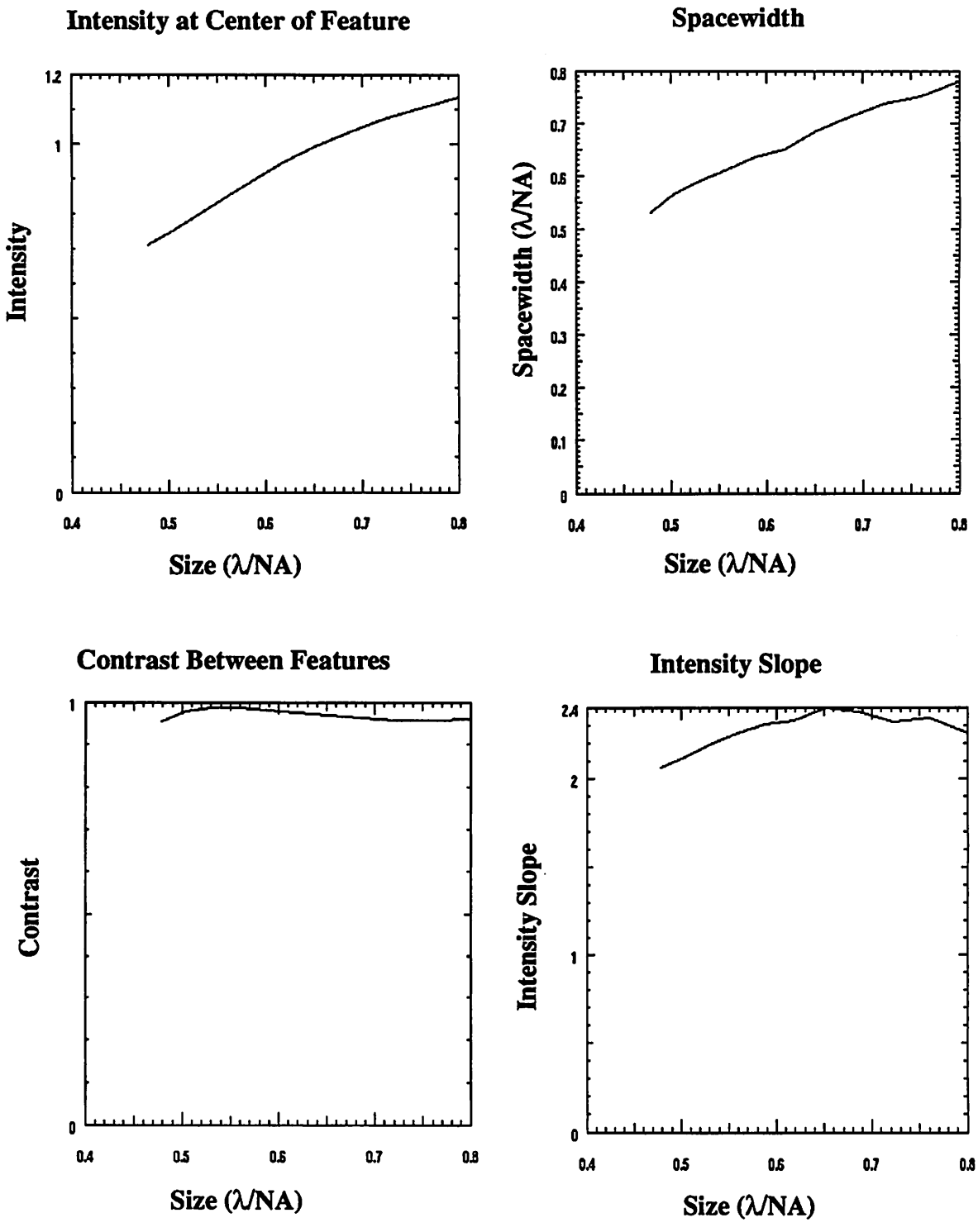


Figure A.3. Raw design graphs generated by MASC. These graphs can be combined in more interesting ways as described in Chapter 3.

Example 1 : Standard Output

```
Options : Mask area
 Bottom Left   = (0.00,0.00)
 Width         = 240.00
 Height        = 240.00

Options : Box Added
(All values in CIF units)
 x length              = 240.00
 y length              = 40.00
 center coordinates    = (120.00,60.00)
 phase, phase increment  = 180.00, 0.00
 move box in x-direction  = 0.00
 move box in y-direction  = 0.00
 bloat or shrink amount  = 0.00
 side1 = 0 side2 = 0 side3 = 0 side4 = 0

Options : Box Added
(All values in CIF units)
 x length              = 80.00
 y length              = 40.00
 center coordinates    = (120.00,180.00)
 phase, phase increment  = 0.00, 0.00
 move box in x-direction  = 0.00
 move box in y-direction  = 0.00
 bloat or shrink amount  = 0.00
 side1 = 0 side2 = 0 side3 = 0 side4 = 0

Options : Box Added
(All values in CIF units)
 x length              = 240.00
 y length              = 40.00
 center coordinates    = (120.00,140.00)
 phase, phase increment  = 0.00, 0.00
 move box in x-direction  = 0.00
 move box in y-direction  = 0.00
 bloat or shrink amount  = 0.00
 side1 = 0 side2 = 0 side3 = 0 side4 = 0

Options : Wavelength
 lambda        = 0.500000
 lambda_inc    = 0.000000

Options : Numerical aperature
```

```
na            = 0.500000
na_increment  = 0.000000

Options : Partial coherence
 sigma        = 0.500000
 sigma_increment = 0.000000

Options : Defocus
 defocus      = 0.000000
 defocus_inc  = 0.000000

Options : Shrink Factor  = 0.95000

Options : Number of Runs  = 11

Options : Minimum spacewidth on mask = 0.80

Options : CIF scale factor = 50.00

Options : Cutline Added
 Coordinates (120.00,0.00) to (120.00,240.00)
 Move cutline after each iteration
  x-direction = 0.00, y-direction = 0.00
 Record intensity at locations 60.00 0.00 0.00 0.00 0.00
 Move each intensity location by 0.00 0.00 0.00 0.00 0.00

Options : Cutline Added
 Coordinates (0.00,0.00) to (0.00,240.00)
 Move cutline after each iteration
  x-direction = 0.00, y-direction = 0.00
 Record intensity at locations 60.00 0.00 0.00 0.00 0.00
 Move each intensity location by 0.00 0.00 0.00 0.00 0.00

Options : Plot Requested
 Type of plot : slope30
 Parameter            = 0
 X_axis beginning value  = 0.80
 Increment for x_axis    = 0.00

Options : Plot Requested
 Type of plot : slope30
 Parameter            = 2
 X_axis beginning value  = 0.80
 Increment for x_axis    = 0.00

Options : Plot Requested
```

Type of plot : maxI
Parameter = 0
X_axis beginning value = 0.80
Increment for x_axis = 0.00

Options : Plot Requested
Type of plot : spacewidth30
Parameter = 1
X_axis beginning value = 0.80
Increment for x_axis = 0.00

Options : Plot Requested
Type of plot : new_contrast
Parameter = 0
X_axis beginning value = 0.80
Increment for x_axis = 0.00

Options : Plot Requested
Type of plot : local_max
Parameter = 0
X_axis beginning value = 0.80
Increment for x_axis = 0.00

Example 2

This example demonstrates a movement analysis in which an end is moved closer to a space. The wavelength, numerical aperture, partial coherence, and defocus are set up as before. The shrink_-factor is set to 1.0 so no shrink analysis takes place. We still run 11 iterations. Fig. A.4 shows that the spacewidth is 40 units and the CIF scale factor is 57 so the minimum spacewidth is 0.7 um or 0.7 λ/NA.The specification for the movement which takes place after each iteration is located in the Geometry file. The last **B** command contains a value for *x_move* which informs MASC to move the lower box 2 CIF units in the y-direction after each iteration. The original data graphs are presented in Fig. A.5.

Command File

```
lambda 0.5;
na 0.5;
sigma 0.5;
defocus 0.0;
shrink_factor 1.0;
iterations 10;
min_space 0.5;
cif_scale 57;
cutline 120 0 120 240 move 0 0 intensity 220 0 0 0 0 0 0 0 0 0;
plot linewidth30 0 x_begin 0.5 increment -0.025;
plot slope30 0 x_begin 0.5 increment -0.025;
plot new_contrast 0 x_begin 0.5 increment -0.025;
```

Geometry File

```
area 240 280 bottom_left 0 0;
B 240 40 120 220 phase 0 0 move 0 0 b_s 0 0 0 0 0;
B 40 120 120 100 phase 180 0 move 0 2 b_s 0 0 0 0 0;
```

Phase-Shift mask used in Example 2

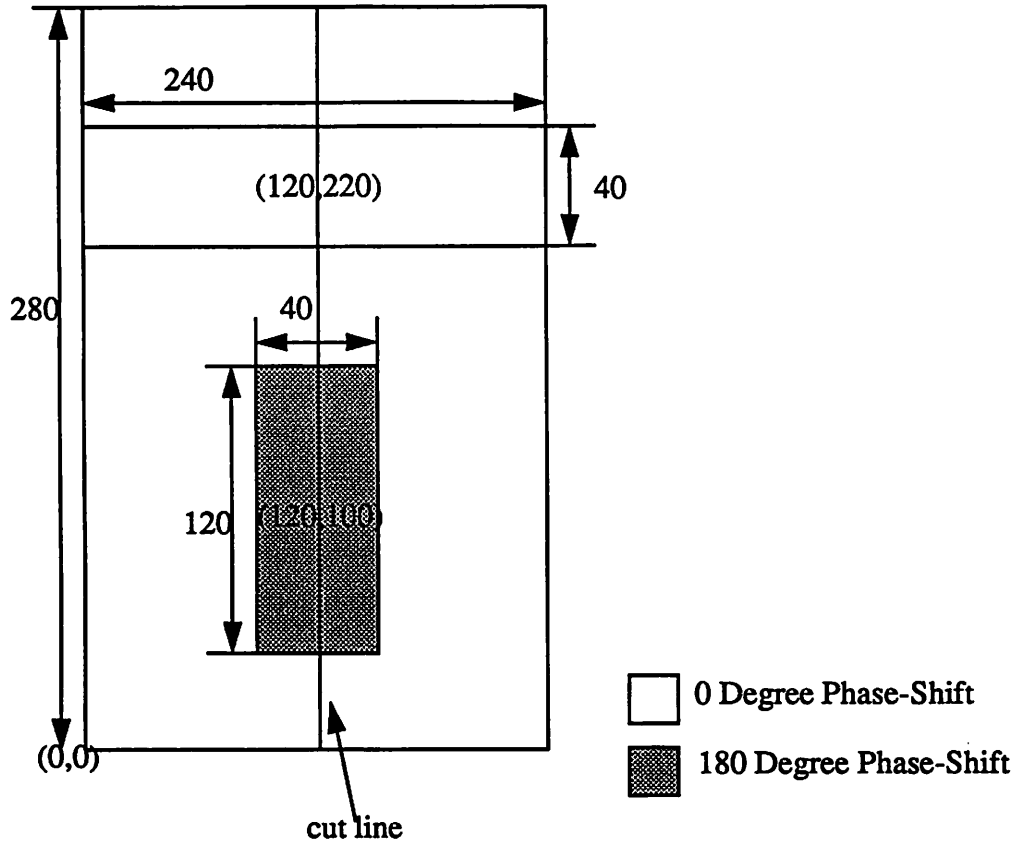


Figure A.4. The actual mask used in Example 3 along with the dimensions of the boxes and the cut line.

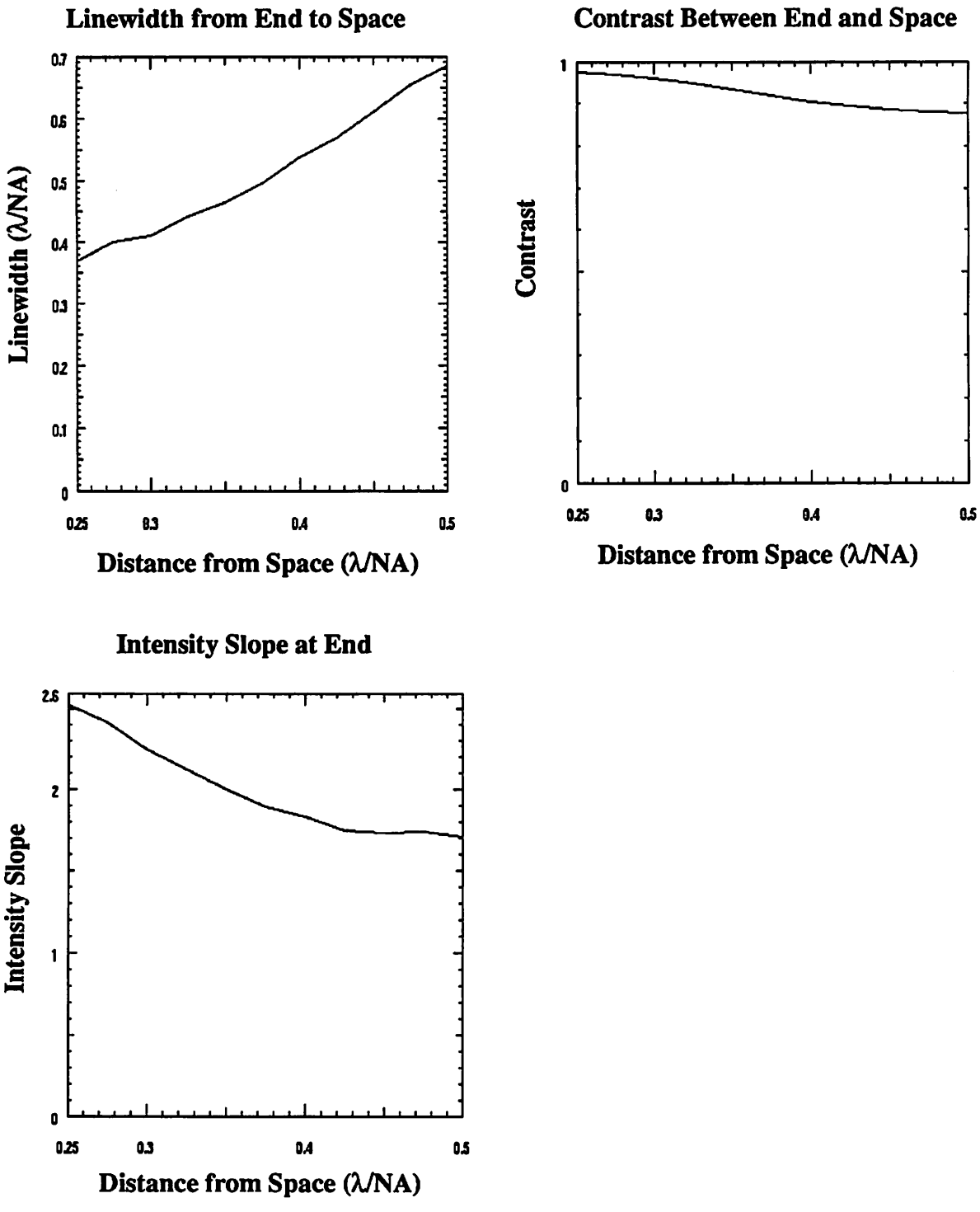


Figure A.5. Design graphs generated by Example 2. Again these graphs can be combined for comparison purposes and put together to yield more interesting result such as those in Chapter 3.

Example 3

Example 3 uses MASC to bloat a phase-shift mask. The mask bloated in this example is a simple array of lines and spaces. The original width of the lines is 0.4 λ/NA and they are bloated until the mask becomes chromeless. This is again accomplished in the **B** command with the use of the b_s option. This example utilizes the symmetry about the x and y axes to make a periodic array of 0.4 λ/NA lines and spaces.The spaces are bloated to create a chromeless array of lines and spaces. After the bloat, the spaces on the mask are 0.8 λ/NA wide. The geometry for this mask is shown in Fig. A.6. The command file is very similar to Example 1 and 2, but the geometry file utilizes the b_s option to bloat the spaces. The way this is accomplished is that the right edge of the phase-shifted box is moved to the right while the left edge of the non-phase-shifted box is moved left. This movement creates larger open areas on the mask by sacrificing the open area between the features. The output graphs are shown in Fig. A.7.

Command File

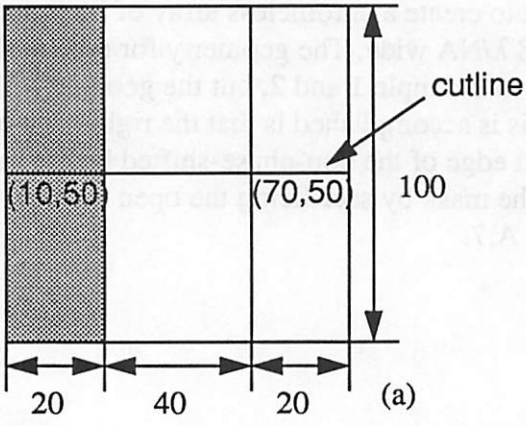
```
lambda 0.5;
na 0.5;
sigma 0.5;
defocus 0.0;
shrink_factor 1.0;
iterations 10;
min_space 0.4;
cif_scale 100;
depth_of_focus contrast 0 0.2 0.001;
cutline 0 50 80 50 move 0 0 intensity 0 0 0 0 0 0 0 0 0 0;
plot slope30 0 x_begin 0.4 increment 0.04;
plot maxI 0 x_begin 0.4 increment 0.04;
plot spacewidth30 1 x_begin 0.4 increment 0.04;
plot contrast 0 x_begin 0.4 increment 0.04;
plot depth_of_focus 0 x_begin 0.4 increment 0.04;
plot linewidth30 0 x_begin 0.4 increment 0.04;
```

Geometry File

```
area 80 100 bottom_left 0 0;
B 20 100 10 50 phase 180 0 move 0 0 b_s 2 0 0 1 0;
B 20 100 70 50 phase 0 0 move 0 0 b_s 2 1 0 0 0;
```

Phase-Shift Masks for Bloating Features

Initial Mask



Final Mask

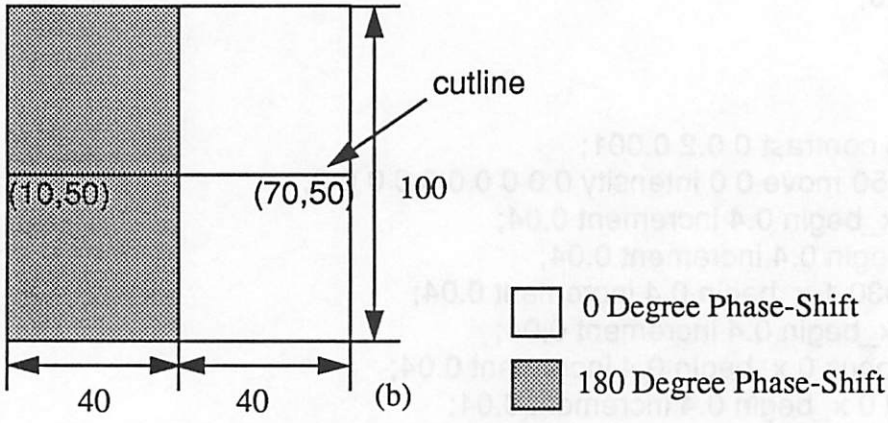


Figure A.6. The initial phase-shift mask for Example 3 is shown in (a). After running MASC, the mask would appear as shown in (b).

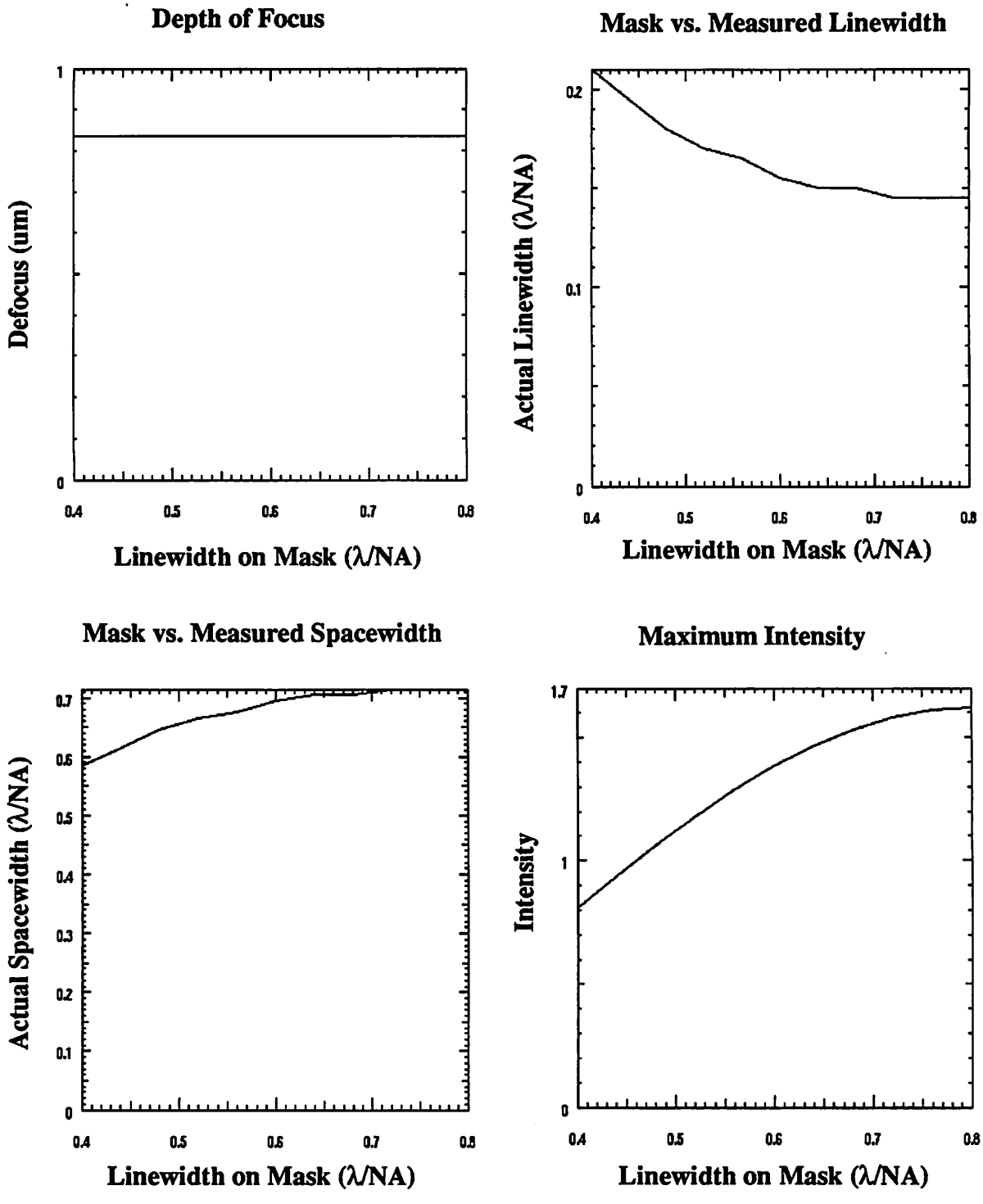


Figure A.7. Design graphs from Example 3. It is interesting to note that the depth of focus (measured by a contrast change) does not change during bloating of the spaces.

Example 4

To demonstrate the multivariable analysis method, this example shows how the depth of focus is affected by changes in the numerical aperture and wavelength. The mask used in this example is the same as the one utilized in Example 3 except the CIF scale factor has been changed to 50 so the lines and spaces are now 0.8 λ/NA. The depth of focus is defined as the point where the contrast changes by 10%. The main different between this example and Example 3 is the use of the multi_variable keyword. To utilize this keyword, we set an *increment* value for both lambda and na. Then, in the multi_variable keyword, lambda and na are specified as the variables to change. MASC will change lambda from 0.3 to 0.6 while keeping na at 0.3. It then moves na to 0.32, resets lambda and repeats. SPLAT will be run 121 times to obtain the result, not including the SPLAT runs needed to find the depth of focus. The contour plot resulting from this analysis is shown in Fig. A.8.

Command File

```
lambda 0.3 0.03;
na 0.3 0.02;
sigma 0.5;
defocus 0.0;
shrink_factor 1.0;
iterations 11;
min_space 0.6;
cif_scale 67;
depth_of_focus contrast 0 0.2 0.02;
cutline 0 50 80 50 move 0 0 intensity 0 0 0 0 0 0 0 0 0 0;
multi_variable lambda 0 na 0;
weight_function depth_of_focus 0 1;
```

Geometry File

```
area 80 100 bottom_left 0 0;
B 20 100 10 50 phase 180 0 move 0 0 b_s 0 0 0 0 0;
B 20 100 70 50 phase 0 0 move 0 0 b_s 0 0 0 0 0;
```

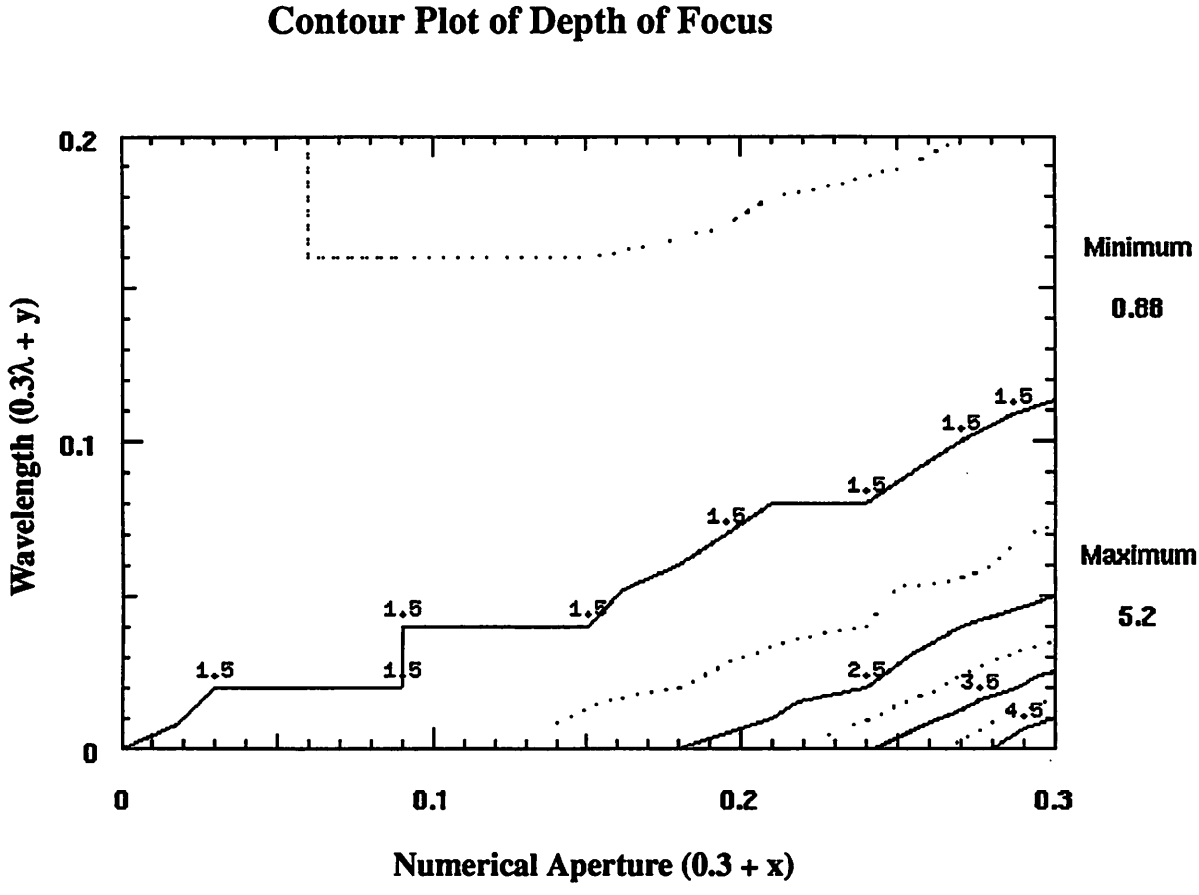


Figure A.8. Contour plot of the results from the multivariable analysis of Example 4. The initial values of wavelength and na are 0.3. As expected the largest depth of focus occurs for the smallest value of na and largest value of wavelength.

Appendix B

Addendum to SPLAT User's Guide

Additional Input Statement for SPLAT

[Statement] 29 filter # parameter 1 parameter 2

This statement informs SPLAT that a spatial filter should be placed in front of the lens pupil. Only one filter has been implemented in this version of SPLAT so the filter # must be set to 1.0. The two parameters for the Hitachi filter are β and θ. The Hitachi filter works by placing two distinct images along the light axis. β is the distance these image reside from the original focal plane. θ is the phase difference between the images.

SPLAT EXAMPLE

Example 1 : Input File

```
# Spatial Filter Example using SPLAT
#
# 0.6 λ/NA contact opening in opaque mask.
#
Statement 1 : Printlevel 3;
Statement 2 : lambda = 0.5 um;
Statement 3 : NA = 0.5;
Statement 4 : Defocus = 0.0 um;
Statement 5 : Sigma = 0.5 um;
Statement 6 : mask = 2.0 um x 2.0 um at 0 transmittance;
Statement 7 : cutout = (0.0, 0.0) 0.3 x 0.3 at 1;
Statement 29 : filter = 1.0, beta = 0.65, theta = 4.4;
Statement 10;
Statement 14 : intensity (-2.0,0) .. (2,0) 'tmp.xsect0';
Statement 0 : end;
```

Example 1 : Line Printer Output

```
2D optical imaging with aberrations — V3.0
                                7/2/91 — KT

Trial  1:  Print level=  3
Trial  2:  Lambda      =  0.5000 microns
Trial  3:  N.A.        =  0.5000
Trial  4:  Defocus     =  1.5000 microns   [ 1.5000 Rayleigh Units ]
Trial  5:  Sigma       =  0.5000
Trial  6:  Field size =  2.0000 x  2.0000 @ 0.0000  scale =1.00
#      6:  x size requires  10 harmonics.
#      6:  y size requires  10 harmonics.
Trial  7:  Cutout      =( 0.0000, 0.0000)x( 0.3000, 0.3000) @ 1.0000 <  0.000
Trial 29:  Pupil Transmittance of cos(2*pi*    0.65 *r**2 -    4.40/2)
Trial 10:  Calculate image Fourier Transform
#      10:  Symmetry :  T( f1, g1, f2, g2) =
#      10:               T(-f1,-g1,-f2,-g2)
#      10:               T( f1,-g1, f2,-g2)
#      10:               T(-f1, g1,-f2, g2)
#      10:         conjg[T( f2, g2, f1, g1)]
#      10:  TCC computation time  =  30.76000 sec.
#      10:  Imaged with  10 by  10 harmonics
#      10:  TCC calls:    1257  zeros:     664
Trial 14:  3-decimal plot line : ( -2.000,  0.000)..(  2.000,  0.000)
          401 points saved in "tmp.xsect0"
Program execution terminated.
User Time (CPU) =   34.45000 sec,  System Time =   0.77000 sec.
```

Example 1 : Drawplot output of 'tmp.xsect0'