STORM VERSION 1.0

by

Ebo Croffie and Marco Zuniga

Memorandum No. UCB/ERL M98/49

30 April 1998

STORM VERSION 1.0

by

Ebo Croffie and Marco Zuniga

Memorandum No. UCB/ERL M98/49

30 April 1998

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

STORM version 1.0

Ebo Croffie, Marco Zuniga

Electronics Research Laboratory

University of California, Berkeley

STORM version 1.0

Ebo Croffie, Marco Zuniga

Electronics Research Laboratory

University of California, Berkeley

1.0 Introduction

1.1 Overview

STORM (Simulation of TSI and Other Resist Mechanisms) is a program for simulating Top Surface Imaging (TSI) and other resist mechanisms in lithography processes. The principal components of STORM are SPLAT for exposure simulation, BAKE for Postexposure bake (PEB) simulation, and SILY2D for silylation simulation. Later versions will include an oxygen reactive ion etching image transfer simulator. The computer program for SILY2D and BAKE solves a set of chemical kinetics nonlinear partial differential equations using the finite element method for space discretization and a backward difference formula time advancement scheme for time discretization. In addition to the chemical kinetics equations, SILY2D also solves the virtual power equation to determine the mechanical stress fields and polymer deformation. The simulation domain may be a cross section or a top view of a photoresist. This release includes the Contour, X-Window™ plotting program to visualize the simulation results.

The algorithm for BAKE is capable of simulating problems such as postexposure bake, line end shortening and resist film top loss. The algorithm for SILY2D is capable of simulating problems such as stress induced reaction retardation effects on polymer swelling during silylation.

This version of STORM comes without GUI; the user must execute the program by modifying the appropriate input parameter files. The included software tools may be run as separate stand

alone programs. Communication between programs is primarily through files. This form of information transfer allows intermediate steps to be saved and retrieved for later use.

STORM is designed for use on engineering workstations running under the Unix™ operating systems. The plotting program require the X window system. Memory requirements depends on the number of nodes approximating the simulation domain. Practical simulations require between 1 and 200 megabytes of physical memory. Run times depends on machine and problem size. For example, BAKE and SILY2D run under 1 minute on 600MHz DEC Alpha and under 10 minutes on a 200MHz Sun Ultra Sparc machine (441 nodes).

1.2 Source Code Information and Acknowledgments

This release of STORM comes without the source code. However, the following information on the source code is given for ease of document modification for future releases. BAKE and SILY2D code of STORM ver1.0 is written in the C++ programming language. The code uses the libraries below. We acknowledge the following people for making these libraries available in the public domain:

Triangle	Jonathan R. Shewchuk
SuperLU	Sherry Li
MV++	R. Pozo
Meschach	David E. Stewart

The class description files are:

bake.h	Postexposure bake simulator class definition
sily.h	Silylation simulator class definition
node.h	Domain grid point class definition
ele.h	Domain element class definition
myblas.h	Auxiliary BLAS header file

The corresponding class implementation files are:

bake.cc	
sily.cc	
node.cc	
ele.cc	
myblas.cc	
main.cc	Class user file

STORM use the following files:

bake.in	PEB initial condition file

bake.par	PEB parameters file
sily.par	Silylation parameters file
viewsplat.m	MATLAB script for viewing SPLAT results
viewbake.m	MATLAB script for viewing BAKE results
viewsily.m	MATLAB script for viewing SILY2D results
sim_region	PERL script for choosing appropriate region

to simulate

1.3 About this manual

Section 1.0	**Introduction** A general overview of STORM is provided.
Section 2.0	**Installation** This section discusses how the simulation program STORM is compiled and installed.
Section 3.0	**Running STORM and individual components** A brief introduction to commands and input file keywords is presented in this section.
Section 4.0	**Lithography Simulation Tutorial** Two tutorial examples to let the user become familiar with the commands and procedures of running STORM are given
.Section 5.0	**Physical Models in STORM** This section describes the physical models implemented in the various programs contained SPLAT, BAKE, and SILY2D.

The conventions used throughout this users' guide are as follows:

- Words which are in ALL CAPITAL LETTERS represent programs or shell scripts.
- Words in **bold** letters represent commands to be typed exactly as it appears on the users' guide.
- Words in *italics* represent input or output filenames for command line arguments
- Words inside [square brackets] are options to commands.
- Words in the `Courier font` represent characters in a file or output from programs.
- The string "ws%" represents the unix prompt on a workstation.
- The string ">>" represents the MATLAB prompt.

2.0 Installation

2.1 Resource Requirements

The computing environment of the user should:

- be capable of compiling and running C and FORTRAN programs.
- have at least 10 mega bytes of disk space available to hold the data files
- have MATLAB ver 4.0 or higher software. (The user could write his/her own script files for another program or even write their own analysis programs.)

2.2 Organization

The distribution files are separated into a STORM archive and two scripts:

1. StormMake (Script to build STORM.)
2. StormClean (Script to clean all object files)
3. STORM Code (Directories containing the code and binaries to build STORM)

2.3 Installation

This software package consists of the main program STORM as well as other supporting routines to be run on the user's workstation.

1. SILY2D (Program for silylation simulation.)
2. BAKE (Programs for Postexposure bake simulation.)
3. SPLAT (Program written in Fortran for calculating aerial images.)
4. CONTOUR (Program to view SPLAT output)

To install STORM, move storm1.0.tar.gz to a directory (say $HOME/bin) where STORM should reside. Uncompress the STORM by typing the following command:

ws% gunzip storm.tar.gz

If done correctly, the .gz suffix that was previously attached to the file should vanish.

Untar the storm1.0.tar file by typing:

ws% tar -xvf storm1.0.tar

ws% cd STORM

You should now see the two shell scripts and following directories:

```
Bake
```

```
Contour
Sily2D
Splat
```

You are now ready to start compiling SPLAT and CONTOUR. Make sure you are in the STORM AND type:

ws% **csh** *StormMake*

You will now be asked to enter the system under which your workstation is running. Please enter this information accurately and properly since the source code will not compile adequately without this information.

Since SPLAT is written in FORTRAN, you will also be asked to enter the FORTRAN compiler you have. The two types of compilers for which Makefiles are provided are the XLF (for the IBM XLF compiler) and f77 compilers. If your compiler is different from either of these two, you will need to construct your own Makefile for SPLAT and compile it individually. When compiling begins you should see the following messages about the status of the compilation on the screen:

```
STORM Release 1.0
 ============================
 (C) 1998 University of California, Berkeley

 compilation and installation

 please send comments/bug reports/suggestions/
...
 to ebo@argon.eecs.berkeley.edu

Please enter target (SunOS, solaris or AIX or
DEC) = SunOS
Please enter the Fortran compiler you have (f77
or xlf) f77

Compiling for OS = SunOS


and so on
```

Check to make sure that the exact configuration, such as the system and machine, matches your current setup.

If the installation is successful, you should see the following message:

```
Compilation completed successfully
Copy or link needed executables to your bin
Have a nice day!!!
```

If compilation is unsuccessful, you should see an error message telling you when and where compilation was terminated. In this case, please see Section 2.4 which gives you suggestions that may help the program compile properly.

After compilation, you may or may not want to remove the object files that were created for each program. To remove ALL the object files that were created, simply type:

ws% **csh** *StormClean* at the Unix prompt.

Once again you should see some messages telling you that object files are being removed.

2.4 Installation Problems

The main reasons for an unsuccessful installation are:

1. Lack of disk space

   Either provide sufficient space or adjust the "build.all" script to build only one module at a time and then delete the installed tar files after the build.

2. Incorrect permission settings

   Make sure you have sufficient permission for the destination directories. Check the status display to see that the destination directories are as desired. If you are installing to a system directory (such as /usr/local/bin), you may have to "su root."

   ##The installation script automatically sets the permission level to 0755 for executables and 0644 for libraries. If you need to set different permission levels, you will have to modify each Makefile manually. There are no provisions for customized permission levels at this point. #

3. Missing header or library files

   This occurs normally because of missing or incorrect X libraries. Check to make sure that your X release is complete and current. If the installation still fails, then contact us.

4. Compilation errors

   Make sure standard compilers (like cc and not gcc, ANSI C, or CC) are used.

3.0 Running STORM and individual components

STORM consists of many powerful tools which can be combined together to simulate a process. In order to run SPLAT, for example, the *splat-inputfile* must be specified. Likewise, to run BAKE and SILY2D, the *bake-inputfile* and *sily-inputfile* must be given (see below for details on the file formats). MATLAB plots can be generated from the output files.A full simulation using the three modules is performed by running the following:

ws% **splat** < *splat-inputfile*

SPLAT outputs a 2D contour file of the exposed resist which can then be baked to generate a hydroxyl group concentration matrix needed for silylation simulation. BAKE can then be run by typing:

ws% **bake**

BAKE generates a hydroxyl group concentration matrix which can be silylated by SILY2D by typing:

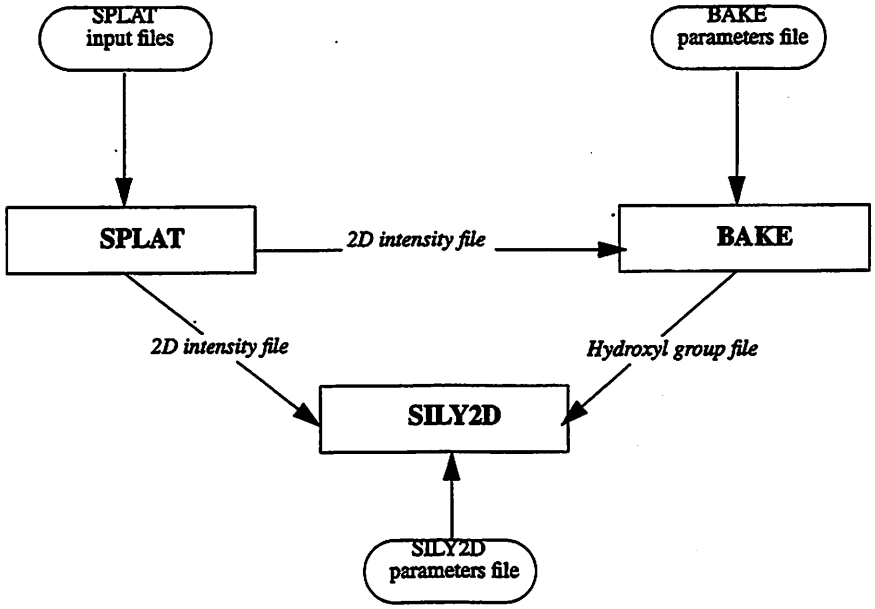
% **sily -bake**

Silylation can be simulated without prior Postexposure bake. In this case SILY2D is run by typing:

% **sily -splat**

The following STORM process flow diagram will serve to explain the dependency of the various modules in a concise manner: In the diagram below, note that SILY2D can be run without running BAKE first.

PROCESS FLOW DIAGRAM FOR STORM



3.1 Running SPLAT

For detailed information on SPLAT, consult the SPLAT User's Manual. The basic information is presented here as an introduction to running SPLAT as a stand alone program.

SPLAT accepts a file of statement numbers, with each statement number followed by a list of arguments. A command lines beginning with the pound sign (#) is ignored, and a line beginning with an ampersand (&) is considered as a continuation of the previous line.

The projection system is specified by defining the illumination wavelength, numerical aperture, defocus, partial coherence and the five primary lens aberrations: spherical aberration, coma, astigmatism, curvature, and distortion.

The mask is specified as a set of rectangles or triangles, each with a given transmittance and phase. Then the image is simulated for a specified region.

The parser does not recognize key words, and simply skips over them. Thus, the following is a basic valid SPLAT command file:

Example thin film SPLAT input file (*splat.tsi*)

```
#file generation TSI resist
stmt 1: print = 3
stmt 2: lambda = 0.248 um
stmt 3: na = 0.6
stmt 4: defocus = 0.0 mode = 1
stmt 5: sigma = 0.5
stmt 6: mask = 0.3 x 0.3 trans = 0;
stmt 7: cutout = (0.00, 0.00) 0.125 x 0.3 trans = 1;
stmt 35: resist layer = RESIST_LAYER 'film.uvII'
stmt 31: mag = 5 mode = 0;
stmt 32: 50; #table look-up
stmt 10: 0 2;
stmt 14: cutline from (-0.3, 0.0) to (0.3, 0.0)
& with 31 points mode = 1 in 'uvII.RESIST_LAYER'
end 0
```

The last line (stmt 14) tells SPLAT to discretize each layer (0.6um wide) using 31 nodes.

In addition to *splat.tsi* input file, parameters specific to the photoresist being exposed must be specified.

Example UVIIHS photoresist input file (*film.uvII*):

```
4.73 -0.14 1.0 0.0
1.68 20.74 0.2 0.012 5 0.8 90
```

The last 2 numbers on the last line (0.8 90) tells SPLAT to divide a 0.8um thick resist into 90 layers. For more information on other parameters, please

at the explanation given under SPLAT in the Physical Models section of this manual.

To run the above, type:

ws%run.layers *splat.tsi* 1 90

run.layers is a unix script to run many thin-film interference simulations using a single input file. The script searches for the word 'RESIST_LAYER' and replaces it with the current resist layer and then calls SPLAT. Therefore, in this particular simulation there will be 90 different 1D intensity contour files in the current directory. To convert these 1D intensity plots to a 2D contour plot, type:

ws%merge.layers uvII.RESIST_LAYER 1 90 >*tsi.cnt*

merge.layers is another unix script used to merge 2D intensity contour files created using run.layers command. In this example the merged contour file is written to *tsi.cnt*. The output

file needs to be edited to reflect the spatial dimensions of the photoresist. **See Tutorial section for example.**

The output file *tsi.cnt* contains the intensity profile information of the resist matrix. Each number in the file (excluding the header lines) is the normalized intensity value at a particular node in the resist. Note that in this particular example, the number of nodes representing the resist matrix is 31X90=2790 nodes. Most uniprocessor machines do not have enough memory resources (> 1 Gigabyte) to run STORM with this many nodes. In most applications, the intensity pattern in the resist is periodic (standing waves in non absorbing resists) or the exposed light is absorbed in the near top region of the resist (TSI resists). In both cases only a small region of the domain can be simulated without much loss of accuracy. In our particular example the top 0.1 micron of the resist can be simulated. Also in most cases, symmetry can be exploited to simulate only half of the selected domain. To select the top 0.1 micron of the resist for BAKE and SILY2D simulation, run the included PERL script file as follows:

ws%sim_region *uvII.cnt* 0.1 0.3

This will output the file *bake.in* in the Bake directory

3.2 Running BAKE

BAKE expects the image to be stored in a file called *bake.in*. Any other image file must be moved to this name for BAKE to execute properly. The PEB parameters must be specified in the parameter file *bake.par*.

Example BAKE parameters input file (*bake.par*):

```
1.10e-3
55.0
5.56
1.0
1.7
0.0
100.0

# Do: Diffusivity Coefficient
# K_1: Reaction Rate e Coefficient
# K_2: Acid Loss Rate
# omega: Nonlinear Diffusion Coupling Coefficient
# fit: Acid fitting parameter
# k_4: Acid Sites Desorption rate constant
# t_final: Total Postexposure Bake Time
```

To run BAKE, type:

ws% bake

The program quits executions under the following 2 conditions:

1) Program time reaches specified PEB time

2). Program reaches steady state before specified PEB time.

A successful BAKE run should output *hydro.in*, an input file for SILY2D.

3.3 Running SILY2D.

SILY2D expects the input files called *bake.in* and *hydro.in*. If you run BAKE before running SILY2D, bake generates the *hydro.in* file and outputs it to the Sily2D directory. You must make sure you copy the bake.in file into the Sily2D directory before running SILY2D. If you are running SILY2 without a previous BAKE run, any the image file from SPLAT must be moved to the Sily2D directory for SILY2D to execute properly.

Before running sily2D, the silylation parameter values must be specified in the file *sily.par.*

Example SILY2D parameters input file (*sily.par*)

```
1.0e-3
1.0
0.01
1.0
1.0
100.0
500.0
10.0
5.5e-05
0.001
0.634
0.01
100.0

# D: Diffusivity Coefficient
# K_1: Reaction RatRate e Coefficient
# psi: Nonlinear Stress Coupling Coefficient
# omega: Nonlinear Diffusion Coupling Coefficient
# t_relax: Polymer Relaxation Time
# eta: Polymer Viscosity
# alpha: Young Modulus?
```

```
# P: Chamber Pressure
# k_3: Silylation Agent Adsorption rate constant
# k_4: Silylation Agent Desorption rate constant
# E_max: Polymer Swelling Parameter (get rid of
later)
# A_max: Polymer Swelling Parameter (get rid of
later)
# t_final: Total Silylation Time
```

To run SILY2D WITHOUT prior BAKE run, type:

ws% **sily -splat**

To run SILY2D WITH prior BAKE run, type:

ws% **sily -bake**

3.4 Running The Plot Programs.

The results obtained from simulation can be plotted using MATLAB™ or CONTOUR. The plot programs, CONTOUR may be run as independent X-Window programs. They are run by typing the program name followed by the plot data file name. MATLAB scripts are provided for plotting SPLAT, BAKE and SILY2D output data. If you get an error message like "*cannot open display*", you need to set your Unix shell DISPLAY variable by typing:

ws% **setenv DISPLAY** *host:display*

where *host* is the name of your workstation and *display* is the number assigned to your terminal screen.

To exit CONTOUR, simply click the mouse in the window or in the QUIT button if one exists. The plotter will then create a PostScript file named *dataplot.ps* and prompt to ask if the file should be sent to the default printer.

4.0 A Tutorial for doing Lithography Simulation

The following section offers a small tutorial for 2D Lithography Simulation. The first tutorial simulates acid diffusion in a DUV resist cross section during PEB. A top view simulation is also presented for completeness. The second tutorial simulates silylation of a TSI resist. This brief tutorial is meant to help the user get a feel for running simulations in STORM. One can also use this tutorial to check that the software has been installed and is running properly.

4.1 Exposure of DUV Resist Cross Section

The SPLAT input deck initializes various process conditions such as the wavelength, diameter of the numerical aperture, partial coherence factor and the size and geometry of the mask. The following command is used to simulate a thin-film version of SPLAT.

```
stmt 35: resist layer = RESIST_LAYER 'film.uvII'
```

"film.uvII" contains the thin-film SPLAT parameter values to be used in this simulation..

The entire SPLAT input file, *splat.uvII,* is shown below. For more information on these commands, please look at the explanation given under SPLAT in the Physical Models section of this manual.

```
# Fast thin-film SPLAT example for uvIIHS resist.
# 0.25 um contact
stmt 1: print = 3
stmt 2: lambda = 0.248 um
stmt 3: na = 0.6
stmt 4: defocus = 0.0 mode = 1
stmt 5: sigma = 0.5
stmt 6: mask = 0.3 x 0.3 trans = 0;
stmt 7: cutout = (0.00, 0.00) 0.125 x 0.3 trans = 1;
stmt 35: resist layer = RESIST_LAYER 'film.uvII'
stmt 31: mag = 5 mode = 0;
stmt 32: 50; #table look-up
stmt 10: 0 2;
stmt 14: cutline from (-0.3, 0.0) to (0.3, 0.0)
& with 31 points mode = 1 in 'uvII.RESIST_LAYER'
end 0
```

The parameter file, *film.uvII,* is also shown below.

```
The thin-film SPLAT input file
```

```
4.73 -0.14 1.0 0.0
1.68 0.74 0.2 0.012 5 0.8 90
```

To run a thin-film SPLAT input deck, type

ws% run.layers *splat.uvII* 1 90

This simulation will create 90 different 1D intensity contour files in the current directory.

Now, in order to create a 2D intensity file that is accepted by BAKE, type

ws% merge.layers uvII.RESIST_LAYER 1 90 >*uvII.cnt*

This will merge the 1D intensity contour files created using the **run.layers** command to create a 2D intensity contour file *uvII.cnt.*

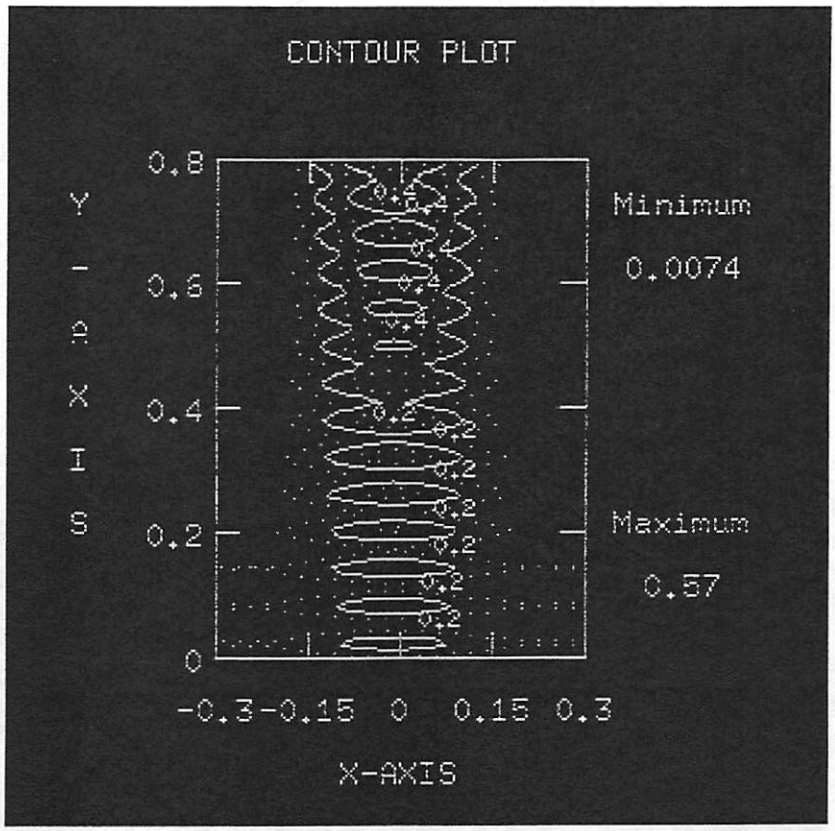
The output file needs to be edited to reflect the spatial dimensions of the photoresist. In our example, the x-axis domain is (-0.3, 0.3) and the y-axis domain is (0.0, 0.8) um. The number of x-points is 31 (from stmt 14). To do this, edit *uvII.cnt* with your favorite editor (e.g. vi, emacs, etc.) and make the following changes:

```
xmin --> -0.3
xmax --> 0.3
ymin --> 0.0
ymax --> 0.8
nxpts --> 31
```

The 2D intensity contours can now be viewed by typing:

ws%contour -cstep 0.1 *uvII.cnt*

A window should appear with a contour plot looking like the following:



The nodes approximating the resist domain in the merged contour file are too many for practical Postexposure bake simulation. As can be seen in the contour plot, the contours of the intensity profile consist of several standing waves.We can study the effects of Postexposure bake on this profile by simulating the top 0.2um (~4 standing waves) of the 0.8um thick resist. Since the feature is symmetric about the x-axis, we simulate only half of the domain, i.e. (xmin, xmax)=(0.0, 0.3um). Furthermore, the exposed region is only a fraction of the domain. All interesting activities will take place in the exposed region of the domain. Thus, we can trim our simulation region even further by simulating a width of 0.25um. To recap, our simulation region is now (xmin, xmax) = (0, 0.25) instead of (-0.3, 0.3) and (ymin, ymax)= (0.6, 0.8) instead of (0.0, 0.8).. To choose such a domain, type the following:

ws%sim_region ***uvII.cnt*** *0.2 0.25*

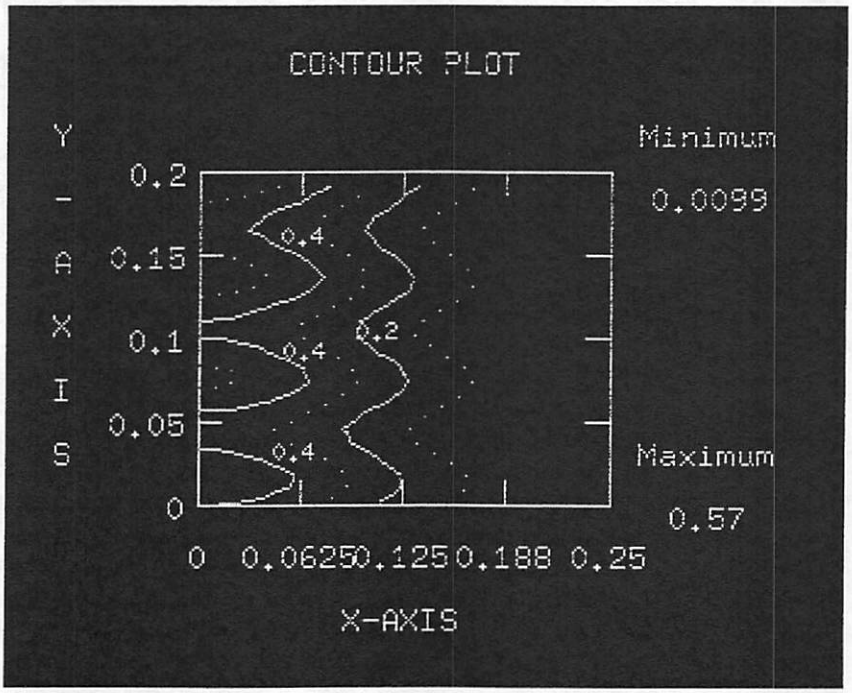
sim_region is a script that take as an argument the resist thickness from top and width from center (0.2um thick and 0.25um wide in our example). It outputs the file ***bake.in*** (in Bake directory) containing the intensity profile of the specified region for BAKE and/or SILY2D simulation. You should see the following when you execute the above command:

```
%sim_region uvII.cnt .2 .25
Original thickness = .8
Original width = .3
Original matrix = (90 x 31)
New thickness = .2
New width = .25
New matrix = (22 x 13)
Output file is '../Bake/bake.in'
```

Make sure you are in the Bake directory and type

ws%contour -cstep 0.1 *bake.in*

to view the new simulation domain. It should look like this:



4.2 Postexposure Bake of DUV Resist Cross Section

We are now ready to run BAKE. Make sure *bake.in* exists in your Bake directory and choose the following PEB parameters in *bake.par*:

```
1.10e-3
5.50
0.556
1.0
1.7
0.0
100.0

# Do: Diffusivity Coefficient
# K_1: Reaction RatRate e Coefficient
# K_2: Acid Loss Rate
# omega: Nonlinear Diffusion Coupling Coefficient
# fit: Acid fitting parameter
# k_4: Acid Sites Desorption rate constant
# t_final: Total Bake Time
```

Type:

ws% bake

to run the PEB simulation. You should see the following output:

```
%bake
************* S T O R M ************


======Reading Post Exposure Bake Parameters========

Diffusivity Coefficient = 0.0011
Reaction Rate Coefficient = 5.5
Acid Loss Rate Coefficient = 0.556
Nonlinear Diffusion Coupling Coefficient = 1
Acid Fitting Parameter (m) = 1.7
Acid Sites Desorption rate constant = 0
Total Post Exposure Bake Time = 100

===========Finished Reading Parameters============

Reading Initial Conditions from SPLAT ...
```

```
Simulating the Top 22 Layers of Resist Matrix

Done Reading Initial Conditions ...

Initializing Triangulation ...

Triangulation Done ...

Allocating Memory for Data Structures ...

Data Structure Set Up Done ...

Starting Systems Solution ...

Program executed successfully. Steady State reached
before specified PEB time

Total Simulation Time : 1.45250000E+00 minutes

Writing SILY input file: hydro.in ...
Writing Matlab input file: bake.ext ...
To Acid Sites concentration in resist
run 'viewsplat' in matlab

To Activated Sites concentration in resist
run 'viewbake' in matlab

To visualize time evolution of PEB variables
run 'bdf2' in matlab

************** S T O R M **************
```

This simulation takes approximately 1.5 minutes on a DEC Alpha Personal Workstation (600MHz machine). A successful run of BAKE should output *hydro.in* to the Sily2D directory.

4.3 SPLAT and BAKE Outputs for DUV Resist Cross

section

The final result obtained from running BAKE can be viewed using MATLAB. Type

ws% matlab

In MATLAB, type

>>viewsplat

You will be prompted to input the number of x points per layer. This is the first number in *bake.in* file on line 2. You will then be prompted to enter the simulation region. Make sure you enter this in an array format [ ]. The following output from the above example is shown for illustration:

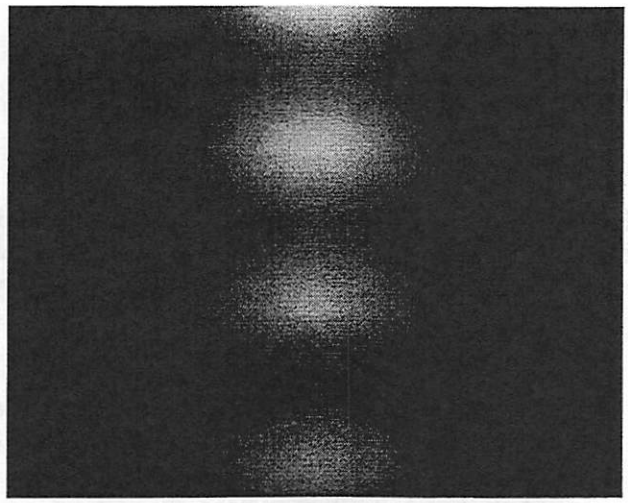
```
>> viewsplat
The number of xpoints can be found in bake.in file
line 2, first #
Enter this number now : 13
Enter simulation region [xmin xmax ymin ymax] : [0 .25
0 .2]

Warning: Duplicate x-y data points detected: using
average of the z values
> In /usr/sww/matlab-5.0/toolbox/matlab/polyfun/grid-
data.m at line 60
 In /home/ebo/SRC/backup/Bake/viewsplat.m at line 53
```

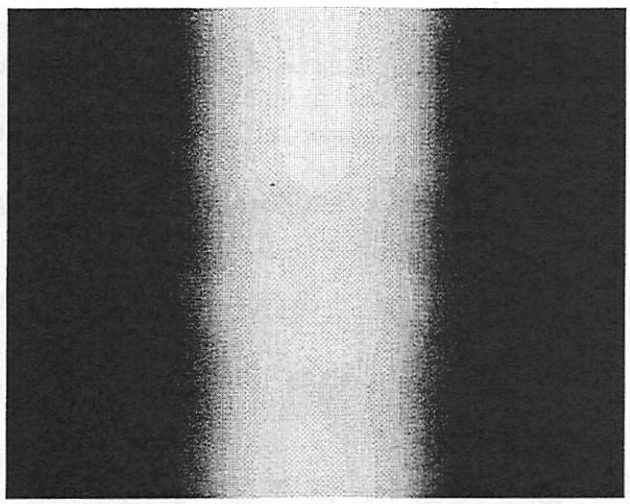
The following two commands plot the output of BAKE on a different graph window.

>>figure

>>viewbake

The to plots generated should look like the ones shown below.





4.4 Exposure Bake of DUV Resist Top View

The following command is used to simulate a top view version of SPLAT.The entire SPLAT input file, *splat.tv,* is shown below. For more information on these commands, please look at the explanation given under SPLAT in the Physical Models section of this manual.

```
# top view SPLAT example for uvIIHS resist.
```

```
# example feature 1
2: wavelength=0.248 microns;
3: NA=0.45;
4: defocus=0.0 microns;
5: partial coherence=0.5;
6: simulated region is 2 x 2 um at 0 transmittance;
7: (0.8,0.0) 0.4 um x 0.8 um at 1 transmittance;
7: (0.0,1.2) 2.0 um x 0.4 um at 1 transmittance;
10;
11: 0.4 0.4 1.2 1.2 25 25 'tv.cnt';
0: end;
```

To run a top view SPLAT input deck, type

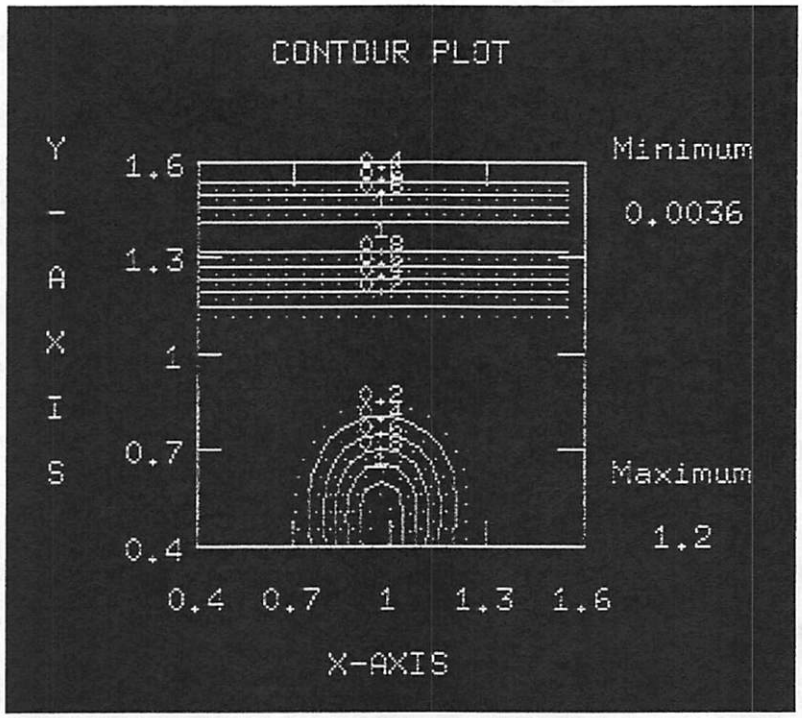
ws% splat < *splat.tv*

This simulation will create a 2D intensity contour files called *tv.cnt* in the current directory.

The 2D intensity contours can now be viewed by typing:

ws%contour -cstep 0.1 *tv.cnt*

A window should appear with a contour plot looking like the following:



4.5 Postexposure of DUV resist (top view)

We are now ready to run BAKE. Make sure to move *tv.cnt* to *bake.in* in your Bake directory and choose the following PEB parameters in *bake.par*:

```
1.10e-3
5.50
0.556
1.0
1.7
0.0
100.0

# Do: Diffusivity Coefficient
# K_1: Reaction RatRate e Coefficient
# K_2: Acid Loss Rate
# omega: Nonlinear Diffusion Coupling Coefficient
# fit: Acid fitting parameter
# k_4: Acid Sites Desorption rate constant
```

```
# t_final: Total Bake Time
```

Type:

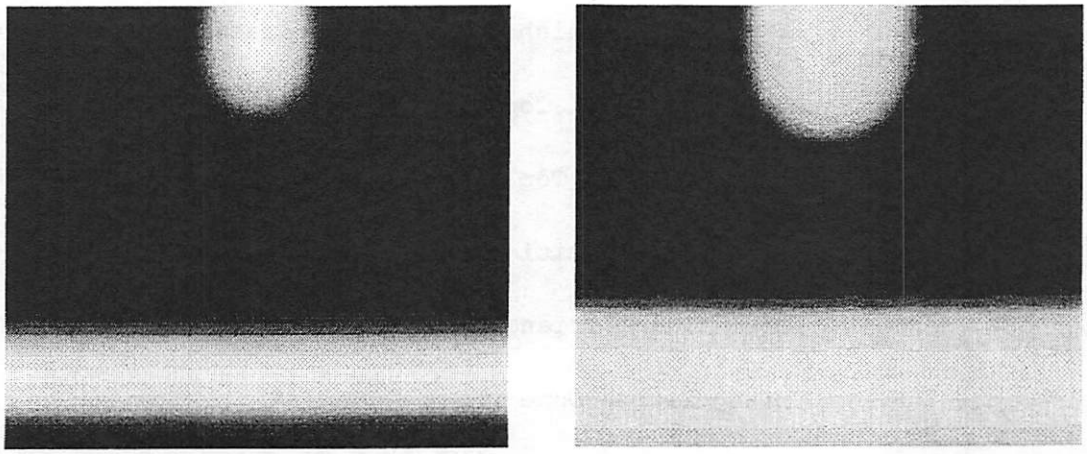
ws% bake

to run the PEB simulation. You should see the following output:

```
************ S T O R M ************


========Reading Post Exposure Bake Parameters=======

Diffusivity Coefficient = 0.0011
Reaction Rate Coefficient = 5.5
Acid Loss Rate Coefficient = 0.556
Nonlinear Diffusion Coupling Coefficient = 1
Acid Fitting Parameter (m) = 1.7
Acid Sites Desorption rate constant = 0
Total Post Exposure Bake Time = 100

============Finished Reading Parameters============

Reading Initial Conditions from SPLAT ...

Simulating the Top 25 Layers of Resist Matrix

Done Reading Initial Conditions ...

Initializing Triangulation ...

Triangulation Done ...

Allocating Memory for Data Structures ...

Data Structure Set Up Done ...

Starting Systems Solution ...

Program executed successfully. Steady State reached
before specified PEB time
```

```
Total Simulation Time : 8.34416667E+00 minutes

Writing SILY input file: hydro.in ...
Writing Matlab input file: bake.ext ...

To Acid Sites concentration in resist
run 'viewsplat' in matlab

To Activated Sites concentration in resist
run 'viewbake' in matlab

To visualize time evolution of PEB variables
run 'bdf2' in matlab

************** S T O R M **************
```

This simulation takes approximately 8.53 minutes on a DEC Alpha Personal Workstation (600MHz machine). This is what the results of this simulation look like in MATLAB:



4.6 Exposure of TSI Resist

The remaining sections of this chapter give an example of simulating the TSI Lithography process. We will simulate the Exposure, postexposure bake and silylation processes.

The following command is used to simulate thin-film SPLAT.

```
stmt 35: resist layer = RESIST_LAYER 'film.tsi'
```

"film.tsi" contains the thin-film SPLAT parameter values to be used in this particular simulation.. The parameters are chosen so as to limit the exposure to the near top region of the reist.

The entire SPLAT input file *splat.tsi* is shown below. For more information on these commands, please look at the explanation given under SPLAT in the Physical Models section of this manual.

```
# Fast thin-film SPLAT example for TSI resist.
# 0.25 um feature
stmt 1: print = 3
stmt 2: lambda = 0.248 um
stmt 3: na = 0.6
stmt 4: defocus = 0.0 mode = 1
stmt 5: sigma = 0.5
stmt 6: mask = 0.3 x 0.3 trans = 0;
stmt 7: cutout = (0.00, 0.00) 0.125 x 0.3 trans = 1;
stmt 35: resist layer = RESIST_LAYER 'film.tsi'
stmt 31: mag = 5 mode = 0;
stmt 32: 50; #table look-up
stmt 10: 0 2;
stmt 14: cutline from (-0.3, 0.0) to (0.3, 0.0)
& with 31 points mode = 1 in 'tsi.RESIST_LAYER'
end 0

The thin-film SPLAT input file
4.73 -0.14 1.0 0.0
1.68 20.74 0.2 0.012 5 0.8 90
```

To run a thin-film SPLAT input deck use the **run.layers** command. Type

ws% run.layers *splat.tsi* 1 90

This simulation will create 90 different 1D intensity contour files in the current directory.

Now, in order to create a 2D intensity file that is accepted by BAKE, type

ws% merge.layers uvII.RESIST_LAYER 1 90 >*tsi.cnt*

This will merge the 1D intensity contour files created using the **run.layers** command to create a 2D intensity contour file *tsi.cnt*.

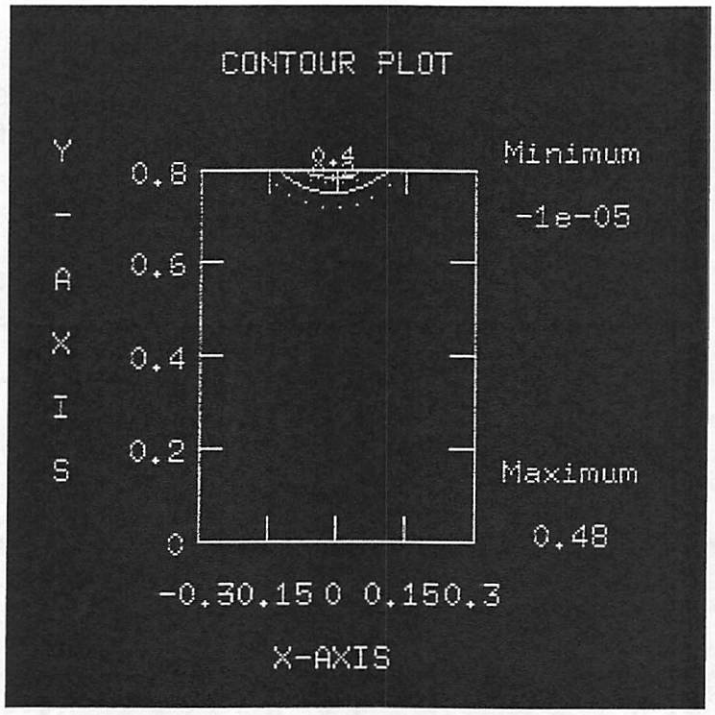
The output file needs to be edited to reflect the spatial dimensions of the photoresist. In our example, the x-axis domain is (-0.3, 0.3) and the y-axis domain is (0.0, 0.8) um. The number of x-points is 31 (from stmt 14). To do this, edit *tsi.cnt* with your favorite editor (e.g. vi, emacs, etc.) and make the following changes:

```
xmin --> 0.3
xmax --> -0.3
ymin --> 0.0
ymax --> 0.8
nxpts --> 31
```

The 2D intensity contours can now be viewed by typing:

ws%contour -cstep 0.1 ***tsi.cnt***

A window should appear with a contour plot looking like the following:



We choose the following domain for our simulation (xmin, xmax) = (0, 0.2) and (ymin, ymax) = (0.7, 0.8) by typing:

ws%sim_region ***uvII.cnt*** *0.1 0.25*

You should see the following:

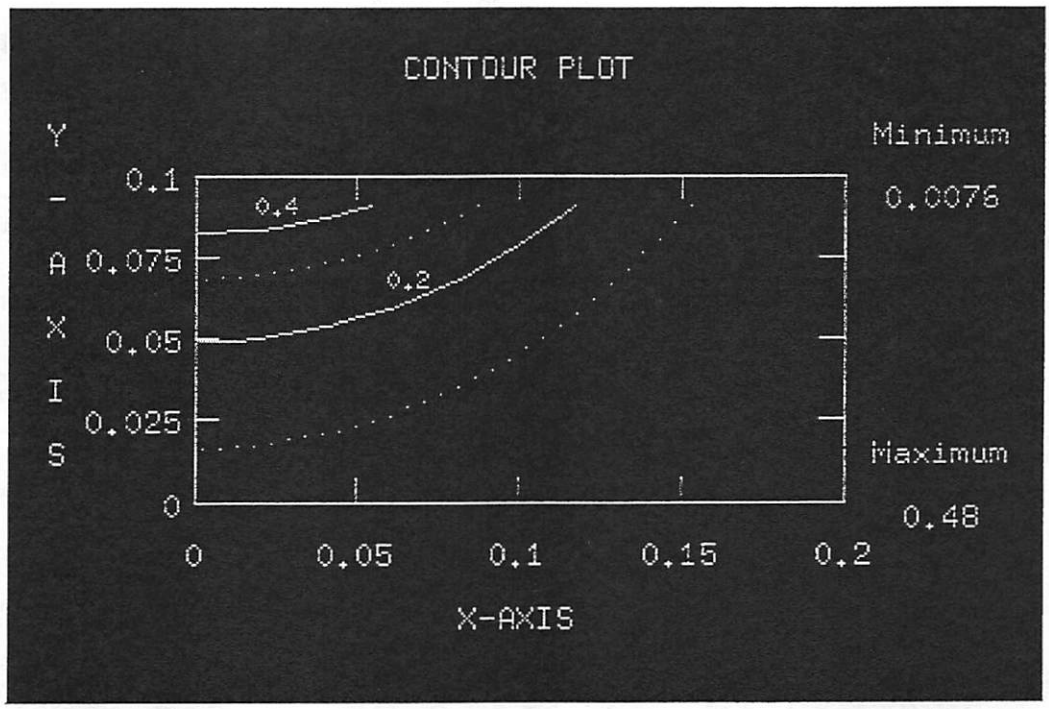
```
%sim_region tsi.cnt .1 .2
Original thickness = .8
```

```
Original width = .3
Original matrix = (90 x 31)
New thickness = .2
New width = .25
New matrix = (11 x 10)
Output file is '../Bake/bake.in'
```

This will outputs the file ***bake.in*** containing the intensity contours of the specified region for BAKE and/or SILY2D simulation. Go to Bake directory and type

ws%contour -cstep 0.1 *bake.in*

to view the new simulation domain. I should look like this:



4.7 Postexposure Bake of TSI resist

We are now ready to run BAKE. Make sure you are in your Bake directory and choose the following PEB parameters for *bake.par*:

```
1.10e-3
5.50
```

```
0.556
1.0
1.7
0.0
100.0

# Do: Diffusivity Coefficient
# K_1: Reaction RatRate e Coefficient
# K_2: Acid Loss Rate
# omega: Nonlinear Diffusion Coupling Coefficient
# fit: Acid fitting parameter
# k_4: Acid Sites Desorption rate constant
# t_final: Total Bake Time
```

Type:

ws% bake

to run the PEB simulation. You should see the following output:

```
%bake
************* S T O R M ************


=======Reading Post Exposure Bake Parameters======

Diffusivity Coefficient = 0.0011
Reaction Rate Coefficient = 5.5
Acid Loss Rate Coefficient = 0.556
Nonlinear Diffusion Coupling Coefficient = 1
Acid Fitting Parameter (m) = 1.7
Acid Sites Desorption rate constant = 0
Total Post Exposure Bake Time = 100

==========Finished Reading Parameters============

Reading Initial Conditions from SPLAT ...

Simulating the Top 11 Layers of Resist Matrix

Done Reading Initial Conditions ...
```

```
Initializing Triangulation ...

Triangulation Done ...

Allocating Memory for Data Structures ...

Data Structure Set Up Done ...

Starting Systems Solution ...

Program executed successfully. Steady State reached
before specified PEB time

Total Simulation Time : 8.02166667E-01 minutes

Writing SILY input file: hydro.in ...
Writing Matlab input file: bake.ext ...
Writing SILY input file: sily.in ...

To Acid Sites concentration in resist
run 'viewsplat' in matlab

To Activated Sites concentration in resist
run 'viewbake' in matlab

To visualize time evolution of peblation variables
run 'bdf2' in matlab

************** S T O R M **************
```

This simulation takes appoximately 0.8 minutes on a 200MHz SUN Ultra Ultra Sparc machine.

4.8 SPLAT and BAKE Outputs of TSI Resist

The final result obtained from running BAKE can be viewed using MATLAB. Type

%matlab

In MATLAB, type

>>viewsplat

You will be prompted to input the number of x points per layer. This is the first number in *bake.in* file on line 2. You will then be prompted to enter the simulation region. Make sure you enter this in an array format [ ]. The following output from the above example is shown for illustration:

```
>> viewsplat
The number of xpoints can be found in bake.in file
line 2, first #
Enter this number now : 10
Enter simulation region [xmin xmax ymin ymax] : [0 .2
0 .1]

Warning: Duplicate x-y data points detected: using
average of the z values
```

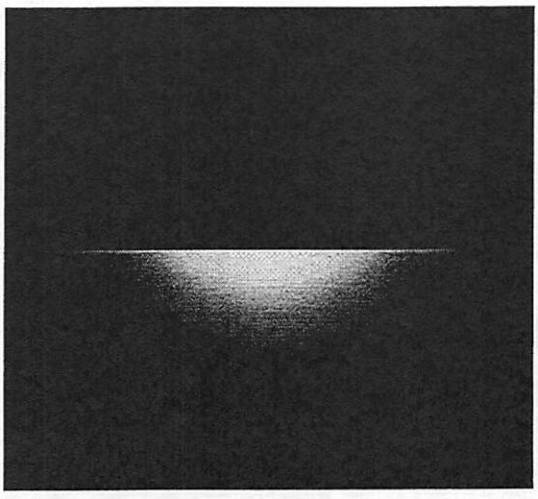
etc..

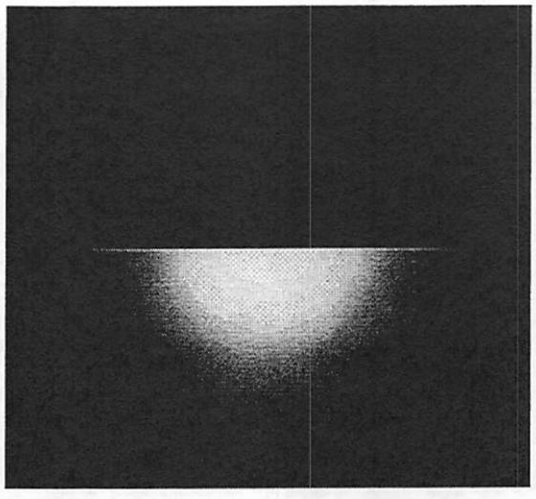
The following two commands plot the output of BAKE on a different graph window.

>>figure

>>viewbake

The to plots generated should look like the ones shown below.





4.9 Silylation of TSI resist without prior PEB

SILY2D expects the input files called *bake.in* (sim_region output) and *hydro.in* (BAKE output).. If SILY2D is being run without a previous BAKE run, any image file from SPLAT must be moved to *bake.in* for SILY2D to execute properly.

To run SILY2D, the silylation parameter values must be specified in the file *sily.par*.

Example SILY2D parameters input file (*sily.par*)

```
1.0e-3
1.0
0.01
1.0
1.0
100.0
500.0
10.0
5.5e-05
0.001
0.634
0.01
100.0

# D: Diffusivity Coefficient
# K_1: Reaction RatRate e Coefficient
# psi: Nonlinear Stress Coupling Coefficient
# omega: Nonlinear Diffusion Coupling Coefficient
# t_relax: Polymer Relaxation Time
# eta: Polymer Viscosity
# alpha: Young Modulus?
# P: Chamber Pressure
# k_3: Silylation Agent Adsorption rate constant
# k_4: Silylation Agent Desorption rate constant
# E_max: Polymer Swelling Parameter (get rid of
later)
# A_max: Polymer Swelling Parameter (get rid of
later)
# t_final: Total Silylation Time
```

Type:

ws% sily -splat

to run SILY2D without prior PEB simulation. You should see the following output:

```
% sily -splat

**************** S T O R M *************

Simulating Silylation WITHOUT prior Postexposure Bake

===========Reading Silylation Parameters==========

Diffusivity Coefficient = 0.001
Reaction Rate Coefficient = 1
Nonlinear Stress Coupling Coefficient = 0.01
Nonlinear Diffusion Coupling Coefficient = 1
Polymer Relaxation Time = 1
Polymer Viscosity = 100
Polymer Young Modulus = 500
Chamber Pressure = 10
Silylation Agent Adsorption rate constant = 5.5e-05
Silylation Agent Desorption rate constant = 0.001
Maximum Expanded Site = 0.63452
Polymer Swelling Parameter (get rid of later) = 0.01
Total Silylation Time = 100

============Finished Reading Parameters==========

Simulating the Top 11 Layers of Resist Matrix

Done Reading Initial Conditions ...

Initializing Triangulation ...

Triangulation Done ...

Allocating Memory for Data Structures ...

Data Structure Set Up Done ...
```

```
Starting Systems Solution ...

Total Simulation Time : 8.80555556E-01 minutes
Writing Matlab input file: sil.ext ...
Writing Matlab input file: bdf.ext ...


To visualize silicon concentration in resist
run 'sily' in matlab

To visualize time evolution of silylation variables
run 'bdf2' in matlab

**************** S T O R M *************
```

This simulation takes appoximately 0.88 minutes on a DEC Alpha Personal Workstation (600MHz machine).

4.10 Viewing SILY2D without PEB

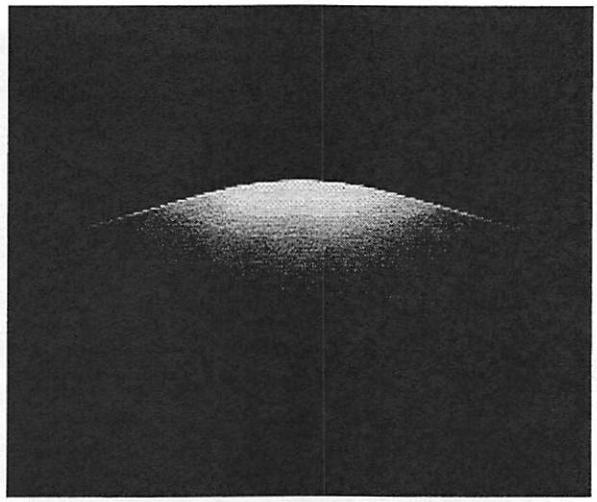
The final result obtained from running SILY2D without PEB can be viewed using MATLAB. Type

%matlab

In MATLAB, type

>>viewsily

This will generate a plot similar to the one shown below.



4.11 Silylation of TSI resist with prior PEB

To simulate silylation after PEB type:

ws% sily -bake

You should see the following output:

```
% sily -bake
Simulating Silylation WITH prior Post Exposure Bake

============Reading Silylation Parameters===========

Diffusivity Coefficient = 0.001
Reaction Rate Coefficient = 1
Nonlinear Stress Coupling Coefficient = 0.01
Nonlinear Diffusion Coupling Coefficient = 1
Polymer Relaxation Time = 1
Polymer Viscosity = 100
Polymer Young Modulus = 500
Chamber Pressure = 10
Silylation Agent Adsorption rate constant = 5.5e-05
Silylation Agent Desorption rate constant = 0.001
Maximun Expanded Site = 0.53452
```

```
Polymer Swelling Parameter (get rid of later) = 0.005
Total Silylation Time = 100

==========Finished Reading Parameters===========

Reading Initial Conditions from BAKE ...

Simulating the Top 11 Layers of Resist Matrix

Done Reading Initial Conditions ...

Initializing Triangulation ...

Triangulation Done ...

Allocating Memory for Data Structures ...

Data Structure Set Up Done ...

Starting Systems Solution ...

Total Simulation Time : 8.86666667E-01 minutes

Writing Matlab input file: sil.ext ...
Writing Matlab input file: bdf.ext ...

To visualize silicon concentration in resist
run 'sily' in matlab

To visualize time evolution of silylation variables
run 'bdf2' in matlab

*************** S T O R M *************
```

This simulation takes appoximately 0.89 minutes on a DEC Alpha Personal Workstation (600MHz machine).

4.12 Viewing SILY2D with PEB

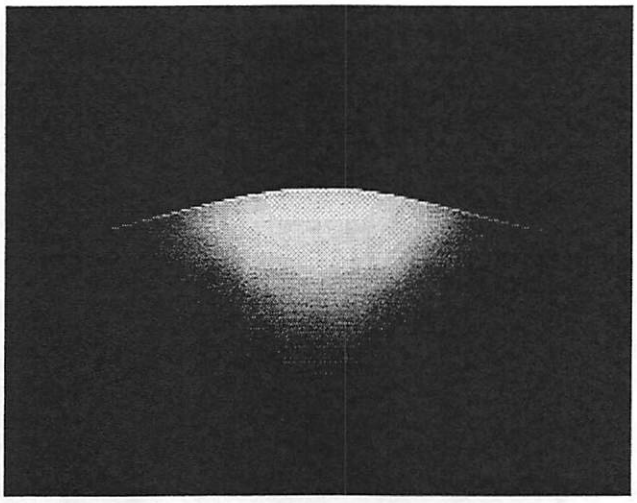
The final result obtained from running SILY2D without PEB can be viewed using MATLAB. Type

%matlab

In MATLAB, type

>>viewsily

This will generate a plot similar to the one shown below.



5.0 Physical Models in STORM

This section presents an overview of the physical models in STORM. The presentation moves from optical applications to postexposure bake and silylation.. For more information, see the references listed in the appendix.

5.1 Optical Image Models and Algorithms - SPLAT

The optical image simulation is carried out by SPLAT version 5.0. This thin-film version of SPLAT allows high NA images to be calculated within the resist layer.

The algorithm is based on Hopkin's approach as extended by Michael Yeung to include thin-film effects in the pupil function. SPLAT version 5.0 also allows aberrations to be specified in Zernike polynomials or wave front maps. Modified illuminations in the form of quadrupole, annular or illumination maps can be specified. Phase shift masks and inlens filtering are also included.

The program is capable of simulating the effects of lens aberrations, apodization, spatial filtering, focus and magnification effects for high NA, and modified illumination, and the five primary lens aberrations: coma, astigmatism, distortion, spherical aberration, and curvature or defocus. Apodization schemes, such as the Hitachi spatial filter, can be added to the illumination system. Annular, offaxis and quadrupole illumination sources can be also be simulated. As mentioned before, the latest version of SPLAT (version 5.0) which comes with this release is also capable of modelling thin-film interference effects.

Here are a few of the more commonly used SPLAT commands from the SPLAT user guide:

[STATEMENT] 1 *printlevel*

STATEMENT 1 sets the level of diagnostic output generated by the program. PRINTLEVEL=1 produces no diagnostic output, while PRINTLEVEL=2 causes the program to echo the input lines to the output. PRINTLEVEL=3, on the other hand, is primarily for interactive/diagnostic use and causes the program to report when statement execution is completed by displaying the input parameters. The default value of PRINTLEVEL is 1.

[STATEMENT] 2 *wavelength*

This statement sets the wavelength, in micrometers, of the light used to illuminate the mask. Currently, the program is set up to accept only a single wavelength. The default is WAVELENGTH = 0.436 μm.

[STATEMENT] 3 *numerical.aperture*

The imaging system is projection-type, with an imaging lens numerical aperture equal to NUMERICAL.APERTURE. Default NUMERICAL.APERTURE is 0.28.

[STATEMENT] 4 *defocus[mode]*

The image can be calculated at a plane other than the plane of best focus. The distance from the plane of best focus, DEFOCUS, is measured in micrometers and defaults to 0.0 Mm. Positive DEFOCUS is defined as being below the gaussian plane, while negative DEFOCUS is above it.MODE is an integer flag used to specify the type of defocus. MODE = 1 specifies that true defocus ($\sqrt{1-\rho^2}$) will be used and is the default mode. MODE = 0 specifies that approximate defocus ($1-\rho^2/2$) will be used.

[STATEMENT] 5 *sigma [sigma_in]*

STATEMENT 5 sets the partial coherence factor, SIGMA, of the imaging system. Only values of partial coherence that lie between 0.0 (full coherence) and 1.0 (partial coherence) are accepted by the program. SIGMA defaults to 0.7, a value common to most projection printers. SIGMA_IN is used for annular illumination schemes. It indicates the radius of the illumination cone which will be blocked out. The actual illumination occurs only in the annulus from SIGMA_IN to SIGMA. SIGMA_IN defaults to 0.0.

[STATEMENT] 6 *xlength ylength [transmittance [scale]]*

The working area is the size of the mask which is to be imaged and is specified by XLENGTH and YLENGTH in micrometers. This specified area is actually only the first quadrant in Cartesian coordinates. The total/unfolded area is bounded by the coordinates (-XLENGTH,-YLENGTH), (- XLENGTH,YLENGTH), (XLENGTH,YLENGTH), (XLENGTH,-YLENGTH), and a periodic extension in both the X and Y directions is assumed. The optional TRANSMITTANCE is the initial transmittance of the mask (usually either 1 for transparent or 0 for opaque). The image calculation time is approximately proportional to the square of the imaged area, and to avoid long computation times, field sizes with total areas greater than 16 micrometers squared should be avoided. One way to do this is to take advantage of symmetry - see STATEMENT 7 for details. The default for TRANSMITTANCE is 0, while XLENGTH and YLENGTH both default to 2.0 micrometers. SCALE is a positive real number that can be used to scale the mask. This scale factor will also apply to any STATEMENTs 7, 8, 27, and 28 that come after a STATEMENT 6.

[STATEMENT] 7 *xcoord ycoord xlen ylen transmittance [phase]*

This statement is used to add a rectangular feature to the mask. The rectangle specified is mirrored across the x and y axes. Rectangles may overlap other rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0.

XCOORD = x coordinate of lower left corner of rectangle

YCOORD = y coordinate of lower left corner of rectangle

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

TRANSMITTANCE= transmittance of the rectangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

As an example, to specify a 2 μm x 2 μm transparent square in a 5 μm x 5 μm opaque mask, the following statements could be used:

STATEMENT 6: 5 um x 5 um at 0 transmittance;

STATEMENT 7: (2.0,2.0) 2.0 um x 2.0 um at 1 transmittance;

However, symmetry can and should be utilized to reduce the computation time by locating the center of the rectangle at the origin of the coordinate system in the following manner:

STATEMENT 6: 2 um x 2 um at 0 transmittance;

STATEMENT 7: (0.0,0.0) 1.0 um x 1.0 um at 1 transmittance;

In the last two input statements above, a 1 μm x 1 μm transparent square is defined in a 2 μm x 2 μm opaque mask. Because the program automatically makes even periodic extensions in the x and y directions, the region specified by the user is actually only the first quadrant in the cartesian coordinate system; the other three quadrants are the mirror images of the first, reflected across the x and y-axes respectively. Therefore, the 1 μm x 1 μm square defined above is actually only part of a 2 μm x 2 μm square, with the center of symmetry of that larger square located at the origin. In a similar manner, an opaque square on a transparent mask can be defined using the statements below, where the transmittance is 1 for the background, and -1 for the square.

STATEMENT 6: 2 um x 2 um at 1 transmittance;

STATEMENT 7: (0.0,0.0) 1.0 um x 1.0 um at -1 transmittance;

[STATEMENT] 8 *xcoord ycoord xlen ylen transmittance [phase]*; or

[STATEMENT] 8 *x1 y1 x2 y2 x3 y3 transmittance [phase]*;

Add a triangular aperture to the mask. The triangle specified is mirrored across the x and y axes. Triangles may overlap other triangles and rectangles - it is the user's responsibility to make sure no part of the mask has transmittance greater than 1 or less than 0. The triangle can be specified in two ways - as a right-angled triangle defined by the corner (right angle) point, base and height or as a general triangle defined by three points. Again, even periodic extensions in the x and y directions are assumed. Please note that if only six arguments are provided, the program assumes that the triangle being defined is a right- angled triangle.

XCOORD = x coordinate of corner of right-angled triangle

YCOORD = y coordinate of corner of right-angled triangle

XLEN = base length of right-angled triangle (may be negative to flip triangle)

YLEN = height of right-angled triangle (may be negative to flip triangle)

X1 = x coordinate of point defining general triangle

Y1 = y coordinate of point defining general triangle

X2 = x coordinate of point defining general triangle

Y2 = y coordinate of point defining general triangle

X3 = x coordinate of point defining general triangle

Y3 = y coordinate of point defining general triangle

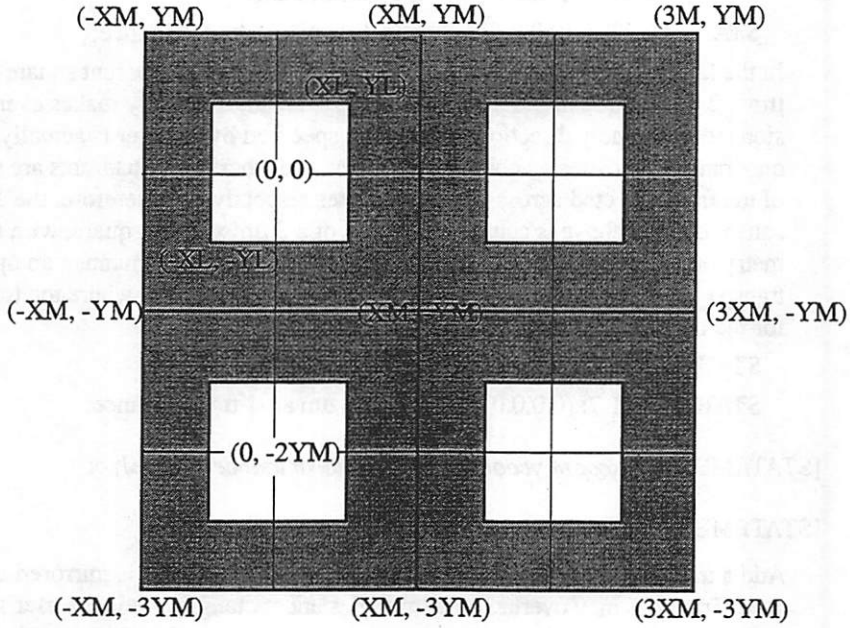
TRANSMITTANCE= transmittance of the triangle relative to the current transmittance of the mask where it is to be placed.

PHASE = phase angle (normally 0 degrees). This option is used for phase shifted masks.

Input Statements:

Statement 6: XM x YM @0;

Statement 7: (0,0) (XL,YL) @1;



[STATEMENT] 9 *nx ny [mode [xllc yllc xlen ylen [difflag]]] ['filename']*;

STATEMENT 9 is used to calculate the transmittance profile of the mask. The transmittance at any point is the sum of the working area(mask) transmittance (see STATEMENT 6) and the transmittances of all rectangles or triangles defined by STATEMENTs 7 and 8 that cover that point. The output file consists of a list of transmittances at points on a grid defined by NX and NY. The default is to calculate for the entire working area, but the user can specify a smaller (or larger) area. MODE defines the type of output obtained from this statement. MODE = 0 produces output to a specified file, MODE = 1 sends a crude contour plot to the terminal, while MODE = 2 does both.

NX = number of divisions along x axis (default = 20)

NY = number of divisions along y axis (default = 20)

MODE = type of output desired

XLLC = x coordinate of lower left corner of rectangle to be plotted

YLLC = y coordinate of lower left corner of rectangle to be plotted

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

DIFFLAG = 1 to limit spatial frequencies (normally not used)

FILENAME= name of file into which to store the calculated intensities.

[STATEMENT] 10 *[mode [force]] ['filename']*;

STATEMENT 10 orders the program to calculate the transmission cross-coefficients (TCCs) as well as the Fourier coefficients of the image intensity profile. This statement can take a long time to execute. MODE = 1 stores the TCCs in *binary* form in the file FILENAME. FORCE is an integer flag which forces the program to choose which of the 3 integration routines should be used for TCC calculation. FORCE = 0 is the default, FORCE = 1 uses a 1-dimensional integration routine, while FORCE = 2 uses a 2-dimensional integration scheme. When aberrations other than approximate defocus are used or when magnification is set, FORCE = 2 is used automatically.

[STATEMENT] 11 *[xllc yllc xlen ylen [nx ny]] ['filename']*;

This statement calculates the image intensity within a rectangle specified by XLLC, YLLC, XLEN, YLEN. The intensities are calculated at each point of a rectilinear NX by NY equally spaced grid within that rectangle. (Note: Maximum NX*NY is 10000) This output file may be used with a 3-dimensional plotting package to provide either a 3- dimensional intensity profile or an intensity contour plot. The parameters are:

XLLC = x coordinate of lower left corner of rectangle to be plotted

YLLC = y coordinate of lower left corner of rectangle to be plotted

XLEN = length of rectangle in x direction

YLEN = length of rectangle in y direction

NX = number of grid-points in the x direction (default is 50)

NY = number of grid-points in the y direction (default is 50)

FILENAME= name of file into which to store the calculated intensities.

[STATEMENT] 12 *'filename'*

Save the Fourier coefficients that describe mask transmittance and image intensity. This input line will produce a relatively large (approx. 30 kbytes) binary output file, which contains the Fourier coefficients as well as the imaging system parameters - numerical aperture, wavelength, coherence factor, mask size. The coefficients can be reloaded with STATEMENT 13 to generate more plot files of the same mask pattern.

[STATEMENT] 13 *'filename'*

STATEMENT 13 is used to load the Fourier mask and image coefficients previously saved with STATEMENT 12. Therefore, for a given set of imaging system parameters (N.A., wavelength, sigma, mask size and pattern), the intensity profile need only be calculated once with STATEMENT 10. STATEMENT 12 saves the data computed by STATEMENT 10 for later use, and STATEMENT 13 reloads this data. Note that the TCCs saved via STATEMENT 10 can be used for any mask pattern, as long as the field size remains con-

stant. On the other hand, STATEMENT 12-13 saves the Fourier coefficients only, so these can be used only for one particular mask pattern.

[STATEMENT] 14 *xi yi xf yf [npts [mode]] ['filename']*;

Calculate the image intensity profile along the line joining points (XI,YI) and (XF,YF). The number of points along that profile and is optionally specified by NPTS, which defaults to 50. The intensity profile is output in the same format as a SAMPLE f77punch7 file. The 'x' values in the file represent the distance along the line from the point (XI,YI), while the 'y' values represent the light intensity normalized to 1.00. Again, MODE is an integer flag, normally 0, used to specify the format of the intensity profile. MODE = 1 outputs the intensities with 5 decimal spaces (as compared to 3 with MODE = 0). This option is particularly useful for analysis of small patterns such as defects, where the intensity is very low.

XI = x coordinate of initial plot point

YI = y coordinate of initial plot point

XF = x coordinate of final plot point

YF = y coordinate of final plot point

NPTS = number of points (default is 50, maximum is 500)

MODE = accuracy flag (defaults to 0)

[STATEMENT] 30 *spot_radius Jx Jy*

STATEMENT 30 allows entry of a single illumination spot. The SPOT_RADIUS is the radius of the illumination spot while (Jx,Jy) indicate its coordinates in the entrance. pupil of the system. The radius of the entrance pupil is normalized to 1. Multiple illumination spots are specified by using up to 10 of these statements. For example, the following commands define a quadrupole illumination scheme for a system with a partial coherence of 0.695. The system partial coherence is computed automatically as the smallest sigma which will overlap all illumination spots.

STATEMENT 30: Spot_radius = 0.2 at (0.35, 0.35);

STATEMENT 30: Spot_radius = 0.2 at (-0.35, 0.35);

STATEMENT 30: Spot_radius = 0.2 at (0.35,-0.35);

STATEMENT 30: Spot_radius = 0.2 at (-0.35,-0.35);

[STATEMENT] 31 *magnification [mode]*

This statement is used to include magnification effects in the calculation of the aerial image. MAGNIFICATION (or more precisely demagnetization) is defined as the object height divided by the image height. MODE = 0, the default mode, specifies that the mask image will be calculated using the mask dimensions at the mask plane. MODE = 1 specifies that the mask image will be calculated using the mask dimensions scaled down by MAGNIFICATION. This statement must be called prior to STATEMENT 6.

[STATEMENT] 32 *[grid_pts [symmetry]]*

This statement turns on the use of the table lookup routines when calculating the TCCs. Instead of recalculating the pupil function every time, the values are stored in a table and subsequently looked up. The main use of this feature is to speed up the thin-film calculations (STATEMENT 35) and it does not provide any speedup for systems with only few aberrations. The number of columns and rows of the lookup table is initialized with dimension *grid_pts* X *grid_pts*. A good number of grid points is 50 for the thin film cases--the results differ by less than 1% from those when not using the table. The maximum number of grid points is 400 and the default is 50. The *symmetry* parameter refers to the use of symmetry in the calculation of the TCCs (STATEMENT 10) which normally reduces the computation time. When *symmetry* = 0, symmetry in the TCC calculation is turned off. This will speed up the simulation time when the mask area is very small since the determination of the symmetry dominates the simulation time. Explicitly disabling the symmetry is only permitted in conjunction with STATEMENT 35. By default *symmetry* = 1 which is the state of normal operation.

[STATEMENT] 35 *resist_layer 'thin-film_data_file'*

Statement 35 allows the image to be calculated within the photoresist. The resist is split up into many thin layers as specified in the *'thin-film_data_file'*. The format of the thin-film input file is given below. The correct number of parameters must be listed on each line and any number of spaces (but no tabs) may be used to separate them.

Thin-film Data File Input Format

substrate_n subsrtate_k medium_n medium_k wavelength

resist_n A B C dose thickness layers

layer1_n layer1_k thickness

layer2_n layer2_k thickness

...

layer12_n layer12_k thickness

The first four parameters are the real and imaginary parts of refractive index of the substrate and medium (Air) respectively. The fifth parameter is the wavelength in the initial medium. The next line specifies the resist properties: the real part of the index of refraction; the resist A, B, C parameters, used to calculate the imaginary part of the index of refraction; dose; thickness; and number of layers. Any lines following the resist parameters specify the real and imaginary parts of the index of refraction of layers under the photoresist and their thicknesses. Up to 12 layers may be specified.

Thin-film Utilities

The following UNIX scripts and or C programs have been created:

run.layers - a unix script to run many thin-film interference simulations using a single input file

Usage: *run.layers splat_input_file start_layer end_layer*

The script searches for the word 'RESIST_LAYER' and replaces it with the current resist layer and then calls SPLAT. At the end of the simulation there will be end_layer 1D intensity files in your current directory. Since this script calls SPLAT, it must be accessible from your current directory.

merge.layers - a unix script used to merge the 1D intensity files created using run.layers into a 2D intensity contour file

Usage: *merge.layers filename_RESIST_LAYER.plot start_layer end_layer*

merge.contours - a unix script used to merge the 2D intensity contour files created using run.layer into 3D intensity file for input in SAMPLE-3D

Usage: *merge.layer filename_RESIST_LAYER.ctr start_layer end_layer*

For an illustration of these scripts please refer to the tutorial on Lithography Simulation in section 3.1 of this document.

5.2 Postexposure bake, Reaction and Diffusion - BAKE

The BAKE program coverts the aerial image and/or the cumulative energy deposited per unit volume to activated sites concentration under appropriate process effects. When an initial aerial image is specified the effect of acid diffusion during the exposure can be calculated.

It is common in resist models to describe the exposure by converting an exposure dose to an initial acid concentration through Dill's ABC parameters. The PEB both diffuses the acid and drives the reaction to alter the solubility of the film. Dissolution models then predict development as a function of extent of reaction in order to arrive at a final resist profile.

The equations utilized for the reaction diffusion kinetics in the BAKE simulator are:

$$\frac{\partial}{\partial t} Cas = K_1(1 - Cas)Ca^m$$

$$\frac{\partial}{\partial t} Ca = \nabla\bullet(D\nabla Ca) - K_2 Ca$$

where Cas is the normalized extent of reaction, K_1 is the reaction rate, Ca is the normalized acid concentration, K_2 is the acid loss rate, m is a fitting parameter, and D is the acid diffusivity. If D is a constant the post exposure bake is *described* by a Fickean process. If the diffusivity is concentration dependent, the PEB is described by a type II diffusion process. The type II diffusion model used in BAKE is of the form D=Doexp(wCas).

5.3 Resist Silylation - SILY2D

The silylation model implemented in STORM is described by the following set of nonlinear differential equations:

$$\frac{\partial S}{\partial t} = -K_1 SH + \nabla Dexp(wE)\nabla S \qquad \frac{\partial H}{\partial t} = -K_1 SH \qquad \frac{\partial U}{\partial t} = K_1 SH - \frac{U}{t_r} \qquad \frac{\partial E}{\partial t} = \frac{U}{t_r}$$

$$\frac{\partial A}{\partial t} = \frac{A_{max}}{E_{max}}\left(\frac{U}{t_r}\right) \qquad \int_w \delta\left(\frac{\partial \varepsilon}{\partial t}\right)\cdot\sigma = \oint_\Gamma \delta(v)\cdot f + \int_w \delta(v)\cdot b$$

where S is the concentration of the silylating agent, H is the concentration of bonding sites, U is the concentration of unexpanded sites, E is the concentration of expanded sites[1], A is the area of the resist film, $\left(\frac{\partial \varepsilon}{\partial t}\right)$ and σ are the resulting strain rate and stress through the polymer film as a function of the induced swelling and resist material parameters. Stress dependence is introduced through the reaction rate $K_1 = K\exp(-(\sigma_n + E_a)/RT)$, where σ_n is the resultant normal stress as calculated from the induced surface tractions and body forces in the film.The nonlinear diffusion model captures the sharp boundary experimentally observed between silylated and unsilylated areas and near linear uptake with processing time.