

DEVELOPMENT OF A MICRO-FABRICATION PROCESS SIMULATOR FOR  
MICRO-ELECTRO-MECHANICAL SYSTEMS (MEMS)

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY  
ALPER YILDIRIM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
MECHANICAL ENGINEERING

DECEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences

---

Prof. Dr. Canan ÖZGEN  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

---

Prof. Dr. Kemal İDER  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

---

Asst. Prof. Dr. Melik DÖLEN  
Supervisor

**Examining Committee Members**

Prof. Dr. Bilgin Kaftanoğlu (METU,ME) \_\_\_\_\_

Asst. Prof. Dr. Melik Dölen (METU,ME) \_\_\_\_\_

Prof. Dr. Metin Akkök (METU,ME) \_\_\_\_\_

Asst. Prof. Dr. Buğra Koku (METU,ME) \_\_\_\_\_

Prof. Dr. Tayfun Akın (METU,EEE) \_\_\_\_\_

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name : Alper YILDIRIM

Signature :

## **ABSTRACT**

### **DEVELOPMENT OF A MICRO-FABRICATION PROCESS SIMULATOR FOR MICRO-ELECTRO-MECHANICAL SYSTEMS (MEMS)**

Yıldırım, Alper

M.S, Department of Mechanical Engineering

Supervisor: Asst. Prof. Dr. Melik Dölen

December 2005, 140 pages

The aim of this study is to devise a computer simulation tool, which will speed-up the design of Micro-Electro-Mechanical Systems by providing the results of the micro-fabrication processes in advance. Anisotropic etching along with isotropic etching of silicon wafers are to be simulated in this environment. Similarly, additive processes like doping and material deposition could be simulated by means of a Cellular Automata based algorithm along with the use of OpenGL library functions. Equipped with an integrated mask design editor, complex mask patterns can be created by the software and the results are displayed by the Cellular Automata cells based on their spatial location and plane. The resultant etched shapes are in agreement with the experimental results both qualitatively and quantitatively.

Keywords: Wet Etching, Anisotropic Etching, Doping, Cellular Automata, Micro-fabrication simulation, Material Deposition, Isotropic Etching, Dry Etching, Deep Reactive Ion Etching

## ÖZ

### MİKRO-ELEKTRO-MEKANİK-SİSTEMLER İÇİN BİR MİKRO-ÜRETİM SİMÜLATÖRÜNÜN GELİŞTİRİLMESİ

Yıldırım, Alper

Yüksek Lisans, Makine Mühendisliği Bölümü

Tez Yöneticisi: Y. Doç.. Dr. Melik Dölen

Aralık 2005, 140 sayfa

Bu çalışmanın amacı mikro-fabrikasyon proseslerinin sonuçlarını önceden sağlayarak Mikro-Elektro-Mekanik-Sistemlerinin dizaynını hızlandıracak bir bilgisayar programı tasarlamaktır. Silikon plakalarının yönbağımlı kazınma ve yönbağımsız kazınmaları bu ortamda benzetimlenecektir. Benzer olarak, katkılama ve kaplama gibi ekleme yöntemleri de bir hücresel otomat bazlı algoritma ile OpenGL kütüphanesi fonksiyonları kullanılarak, benzetimlenebilecektir. Entegre bir maske dizayn editörüne sahip program ile kompleks maskeler tasarlanabilir ve sonuçlar uzamsal konumları ve buldukları düzlemlere göre hücresel otomat hücreleri olarak ekranda gösterilir. Sonuçta bulunan kazınmış şekiller deneysel sonuçlarla nicelik ve nitelik bakımından uzlaşmaktadır.

Anahtar Kelimeler: Islak kazıma, Yönbağımsız Kazıma, Katkılama, Hücresel Otomat, Mikro-üretim Benzetimlenmesi, Malzeme Kaplama, Yönbağımsız Kazıma, Kuru Kazıma, Derin tepkin İyon Kazıması

To My Family

## **ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to my supervisor, Asst. Prof. Dr. Melik DÖLEN for his precious guidance and encouragement throughout the research. To my family, Mahmut, Birsen, Altuğ and Gökalp Yıldırım, and Arzu Baysal, I offer special thanks for their precious love and encouragement during the period of study.

I am indebted to the members of the Department of Mechanical Engineering and all my friends.

## TABLE OF CONTENTS

|   |      |
|---|------|
| PLAGIARISM.....   | iii  |
| ABSTRACT.....   | iv   |
| ÖZ.....   | v    |
| DEDICATION.....   | vi   |
| ACKNOWLEDGMENTS.....  | vii  |
| TABLE OF CONTENTS.....  | viii |
| LIST OF TABLES.....   | xii  |
| LIST OF FIGURES.....  | xiii |
| CHAPTERS  |      |
| 1. INTRODUCTION.....  | 1    |
| 1.1. Organization of Thesis.....                              | 2    |
| 2. REVIEW OF THE STATE OF THE ART.....                        | 4    |
| 2.1. Introduction.....  | 4    |
| 2.2. Micro-fabrication Techniques.....                        | 5    |
| 2.2.1. Photolithography.....                                  | 5    |
| 2.2.2. Thin Film Deposition and Doping.....                   | 7    |
| 2.2.2.1. Oxidation.....                                       | 7    |
| 2.2.2.2. Chemical Vapor Deposition (CVD).....                 | 7    |
| 2.2.2.3. Physical Vapor Deposition (PVD).....                 | 8    |
| 2.2.2.4. Doping.....  | 9    |
| 2.2.4. Wet Chemical Etching.....                              | 10   |
| 2.2.5. Dry Etching (Plasma Etching).....                      | 14   |
| 2.2.6. Micromachining Techniques.....                         | 15   |
| 2.3. Simulation Programs.....                                 | 17   |
| 2.3.1. Anisotropic Silicon Etching Program (ASEP).....        | 17   |
| 2.3.2. SEGS Simulator.....                                    | 18   |
| 2.3.3. Suzana.....  | 22   |
| 2.3.4. Anisotropic Etch Simulator (AnisE).....                | 24   |
| 2.3.5. Anisotropic Crystalline Etching Simulation (ACES)..... | 28   |
| 2.3.6. SIMODE.....  | 31   |
| 2.3.7. Commercial Codes.....                                  | 37   |

|  |    |
|--|----|
| 2.3.7.1. Memulator.....  | 37 |
| 2.3.7.2. Athena .....  | 38 |
| 2.4. Cellular Automata.....                                    | 39 |
| 2.5. Closure.....  | 42 |
| 3. WAFER EDITOR.....   | 43 |
| 3.1. Introduction.....   | 43 |
| 3.2. Crystal Orientations.....                                 | 43 |
| 3.3. Wafer Size Modifications.....                             | 47 |
| 3.4. Visualization.....  | 50 |
| 3.5. Closure.....  | 51 |
| 4. MASK EDITOR.....  | 52 |
| 4.1. Introduction.....   | 52 |
| 4.2. Mask Editor Features.....                                 | 52 |
| 4.3. Mask Primitives.....                                      | 53 |
| 4.4. Modifications.....  | 54 |
| 4.5. Transfer of Mask Pattern onto the Wafer.....              | 56 |
| 4.6. Visualization.....  | 56 |
| 4.6.1. Creating Primitives (Tessellation).....                 | 57 |
| 4.6.2. Modifying Primitives (Selection).....                   | 59 |
| 4.7. Sample Mask Shapes.....                                   | 60 |
| 4.8 Closure .....  | 65 |
| 5. PROCESS EDITOR.....   | 66 |
| 5.1. Introduction.....   | 66 |
| 5.2. Editor Features.....                                      | 66 |
| 5.3. Simulated Processes.....                                  | 68 |
| 5.3.1. Wet Etching.....  | 68 |
| 5.3.2. Dry Etching.....  | 69 |
| 5.3.3. Doping.....   | 69 |
| 5.3.4. Additive Processes.....                                 | 69 |
| 5.4. Virtual Surface Creation.....                             | 70 |
| 5.5. Plane of the Cells.....                                   | 72 |
| 5.6. Process Application.....                                  | 73 |
| 5.6.1. Wet Etching Modifications.....                          | 73 |
| 5.6.2. Dry Etching (Deep Reactive Ion Etch) Modifications..... | 74 |
| 5.6.3. Doping Modifications.....                               | 77 |

|   |     |
|---|-----|
| 5.6.4. Additive Process Modifications.....                | 78  |
| 5.7. Visualization of data.....                           | 79  |
| 5.7.1. Viewing Transformations.....                       | 81  |
| 5.8. Closure.....   | 83  |
| 6. PROGRAM FEATURES.....                                  | 84  |
| 6.1. Introduction.....                                    | 84  |
| 6.2. MemsEagle User Interface.....                        | 84  |
| 6.3. Wafer Editor.....                                    | 85  |
| 6.4. Mask Editor.....                                     | 86  |
| 6.5. Process Editor .....                                 | 88  |
| 6.6. Viewing.....   | 91  |
| 6.7. Taking Measurements.....                             | 94  |
| 6.8. Closure.....   | 94  |
| 7. VERIFICATION OF SIMULATION RESULTS.....                | 95  |
| 7.1. Introduction.....                                    | 95  |
| 7.2. Simulation Results.....                              | 95  |
| 7.2.1. Anisotropic Wet Etching.....                       | 95  |
| 7.2.1.1. Case 1 for wet etching.....                      | 96  |
| 7.2.1.2. Case 2 for wet etching.....                      | 98  |
| 7.2.1.3. Case 3 of wet etching.....                       | 99  |
| 7.2.1.4. Case 4 of Wet Etching.....                       | 101 |
| 7.2.1.5. Case 5 of Wet Etching.....                       | 104 |
| 7.2.2.6 Case 6 of Wet Etching.....                        | 106 |
| 7.2.2.7. Case 7 of Wet Etching.....                       | 107 |
| 7.2.1.8. Case 8 of Wet Etching.....                       | 109 |
| 7.2.2. Deep Reactive Ion Etching.....                     | 114 |
| 7.2.3. Doping.....  | 116 |
| 7.2.4 Additive Process.....                               | 118 |
| 7.3. Etch Rate Verification.....                          | 119 |
| 7.4. Program Performance.....                             | 120 |
| 7.5. Quantitative Analysis on the Simulation Results..... | 121 |
| 7.6. Closure.....   | 122 |
| 8. CONCLUSION AND FUTUREWORK.....                         | 123 |
| 8.1. Future work.....                                     | 124 |

## APPENDICES

|                                     |     |
|-------------------------------------|-----|
| APPENDIX A: Etch Rates.....         | 126 |
| A.1. KOH Etch Rates.....            | 126 |
| A.2. TMAH Etch Rates.....           | 128 |
| A.3. EDP Etch Rates.....            | 129 |
| APPENDIX B: OpenGL Functions.....   | 131 |
| B.1. glBegin – glEnd functions..... | 131 |
| B.2. glVertex Function.....         | 133 |
| B.3. glLoadName Function.....       | 133 |
| B.4. glSelectBuffer Function.....   | 133 |
| B.5. glRenderMode Function.....     | 135 |
| B.6. gluProject Function.....       | 136 |
| REFERENCES.....                     | 138 |

## LIST OF TABLES

| Table   | Page |
|---|------|
| 2.1: KOH formulations.....  | 12   |
| 2.2: Selectivity of TMAH etchants for dielectrics versus (100) Silicon.....                             | 13   |
| 2.3: Common EDP Formulations.....   | 13   |
| 3.1: Spatial coordinates of neighboring Silicon atoms for various<br>crystallographic orientations..... | 46   |
| 5.1: Etch Rates in $\mu\text{m}/\text{min}$ .....   | 69   |
| 5.2: VS Class Members.....  | 70   |
| 5.3: Number of Neighbors for Different Planes.....  | 73   |
| 5.4: Colors Map in MemsEagle.....   | 81   |
| 7.1: Maximum Depth of the Etched Surfaces .....   | 121  |
| 8.1: Features of the Available Softwares .....  | 123  |
| A.1: KOH Etch Rates.....  | 126  |
| A.2: KOH Etch Rates vs. Composition and Temperature.....  | 127  |
| A.3: TMAH Etching Rates vs. Orientation.....  | 128  |
| A.4: TMAH Etch Rates vs. Composition and Temperature.....   | 119  |
| A.5: EDP Etch Rates vs. Composition and Temperature.....  | 130  |

## LIST OF FIGURES

| Figure   | Page |
|--|------|
| 2.1: Photolithography.....   | 6    |
| 2.2: Miller Indices.....   | 10   |
| 2.3: Anisotropic Wet Etching (100) plane.....                                  | 11   |
| 2.4: Anisotropic Wet Etching (110) plane.....                                  | 11   |
| 2.5: Convex Corner Undercutting.....   | 11   |
| 2.6: Plasma & Reactive Ion Etching.....  | 15   |
| 2.7: Surface Micromachining Example.....                                       | 16   |
| 2.8: Bulk & Surface Micromachining.....  | 16   |
| 2.9: SEGS two-dimensional example.....   | 21   |
| 2.10: SEGS three-dimensional example.....                                      | 21   |
| 2.11: Bonding situation of (100), (110) and (111) surface atoms of Suzana..    | 23   |
| 2.12: Schematic Block Diagram of the CAD Architecture for using Suzana..       | 24   |
| 2.13: Silicon Crystal Structure.....   | 25   |
| 2.14: AnisE User Interface.....  | 27   |
| 2.15: AnisE simulation result.....   | 27   |
| 2.16: Different crystal types used in ACES.....                                | 29   |
| 2.17: Top views and link set types of the different crystals used in ACES..... | 30   |
| 2.18: Mask pattern to ACES.....  | 31   |
| 2.19: Simulation result of ACES.....   | 31   |
| 2.20: Simple Etch Trench of SIMODE.....  | 31   |
| 2.21: Mask Design of SIMODE.....   | 32   |
| 2.22: SIMODE Flow Chart.....   | 33   |
| 2.23: Velocity profile of a SIMODE process.....                                | 34   |
| 2.24: Sidewall Profiles.....   | 34   |
| 2.25: Simulation of one etch step in SIMODE.....                               | 35   |
| 2.26: Construction of three-dimensional shape.....                             | 36   |
| 2.27: ATHENA Framework Architecture.....                                       | 39   |
| 2.28: Example of a Starting Pattern.....                                       | 40   |
| 2.29: Determination of the Neighborhood.....                                   | 40   |

|  |    |
|--|----|
| 2.30: First Generation.....  | 41 |
| 2.31: Second Generation.....   | 41 |
| 2.32: Example of 1-D CA Pascal Triangle.....   | 42 |
| 3.1: Crystal Orientations at different planes.....   | 44 |
| 3.2: Top view of various crystal lattices with different crystallographic orientations. .... | 45 |
| 3.3: Wafer editor.....   | 47 |
| 3.4: Cells generated by Wafer Editor.....  | 49 |
| 3.5: Wafer display.....  | 51 |
| 4.1: Mask Editor Interface.....  | 53 |
| 4.2: Mask Creation using Tessellation.....   | 58 |
| 4.3: Co-centric Rings.....   | 60 |
| 4.4: Obtuse-Angle Shapes.....  | 61 |
| 4.5: Rotated Squares.....  | 62 |
| 4.6: Misaligned Masks.....   | 62 |
| 4.7: Paddle.....   | 63 |
| 4.8: 45° Beams.....  | 63 |
| 4.9: Triangle-Cornered Beams.....  | 64 |
| 4.10: Compensation Fingers.....  | 64 |
| 4.11: Spur Gear Created by the Mask Editor.....  | 65 |
| 5.1: Process Editor.....   | 67 |
| 5.2: Process Selector.....   | 67 |
| 5.3: Micro-Fabrication Dialog Boxes.....   | 68 |
| 5.4: Virtual Surface Creation Flowchart.....   | 72 |
| 5.5: Plane Function Flow.....  | 73 |
| 5.6: Wet Etching Flowchart.....  | 75 |
| 5.7: Dry Etching Flowchart.....  | 76 |
| 5.8: Dry Etch Modification.....  | 77 |
| 5.9: Enlarged Area.....  | 77 |
| 5.10: Additive Process Flowchart.....  | 79 |
| 5.11: Displaying Results.....  | 80 |
| 5.12: Top View of the etch result.....   | 82 |
| 5.13: Etch Result without mask from a different angle.....                                   | 82 |
| 5.14: Etch Result From different angle and scaling.....                                      | 83 |
| 6.1: MemsEagle Interface.....  | 85 |

|  |     |
|--|-----|
| 6.2: Wafer Editor.....   | 86  |
| 6.3: Mask Editor.....  | 87  |
| 6.4: Drawing Mask.....   | 88  |
| 6.5: Process Selection.....  | 89  |
| 6.6: Wet Etching Dialog.....                                       | 90  |
| 6.7: Project Editor.....   | 90  |
| 6.8: Simulation Result.....  | 91  |
| 6.9: View Menu.....  | 92  |
| 6.10: Viewing Doping Concentration.....                            | 92  |
| 6.11: Distance between two points.....                             | 93  |
| 6.12: Output Menu.....   | 93  |
| 6.13: Viewing doping concentration of a point.....                 | 94  |
| 7.1: EDP Etch View.....  | 97  |
| 7.2: EDP Etch 3-D View.....  | 97  |
| 7.3: EDP Etch of (100) Silicon Wafer.....                          | 98  |
| 7.4: Etch profile of <100> wafer flat on.....                      | 98  |
| 7.5: 3-D Etch profile of <100> wafer flat on <110>.....            | 99  |
| 7.6: Co-centered Circular Mask Pattern.....                        | 100 |
| 7.7: MemsEagle Result for Co-centered Circular Mask Pattern.....   | 100 |
| 7.8: Experiment Result for Co-centered Circular Mask Pattern ..... | 101 |
| 7.9: Mask Pattern of Merging Shapes.....                           | 102 |
| 7.10: MemsEagle Simulation Result of the mask.....                 | 102 |
| 7.11: Experimental Result for the Mask.....                        | 103 |
| 7.12: Etch Result After 150 minutes.....                           | 103 |
| 7.13: Experiment Result after 150 min.....                         | 104 |
| 7.14: Paddle Mask.....   | 104 |
| 7.15: Simulation Result after 100min.....                          | 105 |
| 7.16: Experiment Result after 100min.....                          | 105 |
| 7.17: Simulation Results after 30min. and 50min.....               | 106 |
| 7.18: Experiment Results after 50min.....                          | 107 |
| 7.19: MemsEagle Simulation for “Tee”.....                          | 108 |
| 7.20: Experiment Result for “Tee”.....                             | 108 |
| 7.21: Misaligned Mask Pattern.....                                 | 109 |
| 7.22: MemsEagle Result after 50min.....                            | 110 |
| 7.23: Experiment Result after 50min.....                           | 110 |

|  |     |
|--|-----|
| 7.24: MemEagle Result after 100min.....                        | 111 |
| 7.25: Experiment Result after 100min.....                      | 111 |
| 7.26: Simulation of the 5°-rotated mask.....                   | 112 |
| 7.27: 3-D View of Simulation Result.....                       | 112 |
| 7.28: Experiment Result of the 5° rotated mask.....            | 112 |
| 7.29: Simulation of the 15° rotated mask.....                  | 113 |
| 7.30: Experiment Result of the 15° rotated mask.....           | 113 |
| 7.31: Experiment Result after 200min.....                      | 114 |
| 7.32: Deep Reactive Ion Etching ARDE Effect.....               | 115 |
| 7.33: Mask Pattern for DRIE.....                               | 115 |
| 7.34: DRIE Simulation Result (a) top view, (b) front view..... | 116 |
| 7.35: Released Part.....                                       | 117 |
| 7.36: Cantilever Beam.....                                     | 118 |
| 7.37: Silicon Nitride Deposition.....                          | 118 |
| 7.38: Silicon Nitride Deposition Top View.....                 | 119 |
| 7.39: Spoke Pattern etched by EDP.....                         | 120 |
| 7.40: Spoke Pattern etched by KOH.....                         | 120 |

## CHAPTER 1

### INTRODUCTION

Commercial CAD tools for micro electro mechanical systems (MEMS) have significantly contributed to the growth that the MEMS industry has experienced over the past two decades by reducing development cycles and enabling the more rapid release of advanced MEMS products. Unfortunately, the CAD community serving for MEMS industry has focused primarily on device performance (for example, mechanical response due to electrostatic loading), with an emphasis on testing and optimizing the performance in a workstation environment. Device manufacturability issues have been long neglected and considered secondary design issues.

Many useful (MEMS) are now being built using silicon etching technologies. Proposals for MEMS computer aided design (CAD) tools have been made in recent years. Hence, considerable work has been done to establish the best architecture for such a system. While significant advancements have been observed in other parts of CAD systems, there remains a need for an improved etch simulator. The fundamental problem is how to model the complex transformation from an initial two-dimensional input mask to the final three-dimensional output shape, particularly when highly anisotropic etchants are used.

This thesis presents the development of a new process simulation program named **MEMSEAGLE** based on Cellular Automata Method. The basic approach is to divide a wafer of silicon into small cells, where each one is given a few primitive rules dictating its rate of removal when exposed to an etchant. If these few simple rules are properly written, then the aggregate behavior of all the cells will accurately represent the complex geometry of a silicon wafer being etched. Finite element analysis (FEA), computational fluid dynamics (CFD), and other

methods are all based upon this approach. This technique permits any etchant to be simulated, by suitable choice of rules for each cell. It also can easily and accurately model complex interactions between etched shapes, such as when one etched region intersects with another, or when an etched shape intersects itself.

When silicon is etched with anisotropic etchants, the resulting shape changes as a function of time. A number of different approaches exist to accurately predict the final shape given an initial mask. The robust Cellular Automata model presented here predicts the three dimensional etched shape as a function of time for any etchant and arbitrary initial mask shape. The model can simulate very complicated geometries and has moderate computational complexity.

The software modeled, MemsEagle, simulates not only the etching processes for bulk micromachining, but also additive processes like doping and deposition. Thus, equipped with a mask editor, MemsEagle has a significant potential of becoming an integrated tool for simulation of micro-fabrication processes including bulk and surface micromachining.

## **1.1. Organization of Thesis**

The organization of the thesis is as follows: Chapter II summarizes the previous researches done in the field of simulation of micro-fabrication techniques, with an emphasis on the anisotropic etching. Moreover, micro-machining processes and Cellular Automata technique is also explained in detail.

The next three chapters describe the three editors utilized in MemsEagle, namely wafer-, mask-, and process editors. Third chapter explains in the detail the operating principles of the wafer editor. The algorithms used for substrate generation and visualization technique for the wafer are included in this part.

Chapter IV concentrates on the capabilities and mechanisms of the integrated mask editor. Information on the user-friendly editor, and the OpenGL functions

used to implement the mask editor are given. Through a number of samples, the mask design capabilities of MemsEagle are also demonstrated.

Chapter V is dedicated to the details of the process editor, which lies at the heart of MemsEagle software. The features of the editor are first explained, as well as the etching and deposition algorithms used. The screen display features of MemsEagle are also explained briefly in this part.

Chapter VI is dedicated to a step-by-step explanation of the user-interface of MemsEagle by making good use of a sample wet etching process. The outputs of the software is also discussed in this chapter.

The simulation capabilities of MemsEagle are verified in Chapter 7. Through several sample micro-fabrication processes, simulation results are compared to the experimental results, utilizing various mask patterns. All four simulation modes (wet etching, dry etching, doping and additive processes) are studied via at least one case per each technique.

Finally, Chapter VII discusses the key points of this study along with the work to be conducted in the near future to improve the designed software.

## CHAPTER 2

### REVIEW OF THE STATE OF THE ART

#### 2.1. Introduction

The recent growth in the number of MEMS devices fabricated via bulk and surface micromachining techniques has brought the need for efficient software design tools. Although there are a number of software packages for simulating the MEMS devices, the tools for simulating the accompanying manufacturing processes are very limited.

The major problem in the simulation of bulk micromachining process is to predict the etch results of silicon due to the anisotropic behavior of etchants used in the process. Furthermore, the primary method for forming mask shapes that will yield an arbitrary structure etched on silicon wafer in return is still based on adhoc techniques. In practice, such approaches leads to the need of manufacturing several prototypes, which in turn increases the cost as well as development time of a particular MEMS design.

This chapter discusses the common micro-fabrication techniques. Recent researches on simulation of these processes and commercial codes are summarized and information about the possible ways of simulation is given. Finally, the technique chosen for simulating the silicon wafer behavior, “cellular automata”, is to be discussed.

## **2.2. Micro-Fabrication Techniques**

The material used for micromachining is mainly silicon. There has been activity in silicon-based micromachining since the early 1960's, when the integrated circuit (IC) technology was developed. The main portion of the researches done were concentrated on anisotropic single-crystalline silicon etching during the 1960's and 1970's. This technology demonstrated simple structures, with initial commercial products being "pressure transducers" [1]. With the beginning 1980's, improvements in thin-film deposition and increased understanding of the micromechanical properties of such films allowed thin-film microstructures to be formed by selective sacrificial etching. Some integration with metal-oxide-semiconductor (MOS) electronics was achieved during this period. Toward the latter half of the 1980's, researchers had demonstrated micro mechanisms and electrostatic micro motors based on polycrystalline surface micromachining. Then, beginning in the 1990's, a significant increase of government research capital had made it possible to have fully integrated complex (MEMS) where sensors, actuators, and control functions are co-fabricated in silicon using micromachining and IC processing.

### **2.2.1. Photolithography**

The process of printing the given two-dimensional pattern onto a thin film layer is called Photolithography. The basic photolithographic process includes a drawing, which defines transparent and opaque areas on a mask. The material used for the mask is a glass plate (soda lime or quartz glass). The resultant mask can be obtained by directly writing on the glass plate or can be drawn much larger and reduced by photolithographic means as illustrated in Fig. 2.1[2].

Ultra violet light is used for transferring the mask pattern so the minimum feature size is restricted by the wavelength of the light. The steps for transferring the pattern from the mask to the substrate are [3]:

- Resist Spinning
- Pre-bake (depending on the resist, typically 10 minutes at 90°)
- Illumination in a mask aligner. The mask aligner enables the precise alignment of the mask pattern to the substrate and the crystal orientation of the silicon wafer.
- Post-bake (depending on the resist, typically 20 minutes at 120°)
- Development

For aligning the mask, three techniques can be used: contact printing, proximity printing and projection printing. Proximity printing uses the shadow of the opaque regions of the mask. In contact printing, because the mask touches the wafer, there is possibility that the mask be damaged. Projection printing is the most expensive but best solution. Minimum line width of the mask is close to the wavelength of the light (whereas in proximity printing for a wavelength of 400nm, the line width should be at least 1 $\mu$ m).

There are two types of photoresist used in mask design. Positive resists become soluble after illumination whereas the negative ones become insoluble. It is harder to work with negative photoresists due to competing chemical reaction of the material with ambient air and poor adhesion.

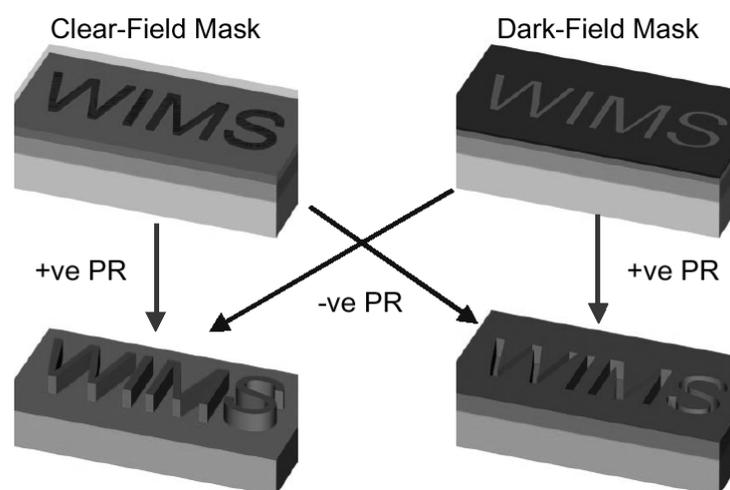


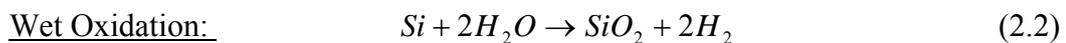
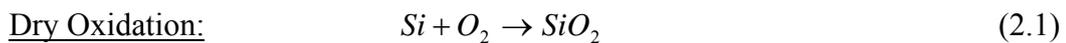
Figure 2.1: Photolithography [2]

## 2.2.2. Thin Film Deposition and Doping

The additive processes mainly used in surface fabrication techniques are discussed and information about recent research was given below. The additive processes are oxidation, chemical and physical vapor depositions and doping.

### 2.2.2.1. Oxidation

Silicon Dioxide ( $\text{SiO}_2$ ) is usually used as mask material for etching processes. Another important property of  $\text{SiO}_2$  is its dielectric behavior. The  $\text{SiO}_2$  growth rate is strongly dependent on the temperature [4]. The oxidation process typically performed in furnaces with temperatures of 900-1150°C. The growth rate of wet and dry oxidation is different. In wet oxidation process, steam is added to the oxygen, which considerably increases the growth rate of  $\text{SiO}_2$ .



### 2.2.2.2. Chemical Vapor Deposition (CVD)

Polysilicon, silicon nitride and phosphor silicate glass are deposited using this technique. Phosphor silicate glass is often used as sacrificial layer in surface micromachining processes.

A pipe, which contains silicon wafers, is fed with the gas form of the materials to be deposited at elevated temperatures. There are three types of CVD: Atmospheric Pressure CVD (APCVD), Low Pressure CVD (LPCVD) and Plasma Enhanced CVD (PECVD) [5]. In LPCVD, the integration of atoms in the surface determines the growth rate of the film and it provides better uniformity and reproducibility. The deposition temperatures are on the levels of 700-900°C. Residual stress (tension) occurs in films that are deposited in amorphous state

and they resulting structure eventually recrystallize later. However, these stresses in the films can be relieved by an annealing process.

### **2.2.2.3. Physical Vapor Deposition (PVD)**

In this process, the material to be deposited (with sufficiently high vapor pressure  $\approx 10\text{mTorr}$ ) is placed in a vacuum chamber and evaporated by using different techniques like resistive heating. Evaporation, sputtering and ion beam deposition are the available PVD techniques.

In evaporation, the material to be deposited is heated until it evaporates and the molecules land on the wafer. The heating can be done by resistive heating, **radio frequency** (RF) heating, laser ablation or electron beam heating. **Electron beam** (E-Beam) heating has certain advantages like less contamination, a better process control and more efficient heat transfer. In E-beam heating, high temperatures can be achieved which makes most materials to be deposited using this method. Evaporation method does not have good step coverage.

In deposition processes using sputtering, high-energy ions hit the substrate and sputter material from the target. Usually Argon is used for creating the plasma, the Ar ions hit the material to be deposited and knock off atoms. The main advantage of the sputtering method is practically all the materials can be deposited using this method. In addition, the film obtained is more homogenous.

The ion-beam deposition technique uses an ion beam to bombard the source to create the atoms to be deposited. Through an arc discharge in a pressure range of  $1\text{-}100\mu\text{Torr}$ , with voltages of  $500\text{-}1000\text{V}$ , the ion beam is generated. This method could also be used for etching by directing the ion beam to the wafer. The most important property of this method is, it has good cleanliness and control [2].

#### 2.2.2.4. Doping

Doping is the introduction of impurities into the silicon wafer to alter its electrical, electrochemical, chemical and mechanical states. The dopants used can be n-type (Phosphorus (P), Arsenic (As), Antimony (Sb)) or p-type (Boron (B), Aluminum (Al)). Wet etch rate of the silicon depend on the voltage difference between the silicon wafer and the etchant solution, and this is directly dependent on the type and the concentration of the dopant. Boron doping can be used as an etch stop in etching processes.

In general, diffusion and ion-implantation are used for doping. In diffusion method, the wafer is placed into the furnaces and a carrier gas is flown through the furnace. If the source is in solid form, the dopant wafers are also placed into the furnace next to silicon wafers. The sublimated atoms diffuse into the silicon wafer. In this process oxide is also formed. The other alternative is to use a liquid source, which allows the carrier gas passes through. The diffusion process is carried on in two steps: pre-deposition and drive-in. After a highly doped region is formed in the pre-deposition step, using this region the impurities are forced into deeper regions in the drive-in step. The temperatures used in Doping are 800-1200°C. The total impurity dose ( $Q$ ) is calculated by using (2.1).

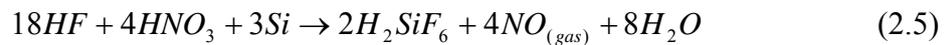
$$Q = 1.13N_0\sqrt{D_1t_1} \quad (2.3)$$

$$N(x,t) = \left[ \frac{Q}{\sqrt{\pi Dt}} \right] e^{-\frac{x^2}{4Dt}} \quad (2.4)$$

where  $D_1$  and  $t_1$  are the pre-deposition diffusion coefficient and time [6]. Notice that (2.4) expresses the concentration profile after the drive-in process. If the wafers are not to be exposed to high temperatures, ion implantation method could be used for doping. First, impurities are introduced to the wafer using a high-energy beam of the ions to be implanted. Then, via an annealing process, the atoms penetrate through the wafer. Shallow junctions, which cannot be implemented by diffusion process, can be obtained.

#### 2.2.4. Wet Chemical Etching

The oldest micromachining process used is wet chemical etching. Wet etching processes are divided into two categories: isotropic and anisotropic. In isotropic etching the etch rate is not dependent on direction and mask orientation. The most common isotropic silicon etchant is “HNA”, a mixture of HF (Hydrofluoric acid), HNO<sub>3</sub> (Nitric acid), and CH<sub>3</sub>COOH (Acetic acid). The reaction between HNA and Silicon is [7]:



Doping can be used as an etch stop technique for HNA, because the etch rate of HNA is nearly 150 times slower in lightly doped ( $<10^{17} \text{ cm}^{-3}$  n or p type) regions than the heavily doped ones.

Etchants erode the silicon wafer at different rates in miscellaneous directions in anisotropic wet etching. Most anisotropic etchants slow down at the (111) planes. The dominant planes in anisotropic etching [8] are (100), (110) and (111) as shown in Fig. 2.2.

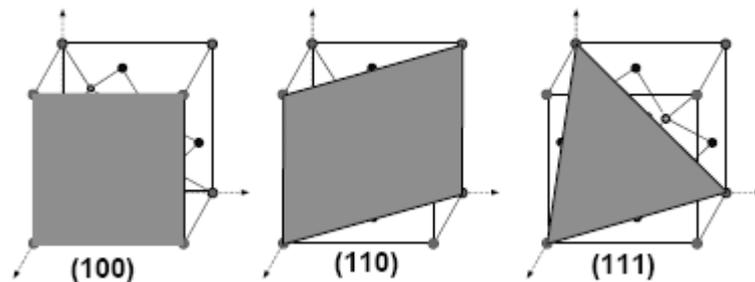


Figure 2.2: Miller Indices of various planes

The slowest etch planes are exposed during the etch processes and the etching tends to stop at these planes, in most cases (111) plane as shown in Figs 2.3. and 2.4. Another important property of anisotropic etching is the termination of etching at concave corners and undercutting of convex corners in (100) wafers as shown in Fig. 2.5.

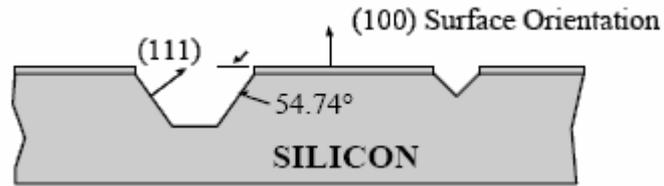


Figure 2.3: Anisotropic Wet Etching (100) plane [4].

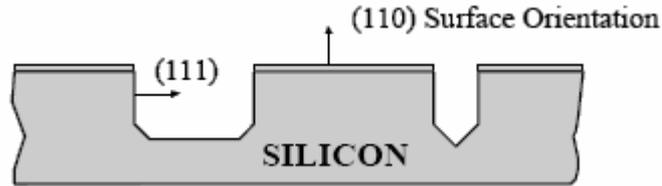


Figure 2.4: Anisotropic Wet Etching (110) plane [4]

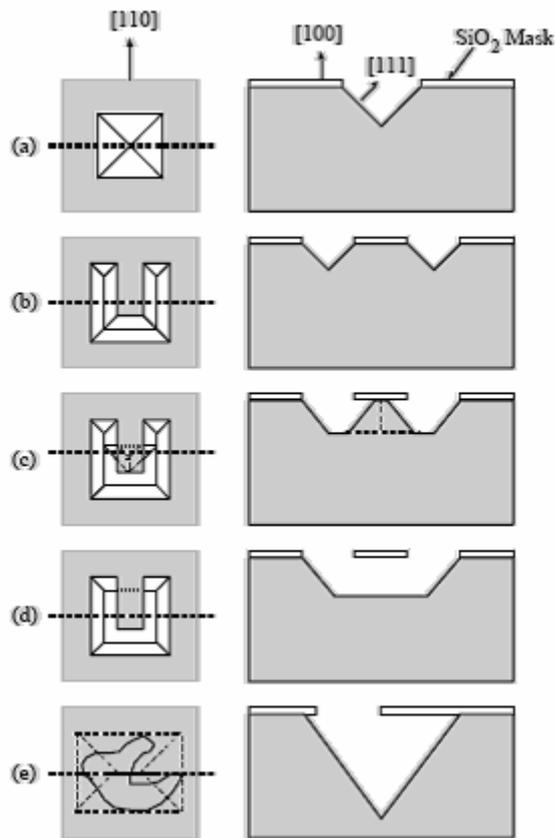
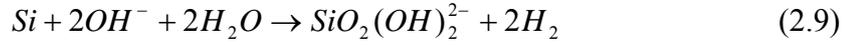
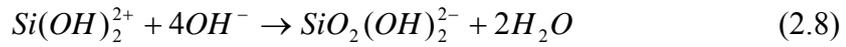
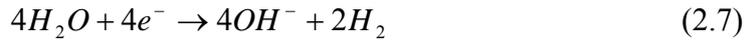
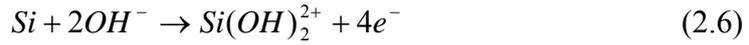


Figure 2.5: Convex Corner Undercutting [4]

Hydroxides of alkali metals KOH, NaOH, CeOH and RbOH are being used as orientation dependent etchants in micro-fabrication processes. For such etchants, the following reactions take place between the silicon wafer and etchant [9]:



Different KOH concentrations and temperatures considerably changes the etch rate. However, the Alkali hydroxide etchants have good selectivity for (100) plane as seen on Table 2.1. These etchants can also be selective to doping concentration [11].

Table 2.1: KOH Formulations [10]

| <b>Formulation</b>                         | <b>Temp<br/>°C</b> | <b>Etch Rate<br/>(<math>\mu\text{m}/\text{min}</math>)</b> | <b>(100)/(111)<br/>Etch Ratio</b> | <b>Masking Films<br/>(etch Rate)</b>  |
|--|--------------------|--|-----------------------------------|---|
| KOH (44g)<br>Water, Isopropanol<br>(100ml) | 85                 | 1.4  | 400:1                             | SiO <sub>2</sub> (1.4nm/min)<br>Si <sub>3</sub> N <sub>4</sub> (negligible) |
| KOH (44g)<br>Water, Isopropanol<br>(100ml) | 50                 | 1.0  | 400:1                             | approx. As above  |
| KOH (44g)<br>Water, Isopropanol<br>(100ml) | 65                 | 0.25 to 1.0  | -                                 | SiO <sub>2</sub> (1.4nm/min)<br>Si <sub>3</sub> N <sub>4</sub> (negligible) |

Another common etchant used is “Tetra methyl Ammonium Hydroxide” (TMAH, (CH<sub>3</sub>)<sub>4</sub>NOH). It is considerably cheaper, can be modified to avoid etching aluminum, and may have concentration etch stops [12]. Table 2.2 outlines the lower plane selectivity of TMAH. Hence, the surfaces created by TMAH are not as smooth as the ones obtained through EDP or alkali hydroxide etchants.

Table 2.2: Selectivity of TMAH etchants for dielectrics versus (100) Silicon [13]

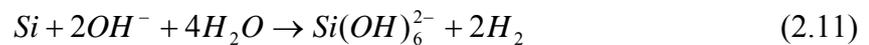
| Dielectric                  | Selectivity 4 wt% TMAH, 80°C | Selectivity (Si-doped, 13.5g/l), 4 wt% TMAH, 80°C | Selectivity 20 wt% TMAH, 95°C                                    |
|-----------------------------|------------------------------|---|--|
| Thermal Silicon Oxide       | 5.3x10 <sup>3</sup>          | 34.7x10 <sup>3</sup>                              | 5.2x10 <sup>3</sup>  |
| Low-Temperature Oxide (LTO) | 1.3x10 <sup>3</sup>          | 4.2x10 <sup>3</sup>                               | 2.8x10 <sup>3</sup> (360° LTO)<br>3.4x10 <sup>3</sup> (360° LTO) |
| PECVD Oxide                 | 1.4x10 <sup>3</sup>          | 4.3x10 <sup>3</sup>                               | no value given   |
| LPCVD Silicon Nitride       | 24.4x10 <sup>3</sup>         | 49.3x10 <sup>3</sup>                              | 38x10 <sup>3</sup>   |
| PECVD Silicon Nitride       | 9.2x10 <sup>3</sup>          | 18.5x10 <sup>3</sup>                              | 3.6x10 <sup>3</sup>  |

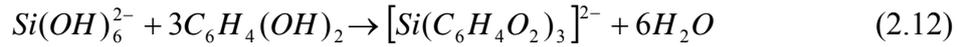
Ethylene Diamine Pyrocatechol (EDP) is one of the most common anisotropic etchant used [13]. The selectivity between (100) and (111) planes are on the magnitudes of 35, but have greater selectivity to doping concentration. Most common formulations of EDP are shown on Table 2.3.

Table 2.3: Common EDP Formulations [10]

| Formulation  | Temp °C | Etch Rate (µm/min) | (100)/(111) Etch Ratio | Masking Films (etch Rate)   |
|--|---------|--------------------|------------------------|---|
| Ethylene diamine (750ml) Pyrocatechol (120g) Water (100ml) | 115     | 0.75               | 35:1                   | SiO <sub>2</sub> (0.2nm/min)<br>Si <sub>3</sub> N <sub>4</sub> (0.1nm/min)<br>Au, Cr, Ag, Cu, Ta (negligible) |
| Ethylene diamine (750ml) Pyrocatechol (120g) Water (240ml) | 115     | 1.25               | 35:1                   | SiO <sub>2</sub> (0.2nm/min)<br>Si <sub>3</sub> N <sub>4</sub> (0.1nm/min)<br>Au, Cr, Ag, Cu, Ta (negligible) |

The chemical reactions that take place between the silicon wafer and the EDP are:





### 2.2.5. Dry Etching (Plasma Etching)

In plasma etching, ions impinge on the substrate, and neutral particles arrive to the substrate by diffusion. The etchant gases and the wafers are in a chamber, and the gases are ionized by RF glow discharge. The etching temperatures are on the levels of 150°C to 200°C and in some cases room temperature. The main disadvantage of dry etching is the worse selectivity it has.

The silicon wafer is etched by the etch gas (that involves F) with the following chemical reactions [14]:



There are three common types of dry etching. Plasma etching is an isotropic chemical etch process. The wafer is grounded in plasma etching.

In reactive ion etching, the wafer is placed on an electrode that is driven by RF signal. The etching is anisotropic in nature, but anisotropic behavior is not the result of crystallographic properties but a result of the direction of the ion flux towards the substrate. The etch process is both chemical and physical. In Deep Reactive Ion Etch process, because of the special gases that form a polymer on sidewalls, tall and narrow holes can be drilled onto the wafer.

Ion Beam Etching process is a physical etching process. Argon ions are used to bombard the wafer surface and the selectivity of the process is very low due to the physical behavior of the etch mechanism. If a reactive gas is used, the process is called Reactive Ion Beam Etching, where the etch process is both physical and chemical in nature.

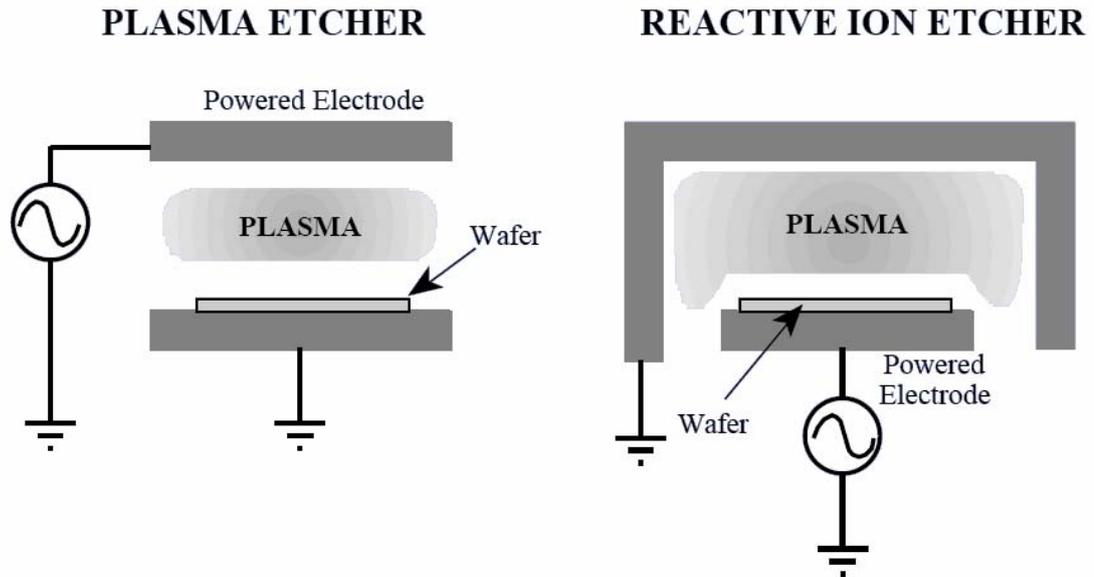


Figure 2.6: Plasma & Reactive Ion Etching [4]

### 2.2.6. Micromachining Techniques

There are two main micromachining technology used in the fabrication of MEMS devices: Bulk Micromachining and Surface Micromachining. There are also other techniques like LIGA and Electroplating. However, the processes simulated in MemsEagle are mainly used for bulk micromachining. The additive processes, which are used frequently in surface micromachining, are just added to the software to have flexibility for bulk micromachining processes.

Surface Micromachining is characterized by the fabrication of micromechanical structures from deposited thin films [1]. In such processes, the substrate is used as a base to build the structure upon it. Although a wide variety of materials could be utilized in this process, the technology has evolved over the years to use the silicon dioxide as the sacrificial material and Polysilicon as the structural material.

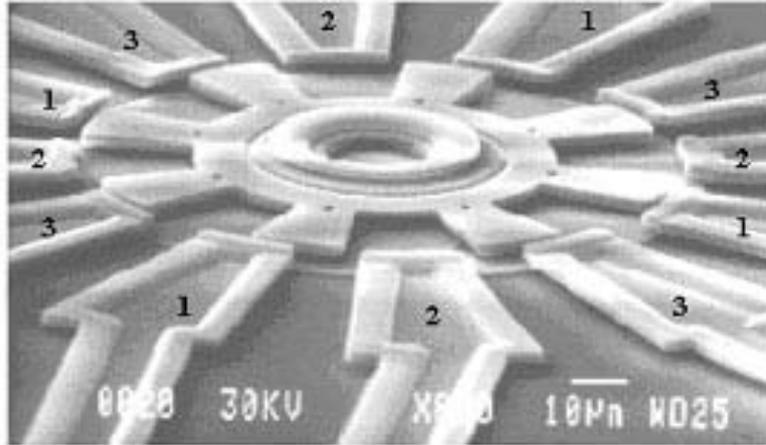


Figure 2.7: Surface Micromachining Example [4].

Subtractive processes involving removal of appreciable regions of the substrate (usually silicon, but possibly glass, organics, metals, etc.) is called Bulk Micromachining. The purpose of bulk micromachining is to selectively remove significant amounts of silicon from a substrate. This versatile process is used to perform a wide variety of tasks:

- to “undercut” moving structures;
- to form membranes on one side of a wafer;
- to make a variety of trenches, holes, or other structures.

Fig. 2.8 shows the difference between surface and bulk micromachining. The third technology is called LIGA (Lithographie, Galvanoformung, Abformung) and it is used to expose thick layers by using X-Rays.

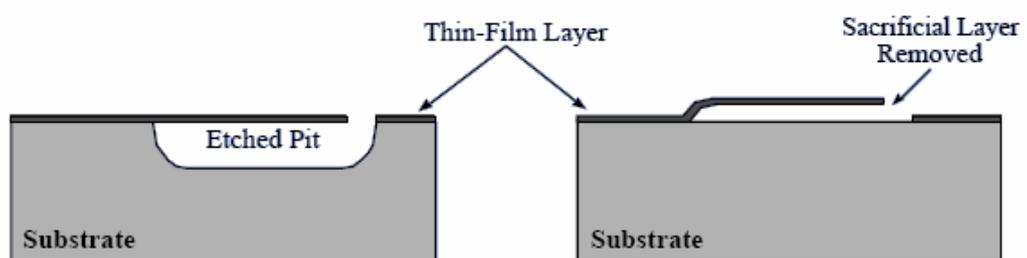


Figure 2.8: Bulk & Surface Micromachining [4]

### **2.3. Simulation Programs**

Early works on the area of software simulation tools for MEMS fabrication have concentrated on simulating the Anisotropic Etching of silicon, for different etchants and for different silicon crystal orientations. The recent developments in the computer industry now enable the users enjoy increased computational power with reduced hardware costs. As a result, simulating relatively complicated etch processes on personal computers with modest resources has become feasible.

Two techniques are mainly used to simulate the anisotropic etch processes: Geometric Models and Cellular Automata Models. In geometric models, the substrate is considered as a continuous entity and the result of the etch process is found by using geometric rules. The major disadvantage of geometric simulation models is that they do not efficiently take into account the merging shapes. In cellular automata method, the substrate is taken as discrete cells of silicon atoms and the etch process is simulated by taking into account the bonds of an atom with its neighbors. With this method, complex shapes and merging planes can be simulated with high accuracy and efficiency. However, the main disadvantage of this method is the requirement for high computational power, with the development of faster computers; this is no longer a tiebreaker between the two competing approaches.

There are a number of software programs reported for the simulation of manufacturing processes. The simulation principles of these programs are briefly explained below.

#### **2.3.1. Anisotropic Silicon Etching Program (ASEP)**

ASEP (Anisotropic Silicon Etching Program) uses traveling planes to predict the final shape of the substrate [15]. The planes move in accordance with the pre-defined etch rates and complex shapes can be simulated. ASEP can simulate the etching of <100> oriented silicon wafers in aqueous solutions of KOH. ASEP is

based on the concept that a few lattice planes are dominant in the anisotropic etching of the silicon wafers. In KOH, the most important ones are the  $\langle 111 \rangle$  planes, which have the slowest etch rate that determine the concave corners. The  $\langle 311 \rangle$  planes that have the fastest etch rates and determine the convex corners. Whereas the  $\langle 100 \rangle$  planes have intermediate etch rates. Input from a user specified mask file is identified as crystal directions. The program decides whether the inside or the outside of the mask shape will be exposed to etchant according to the rotation type of the polygon. (CW or CCW). Special problems occur at the convex corners, where the fast etching planes are revealed [16]. Eight types of corners are described by the code. The program distinguishes the convex and concave corners by using the determinant formed by the first two Miller indices of the neighboring planes. The angles are determined by using the cosine of the angle between the two neighboring planes. When a corner is overetched, the program issues a warning message.

While the etch process continues, some planes may disappear and new ones could emerge. The status of each plane is checked by determining the Miller indices of its left and right neighbors, and by establishing whether or not the plane lies in front of the intersection of two neighbors, with respect to the etch opening. ASEP decides on modifying or eliminating the plane after that.

### **2.3.2. SEGS Simulator**

SEGS program is a hybrid, trying to take advantage of both approaches: the accuracy of cellular automata and the speed of geometric models [17]. The model represents the shapes as a large number of small segments (or facets), but they also retain geometrical information. The basic approach is to start with the polygonal boundary of a vector method, then subdivide each straight-line segment into many smaller segments. The program gives visual etch results faster than ordinary cellular automata methods. In SEGS method, first the local intersections are computed and then the global intersections are processed. The local calculations commence with taking two nearest neighbors and checking the

relative validity of adjacent line segments. A segment is valid if it lies in the still un-etched halfplanes of its two neighbors. By eliminating the invalid segments, the test is carried out for all the segments in the shapes.

By hybridizing the cellular automata and geometric approach thus decoupling the local and global interaction calculations, each can be optimized individually.

SEGS reads the etch rate data from a file as three-dimensional vectors each with an etch rate magnitude. The three-dimensional rates are decomposed into two components:

- $r_{2d}$ : A two-dimensional etch rate in the plane of the wafer surface;
- $r_z$ : A lateral etch rate due to depth.

The rate of the plane that moves in the mask surface is  $r_{2d}$ .  $r_z$  is the lateral distance between the top or mask layer edge of a etch facet and the bottom edge of a etch facet. The mask data are first read from a file as a set of polygons. Then the data is divided into an array of N-by-N cells. Each element of array contains the information of location  $x$ ,  $y$  and the local slope with the calculated local tangent and normal vectors. [17] calls these elements as directed line segments.

In this technique, at each time step the  $x$ ,  $y$  positions of each segment in the array are calculated by adding a velocity vector in the direction of the local normal equal to the etch rate  $r_{2d}$  multiplied by the time step. Then each segment is compared to its neighbor. Using dot products of the local normal, local tangent and the vectors from segments, the relative location of the segment is determined [17]. If a segment lies above the half plane defined by the local tangent of segment then it is un-etched. Otherwise, it is removed from the array. This process is repeated for all segments in the array. Next, the comparison is done in the opposite winding direction for all segments. This winding in both directions is repeated until the length of the polygon list stops to decrease. If the length drops to zero, then the polygon is etched away.

At each time step, the x, y (spatial) position is converted into a cell location in the N-by-N grid and the polygon index and list index are then written to the cell array at this location for each line segment. A global intersection occurs when two segments are written to the same cell. The intersecting polygon lists are cut at the intersections and rejoined into a new longer list. The new list is then used as the input for the next time step.

The two dimensional simulation provides the top or mask layer edge of the etch facets. The bottom edge is found by using the new list at the end of each time step and updating the list with  $r_z$  instead of  $r_{2d}$ . The distance between the top and bottom edges of  $r_z$  is increased by a depth  $d$  for each time step. The new list generated is then checked for local and global intersections as before.

The simulation time is tens of seconds for various complex shapes on a typical workstation. Shapes are input via Crystallographic Information File (CIF), Gridded Data Set (GDSII), or a public domain drawing program (xfig). While process (etch rate) information is input via text files, and output is available in several formats including PostScript, and IGES for subsequent 3-D solid modeling and finite element analysis.

Fig. 2.9 illustrates a two-dimensional example of SEGS simulator result, while Fig 2.10. shows a three-dimensional example. The contours show the different etch result of the wafer at different times.

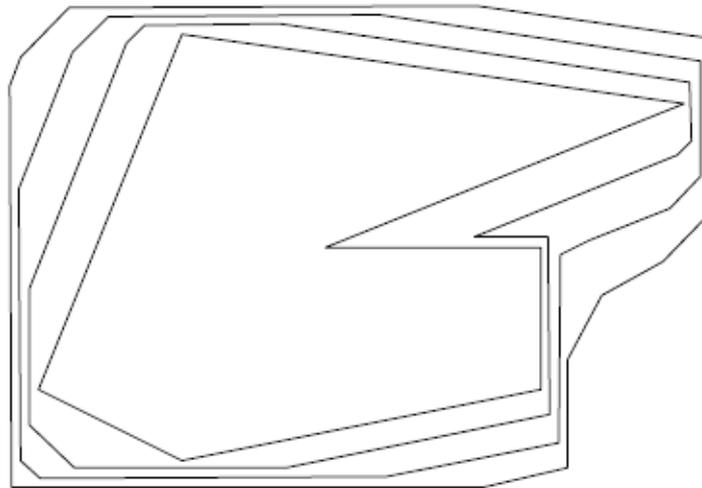


Figure 2.9: SEGS two-dimensional example [17]

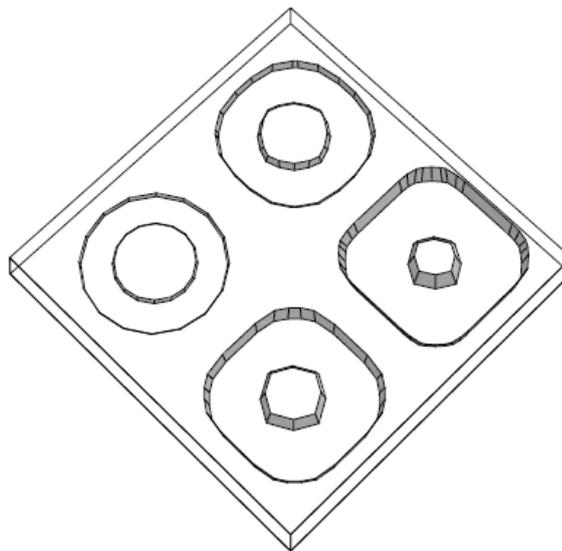


Figure 2.10: SEGS three-dimensional example [17]

The SEGS on-line simulator provides a tool to predict etch process, but has some reported limitations. Because of the polygonization, some input shapes are approximated like circles. In addition, some output shapes are shifted by one segment length due to the discretization. When simulating very complex shapes, invalid results may be generated. This can occur during grazing

intersections of two separate polygons when the calculation results are small and subject to round off error. This may also occur when two closely-spaced parallel lines are simulated, since the calculated dot products are also very small and susceptible to round-off error.

### 2.3.3. Suzana

The Suzana program was introduced by [18]. The program uses cellular automata to find the resultant etch shape. The cells can take two states etched or remained; and the etch status is decided by using the neighboring cell locations and a random number. Two different crystal oriented silicon can be simulated.

The etch rates used by Suzana is dependent on temperature  $T$  and concentration  $c$ , which can be expressed as:

$$R_{(hkl)}(T, c) = R_{0(hkl)}(c) \cdot \exp(-E_{a(hkl)} / kT) \quad (2.15)$$

where  $k$  is the Boltzman constant and the activation energy  $E_a$ , etch rate  $R_0$  depend on the particular crystal plane ( $hkl$ ). Note that  $h$ ,  $k$  and  $l$  are the integers used to reference a particular crystal plane. Using the etch rates the etch probabilities  $P_{hkl}$  are calculated:

$$P_{hkl} = f(R_{(100)}, R_{(110)}, R_{(111)}) \quad (2.16)$$

The probabilities are normalized as:

$$\max(P_{100}, P_{110}, P_{111}) = 1 \quad (2.17)$$

Taking into account (2.15), (2.16), and (2.17) yields the following rules for a cell:

1. An etch front cell will be removed if it has:
  - a. two neighbors and if a random number from the range  $[0,1]$  lies in the interval  $[0, P_{100}]$ , or
  - b. three neighbors, of which at least one is located in the etch front, and if a random number from the range  $[0,1]$  lies in the interval  $[0, P_{110}]$ , or

- c. three neighbors, of which no one is located in the etch front, and if a random number from the range  $[0,1]$  lies in the interval  $[0, P_{111}]$ .
2. All the cells fulfilling none of the rules 1a, 1b and 1c will be removed.

To explain the equilibrium utilizing the theory of crystal growth [18], the second nearest atoms should also be included in the model, which is done implicitly in Suzana.

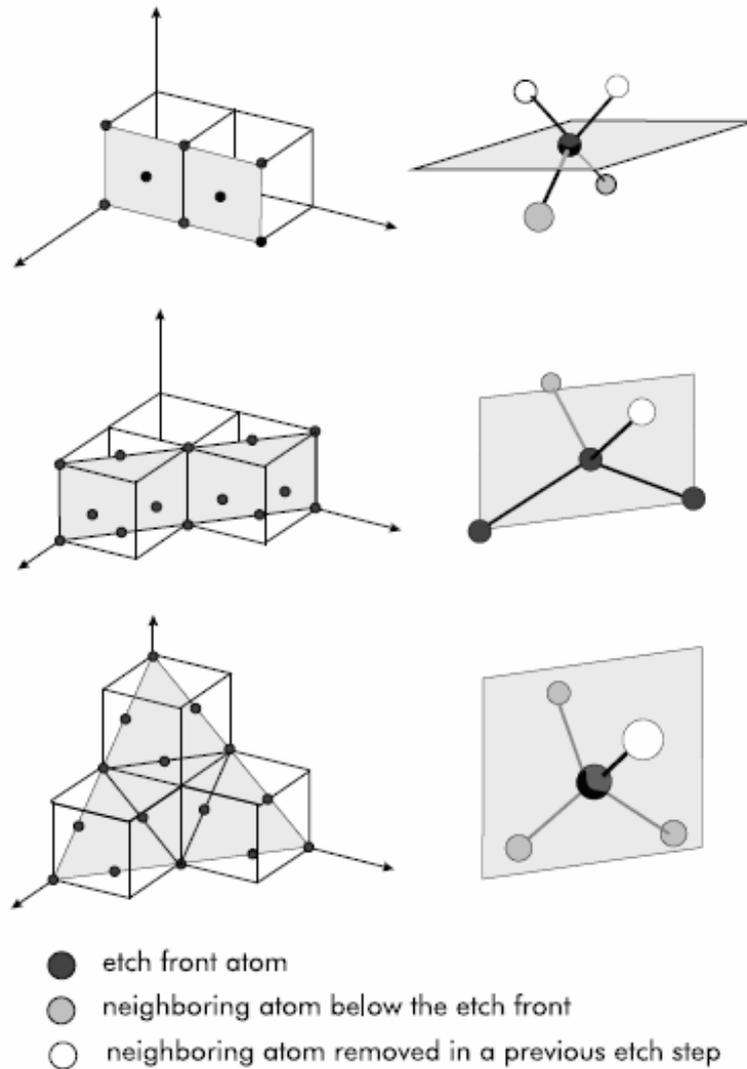


Figure 2.11: Bonding situation of (100), (110) and (111) surface atoms of Suzana[18].

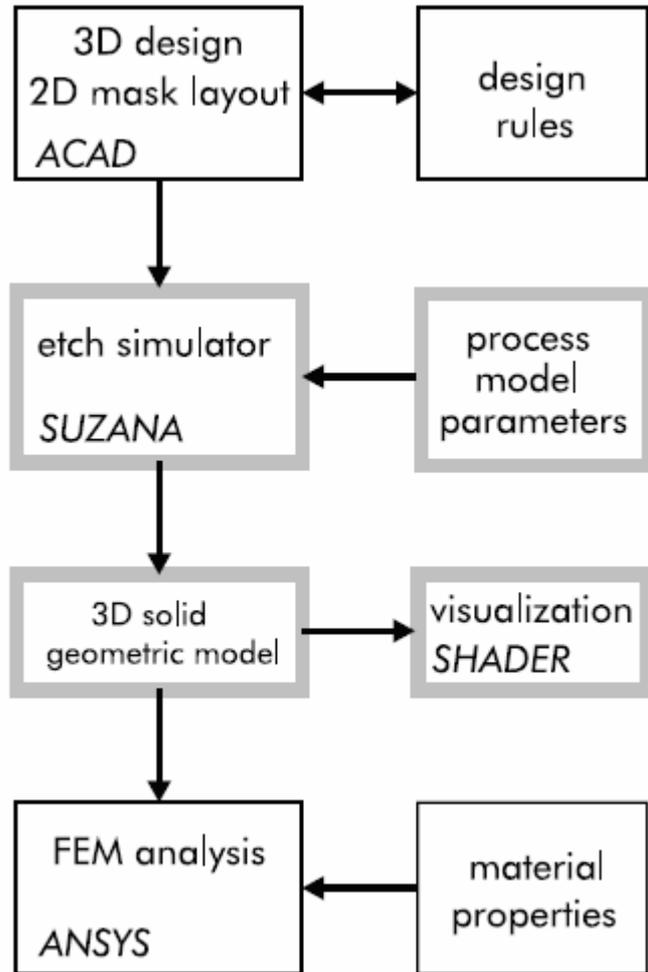


Figure 2.12: Schematic Block Diagram of the CAD Architecture for using Suzana[18]

For visualization, the three dimensional model is translated into a surface model which is imported to Shader. Shader then calculates the pictures of the simulation results and exports the data to display on the screen.

#### 2.3.4. Anisotropic Etch Simulator (AnisE)

AnisE has been developed by Intellisense Software [19]. The method used for simulating the etch process is cellular automata. Figure 2.13 shows the crystal structure used for modeling the silicon wafer in AnisE.

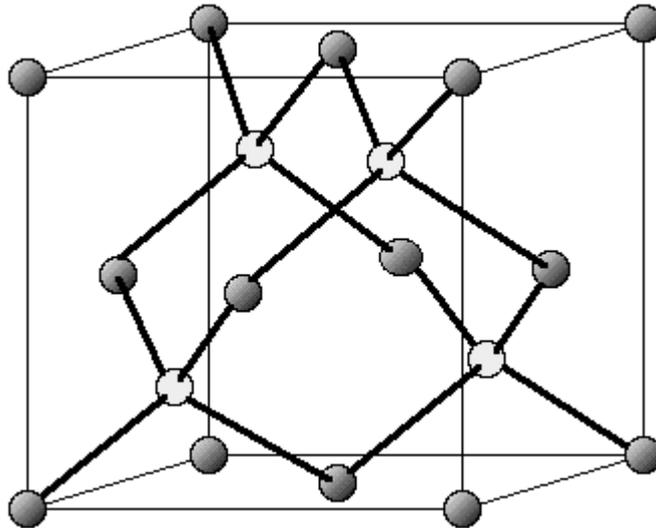


Figure 2.13: Silicon Crystal Structure [19]

The cellular automata model used in AnisE is based on the following rules [19]:

- The lattice structure of the cells: A silicon atom is represented by a cell in the model in terms of its lattice configuration. The size of each cell is approximately  $1\text{mm}^3$  depending on the selected resolution of the model.
- The possible states of a cell: There are two possible states for the crystal atoms, etched or non-etched.
- The effect of neighboring cells: The atom is covalently bonded to four other atoms. The behavior of each atom depends on the interaction of the atom with each four neighbor.

Rules to determine the state of the cell are as follows. The conditions of the neighboring atoms strongly affect the state of the cell. The location of the four surrounding cells and their states are used for defining the state.

In order to remove the cell, which lies on the  $\langle 100 \rangle$  plane, two neighboring cells should be etched as shown on Figure 2.11. That is, in the previous etch step, two neighboring cells were removed. In order to remove the cell from the crystal lattice, two covalent bonds that lie below the etch-front plane must be broken.

When considering the etch front for the  $\langle 110 \rangle$  plane, there are three etch-front atoms. Three covalent bonds, two of which lie on the etch-front plane, must be broken to remove any of these cells from the crystal lattice.

For the  $\langle 111 \rangle$  plane, there is only one etch-front atom. One neighboring cell must be etched in a previous step to expose this etch-front atom. To remove the cells that lie on  $\langle 111 \rangle$  planes three covalent bonds should be broken and all of them lie under the etch-front plane. This makes the removal of these cells very slow.

From the above orientations of the cells with respect to the planar etch front, the probability is calculated whether the cell will be removed. This probability takes into account the experimental silicon etch rates for the  $\langle 100 \rangle$ ,  $\langle 110 \rangle$ , and  $\langle 111 \rangle$  planes. A higher etch rate for a given plane will increase the probability that the etch-front atom in that plane will be removed [20].

The model used in AnisE implicitly considers the second order effects resulting from the second nearest neighbors. Model takes into account the number of neighboring cells and their location with respect to the etch front. As a result, the simulation predicts the appearance of higher order etch planes.

There is a simple mask editor embedded inside AnisE but users can import masks in DXF or GDS II formats. In order start the simulation, the user needs to enter the etchant type (KOH, TMAH, EDP), etchant temperature, concentration, wafer orientation ( $\langle 100 \rangle$  or  $\langle 110 \rangle$ ) and etch time. The result of a simulation done by AnisE, on which the etch stops and double-sided etching can be clearly seen is illustrated on Fig 2.15 [21].

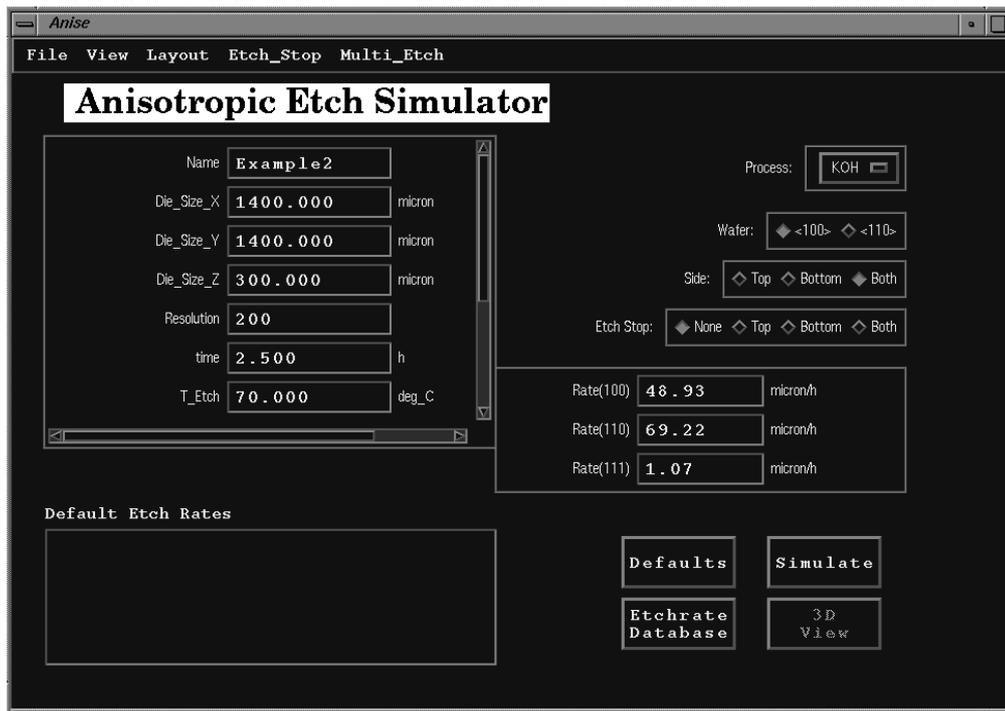


Figure 2.14: AnisE User Interface [19]

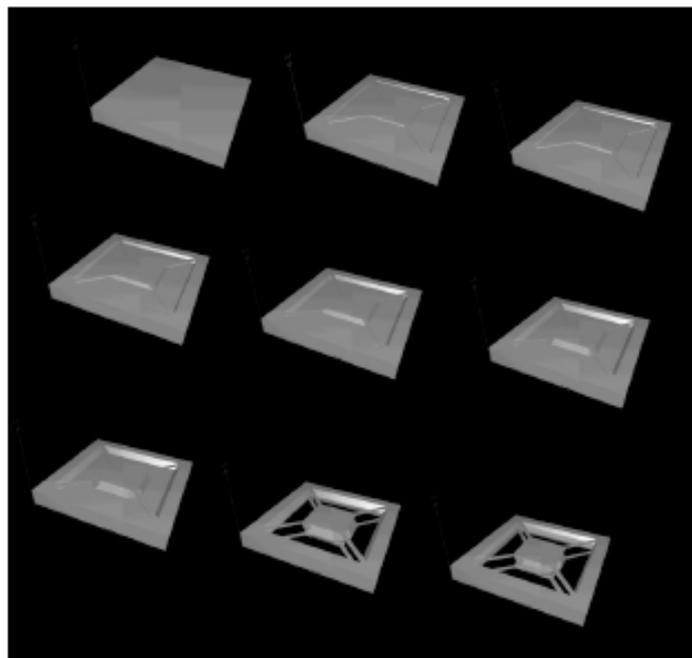


Figure 2.15: AnisE simulation result [19]

### 2.3.5. Anisotropic Crystalline Etching Simulation (ACES)

ACES is an anisotropic etch simulator based on cellular automata method. The software uses a continuous cellular automata method in which each cell can take non-discrete state variables. A cell can take values between 0 and 1 for representing its mass corresponding to its extent of removal.

Assuming that the desired etch rate on a particular crystal plane is  $E_s \in [0,1]$ , and the elapsed time of each etch step is  $T$  (with the default value being one); the number of etch steps ( $N_T$ ) that are required to completely remove a cell equals [22]:

$$N_T = \left\lceil \frac{M}{E_s T} \right\rceil \quad (2.18)$$

If multiplication of  $E_s$  and  $T$  is exactly equal to  $M$  or  $M$  is a multiple of  $E_s T$ , then the effective etch rate  $E_s'$  is equal to the desired etch rate:

$$E_s' = \frac{M}{N_T T} = \frac{M}{\left( \frac{M}{E_s T} \right) T} = E_s \quad (2.19)$$

Whereas in cases when  $M$  is not a multiple of  $E_s T$ , the value of the effective etch rate differs from the desired one:

$$E_s' = \frac{M}{N_T T} = \frac{M}{\left( \frac{M}{E_s T} \right) T} \neq E_s \quad (2.20)$$

When this situation occurs the neighbors of the cell being etched should also be exposed to etchant before the next time step. A time compensation factor is used to achieve this effect. When a cell is removed during an etch-step  $k$ , the etching of the next cell will not begin immediately until the beginning of the next etch-step  $k+1$ . The time balance of the etch-step  $Tb_k$  will be compensated in the step for etching of the next cell. Thus, the time of a specific etch-step  $k$ ,  $T_k$  is not always equal to  $T$ ; rather, based on the  $M$ -value of a cell in step  $k$  ( $M_k$ ), the compensation can be computed by the following equations using the initial conditions  $Tb_0=0$  and  $M=0$ .

If  $M_k \geq E_s(T + Tb_{k-1})$ , then

$$T_k = T + Tb_{k-1} \quad (2.21)$$

$$M_{k+1} = M_k - E_s T_k \quad (2.22)$$

$$Tb_k = 0 \quad (2.23)$$

otherwise:

$$T_k = \frac{M_k}{E_s} \quad (2.24)$$

$$M_{k+1} = 0 \quad (2.25)$$

$$Tb_k = T - T_k \quad (2.26)$$

Introducing this time compensation factor equals  $E_s T_k$  and  $M_k$  for every step  $k$ . The program also uses dynamic cellular automata algorithm in which only the cells on the etchant-wafer surface is taken into computations. This is significantly increasing the speed.

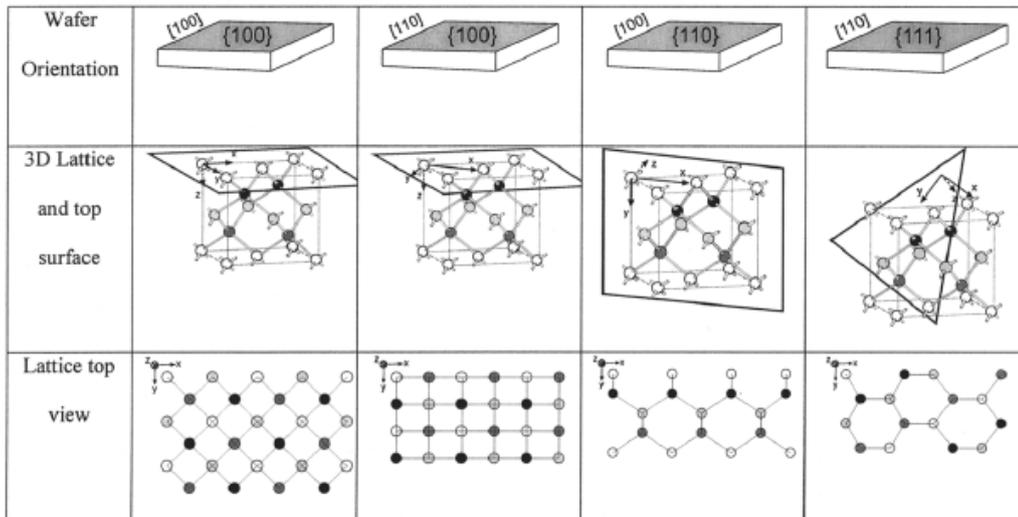


Figure 2.16: Different crystal types used in ACES [23]

|                               |         |          |         |          |         |          |         |          |
|-------------------------------|---------|----------|---------|----------|---------|----------|---------|----------|
| Initial<br>virtual<br>surface |         |          |         |          |         |          |         |          |
|                               | 3D view | Top view | 3D view | Top view | 3D view | Top view | 3D view | Top view |
| Link-set<br>type I<br>        |         |          |         |          |         |          |         |          |
| Link-set<br>type II<br>       |         |          |         |          |         |          |         |          |

Figure 2.17: Top views and link set types of the different crystals used in ACES [23]

Four different crystal types can be etched in ACES as shown in Fig. 2.16 and Fig. 2.17. There are two types of cells in the crystal wafer with different link set types. Cells with different set types are interconnected, in other words the neighbors of a cell must of opposite link set type. Based on these properties the rules to initialize the virtual surfaces are developed as:

- (E1) A virtual surface is started in a horizontal plane. Active cells' locations and link-set types are set based on the orientation of the lattice.
- (E2) When an active cell (A1) is to be etched away based on CA rules, its neighbors will be added to the virtual surface if they are not in the surface. Positions of neighboring cells are calculated from A1's position and its link-set types.
- (E3) A newly added active cell's link-set type is the opposite of that of its neighbors.

The first two rules are related to the wafer crystal type. The information gathered by the two-dimensional mask patterns are also used for initializing the etch surface. The visualization is done by assigning different colors to atoms based on their orientation or depth. Apart from the deep reactive ion etching and doping, the software can simulate the anisotropic and isotropic etching.

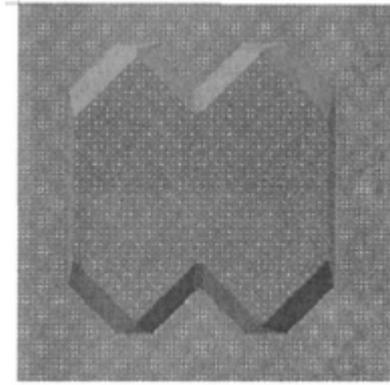


Figure 2.18: Mask pattern to ACES      Figure 2.19: Simulation result of ACES

### 2.3.6. SIMODE

The program SIMODE is a tool for simulation of the orientation dependent etching on monocrystalline materials. The software focused on the anisotropic etch process, based on a graphic data file describing the etch mask. The time needed for the simulation is reduced by simplifying the three-dimensional problem of calculating the etch relief. First, only the two relevant contours (upper and lower edge) are determined in a two-dimensional calculation which describe the etch relief significantly. Fig 2.20 demonstrates the simple etch trench of SIMODE.

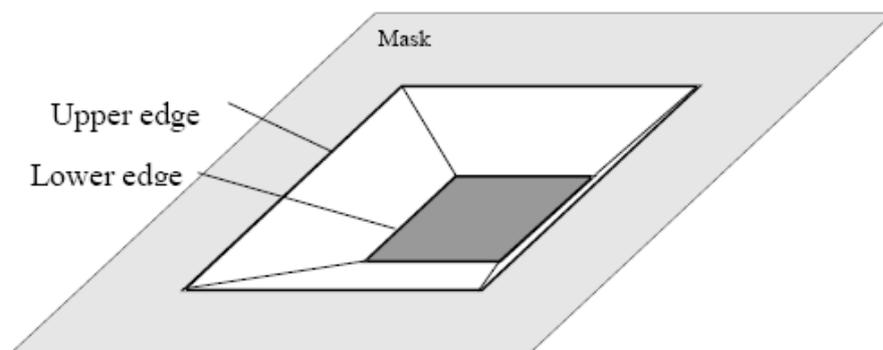


Figure 2.20: Simple Etch Trench of SIMODE[24]

The program can simulate processes with different temperatures, concentrations and etchants. The masks are defined with closed polygons. There should be no intersections within the polygons. In order to make a distinction between

windows (concave polygons) and masked areas (convex polygons), the revolution of the polygon lines is used as illustrated in Fig. 2.21. A polygon line with a clockwise revolution (mathematically negative) is a concave polygon; however, a polygon line with a counter-clockwise revolution (mathematically positive) is a convex polygon.

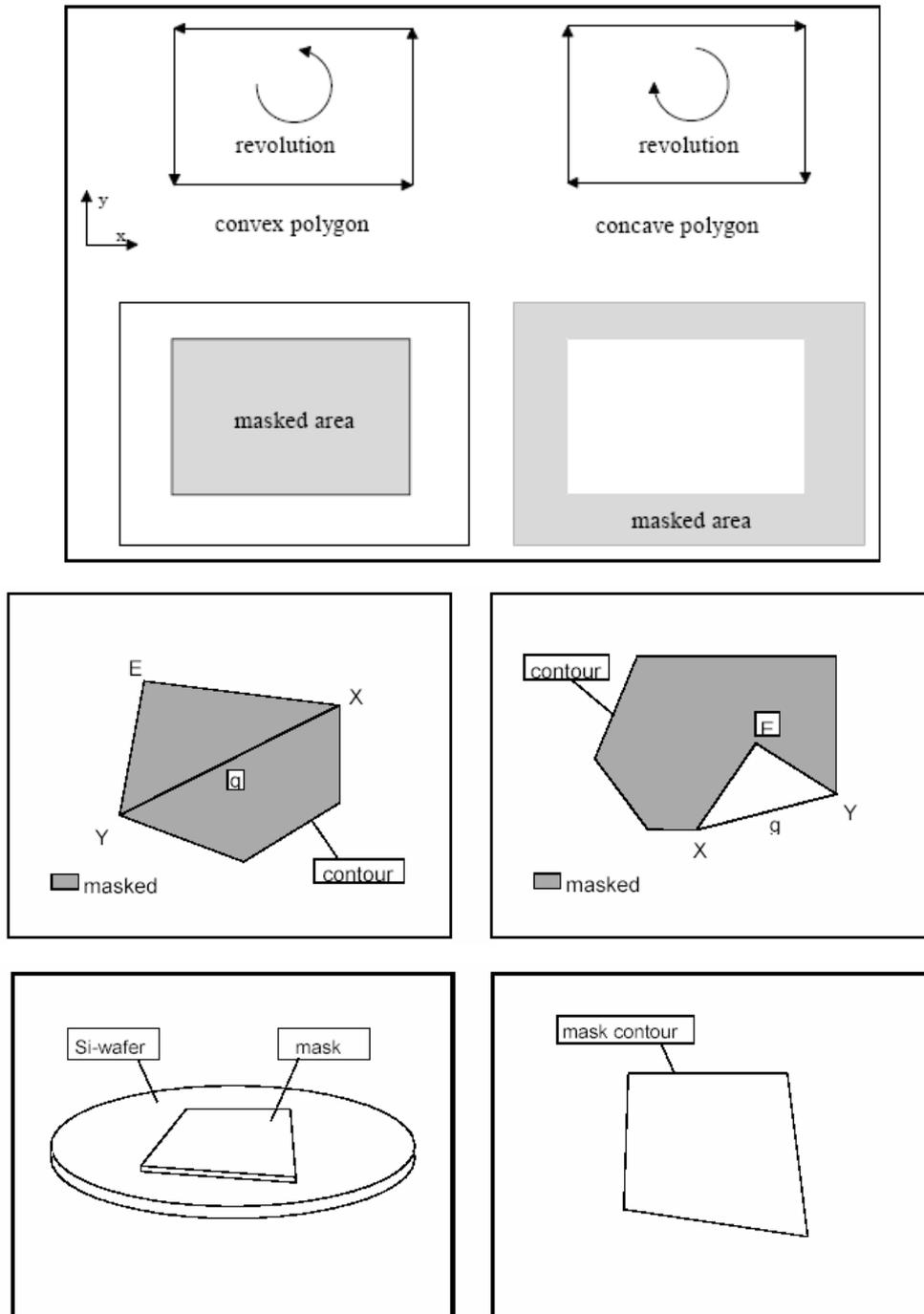


Figure 2.21: Mask Design of SIMODE [24]

SIMODE uses geometric rules to simulate the etch processes as shown in Fig 2.22. Based on the etch rates of the several sidewalls, etch rates of the polygon lines are defined. The etch rate of the polygon lines on the lower etch are composed of the etch rate of the sidewall and of the etch bottom, respectively as summarized in Fig. 2.23.

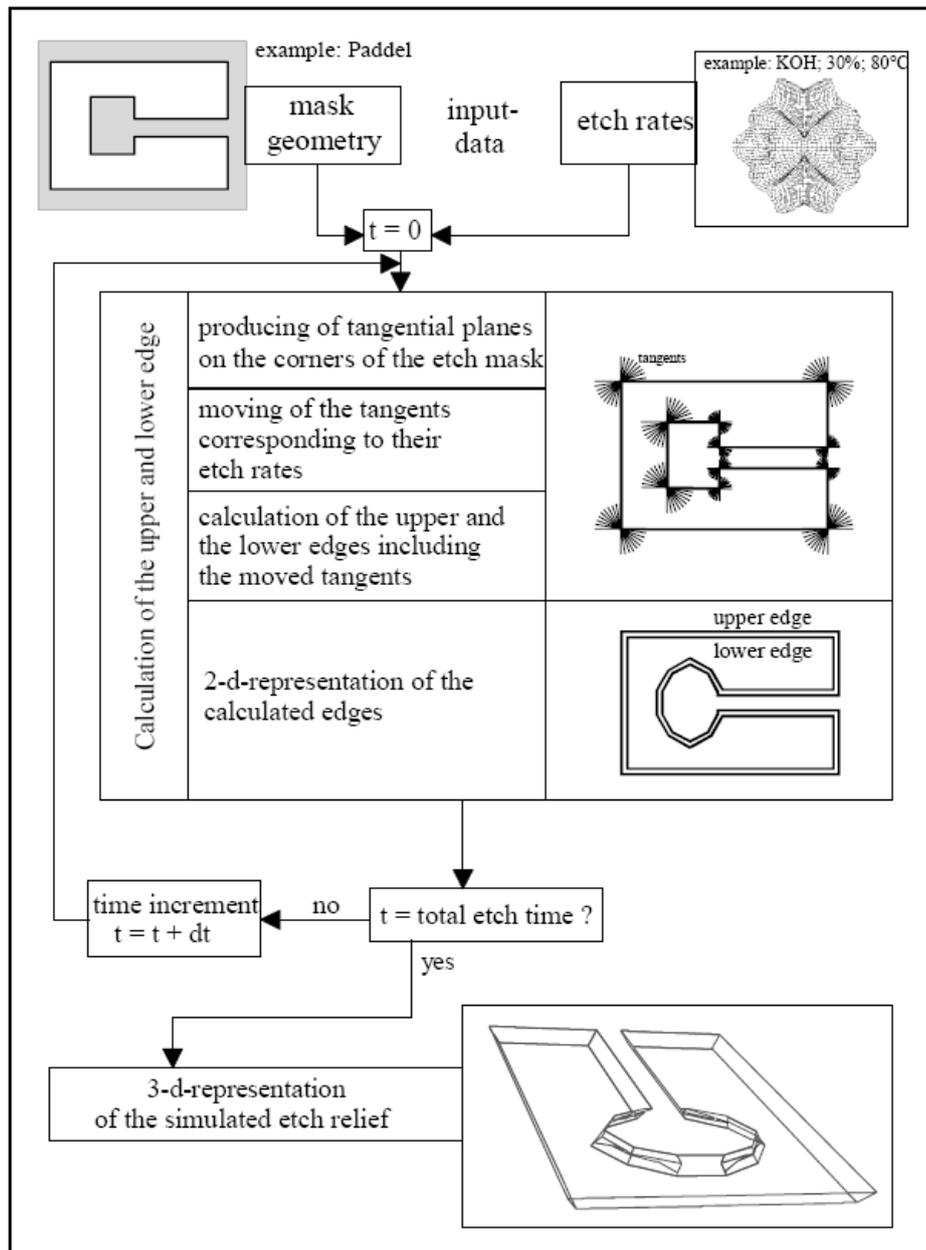


Figure 2.22: SIMODE Flow Chart [24]

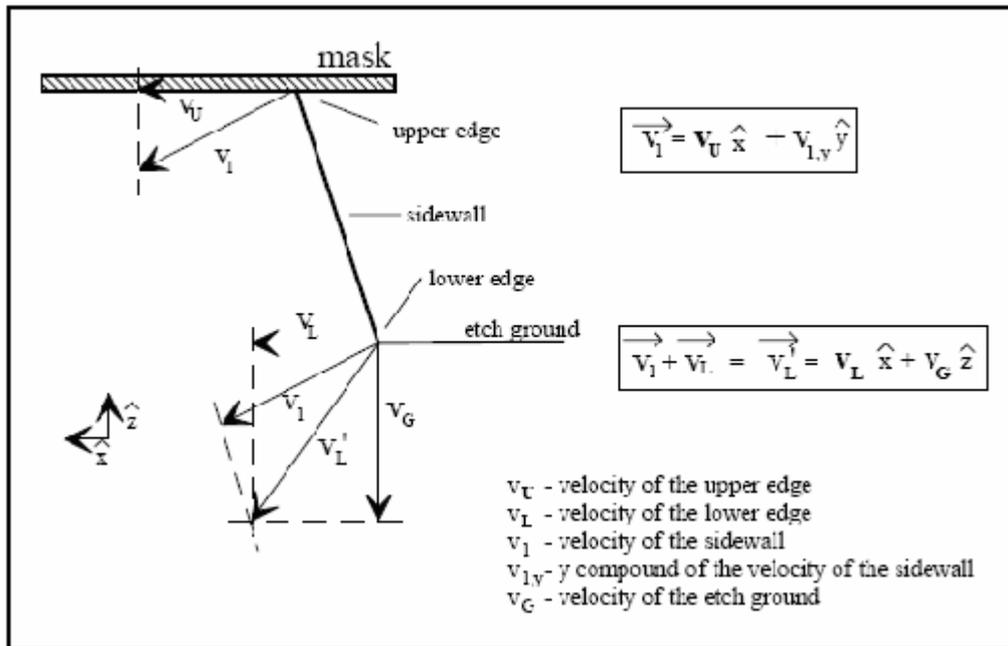


Figure 2.23: Velocity profile of a SIMODE process [24]

The sidewalls are described by the polygon line (of mask) and the sidewall angle, which is based on the etchant used. The sidewall may consist of two planes also (two-part sidewall) as illustrated in Fig. 2.24. The etch rates of the upper and lower surfaces; the sidewall angles and the ratio between the planes of the two-part sidewalls are stored for each etchant within the code.

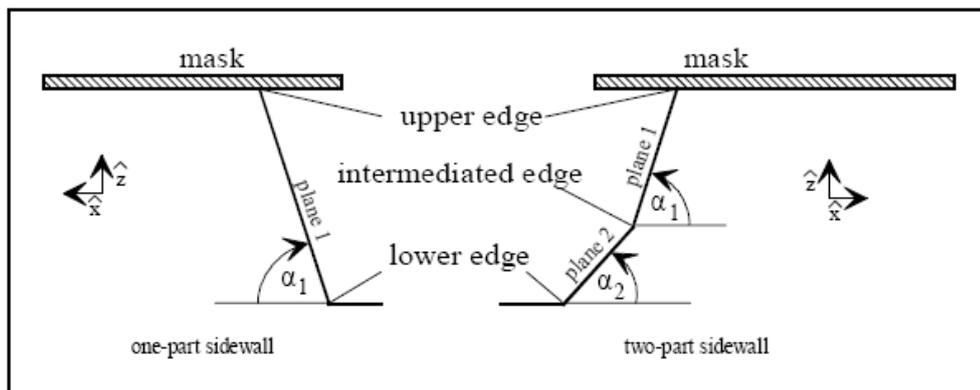


Figure 2.24: Sidewall Profiles [24]

At each time step, in every corner with angle greater than  $3^\circ$  tangents are produced with an interval of  $3^\circ$  beginning with the first at the polygon line  $n$  up to the polygon line  $(n+1)$ . Every tangent represents a possible new sidewall. The procedure for creating the tangents is different for convex and concave corners.

For concave corners, only the tangents that lie in the masked areas are created whereas in convex corners the tangents, which are outside the masked areas are taken into account. After the creation of the tangents they are shifted into the direction of the masked area according to their etch rates. The new contour is created using these shifted tangents. The process is summarized in Fig.2.25.

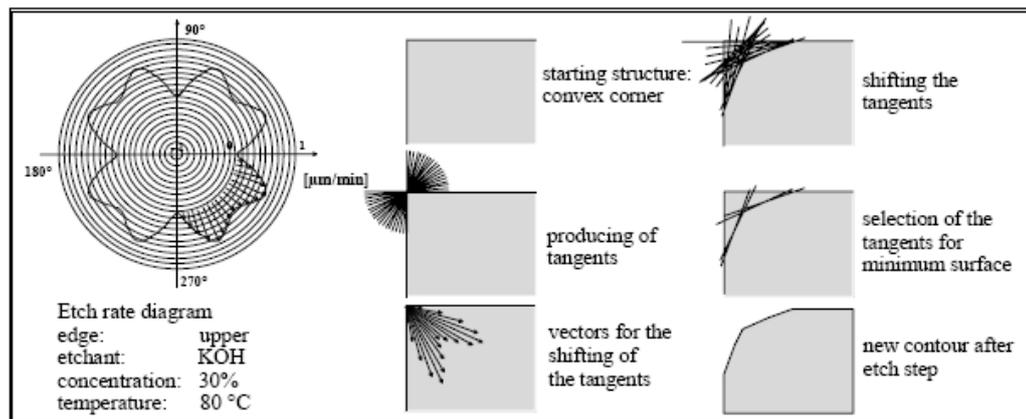


Figure 2.25: Simulation of one etch step in SIMODE [24]

In certain cases due to points of intersection of non-neighboring lines, two or more polygons can be developed. After the calculation of the upper and lower edges, the construction of the three-dimensional shape is started. First, two half shapes are generated for the upper and lower edges. For the construction of the half shapes, all polygon lines of the upper and lower edge will be transformed into planes. This is realized by assigning typical etch sidewalls to the polygon lines corresponding to their angle in the plane. Fig. 2.26 shows the flow chart of the construction of the three-dimensional shape.

Due to the two-dimensional simplification of the orientation, dependent etching some basic conditions were kept to get accurate results. The model describing the etch relief by two-dimensional polygons requires that the etch relief can be described fully by projection of typical etch sidewalls. Therefore, it has to be guaranteed that no planes, which do not correspond to the typical sidewalls produced by this etchant, develop during etching. A basic condition for this is

that the start conditions of the etch process are always flat masked wafers. This excludes a change of the etching fluid as well as a change of the mask during the etch process.

On edges where sidewalls enclose an angle greater than the used angle resolution ( $3^\circ$ ), planes can occur, which do not appear on unaffected mask edges. As the distance between the border of the etch relief and the start etch mask is increased, the more it is possible that such edges appear.

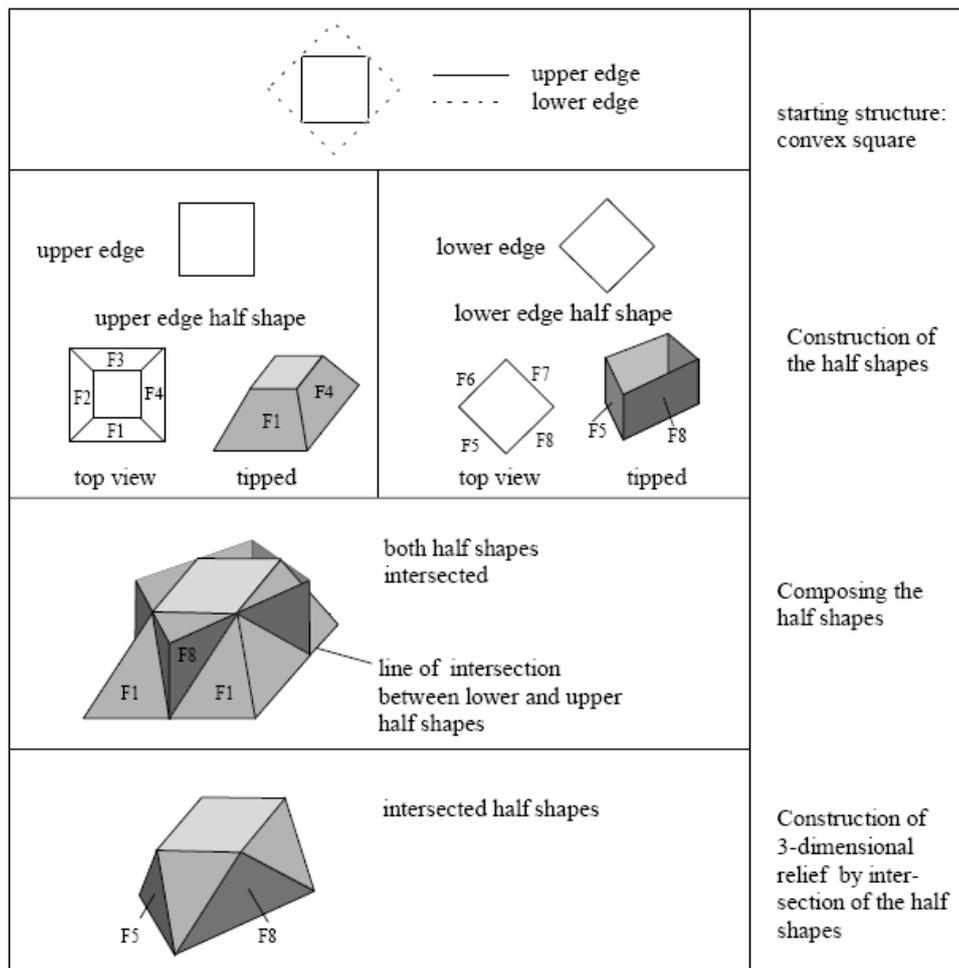


Figure 2.26: Construction of three-dimensional shape [24]

For the three-dimensional projection, there are limits related with the intersection of the two half shapes. During the etch process; planes from the upper or lower edge can be separated. Separated planes have a temporary, but at this moment, an

important influence for the development of the three-dimensional etch relief. The occurrence of this phenomenon can be realized when the calculated lower edge of the three-dimensional etch relief is not equal to the edges of the two-dimensional calculation. The value of the resulting error has the same order of magnitude like this difference.

### **2.3.7. Commercial Packages**

There are also a number of commercial codes for simulation of MEMS devices. But these codes mainly concentrated on the simulation of the behavior and properties of the devices created, and few programs on simulation of micro-fabrication processes can be found in the market.

#### **2.3.7.1. Memulator**

Memulator is a software from Coventor for process emulation and virtual prototyping of MEMS and other semiconductor devices [25]. Memulator is based on volume element technology and supports mask updates, process changes and complex semiconductor fabrication processes.

Memulator can simulate the material addition processes: physical vapor deposition, chemical vapor deposition, conformal deposition, metal lift-off deposition, epitaxial deposition, snowfall deposition, straight deposition, wafer bonding and electrochemical deposition. The program can also simulate the removal processes: reactive ion etching, wet release etching, wet isotropic etching, wet anisotropic etching, rate dependent etching and chemical mechanical polishing.

The software is getting the mask inputs in GDSII format. The mask shapes can be interconnected using the Boolean operations. The program gives three-dimensional visual outputs and dimensions, besides the results can be exported to Ansys for further processing.

### 2.3.7.2. Athena

Athena Process Simulation Framework is the product of Silvaco company [26].

There are seven modules in this framework as shown in Fig. 2.27:

- “SSuprem4” is a two-dimensional process simulator, which includes diffusion, implantation, oxidation, silicidation and epitaxy.
- “MC Implant” is a physically based 3D ion implantation simulator to model stopping and ranges in crystalline and amorphous materials. It accurately predicts implant profiles and damage for all major ion/target combinations.
- “Elite” is a two-dimensional moving boundary topography simulator for modeling physical etch, deposition, reflow and CMP planarization processes.
- “MC Etch/Depo” is an advanced topology simulator. It includes several Monte Carlo based models for simulation of various etch and deposit processes, which use a flux of atomic particles.
- “Optolith” is a non-planar 2D lithography simulator that models all aspects of submicron lithography: imaging, exposure, photoresist bake and development.
- “SSuprem3” is a one dimensional silicon process simulator used in the prediction of doping profiles and layer thicknesses.
- “SPDB” is a database manager containing experimental and simulated doping profiles and process recipes.

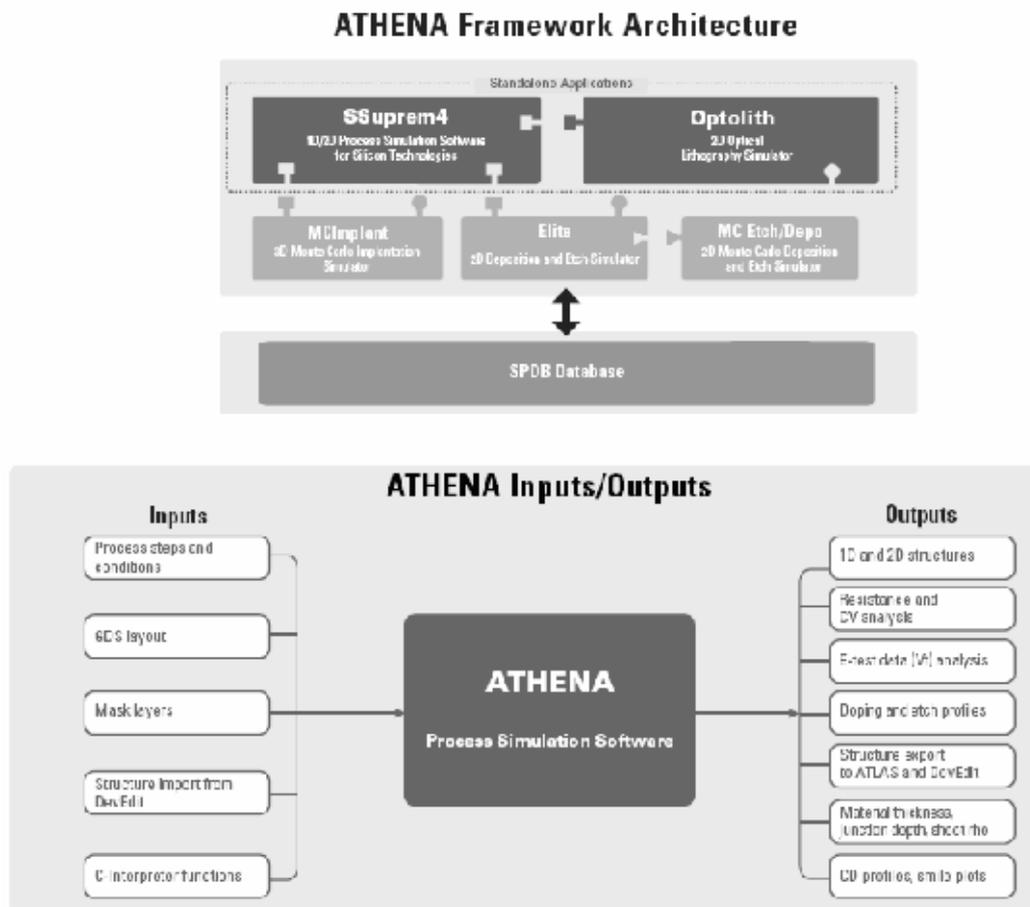


Figure 2.27: ATHENA Framework Architecture [26]

## 2.4. Cellular Automata

Cellular Automata is defined as a system defined at discrete time steps with a discrete spatial geometry (generally a regular lattice) [27]. In terms of discrete variables, the state of the system is defined at each point in time. The Cellular automata is specified in terms of rules that define how the state changes between time intervals.

The history of the Cellular Automata dates back to 1940's with Stanislas Ulam. This mathematician was interested in the evolution of graphic constructions generated by simple rules. The base of his construction was a two-dimensional space divided into "cells", a sort of grid. Each of these cells could have two states: ON or OFF. Starting from a given pattern, the following generation was

determined according to neighborhood rules. For example, if a cell was in contact with two "ON" cells, it would switch on too; otherwise it would switch off. By using one of the first computers, he found out that this simple rule generates complex and graceful shapes.

In 1970 the cellular automata is introduced with "The Game of Life" by John Horton Conway. The game of life is based on a grid constituted of cells shown in Fig. 2.28:

|    |    |    |    |    |
|----|----|----|----|----|
| 00 | 01 | 02 | 03 | 04 |
| 10 | 11 | 12 | 13 | 14 |
| 20 | 21 | 22 | 23 | 24 |

Figure 2.28: Example of a Starting Pattern [27]

The universe of the game is rectangle cells of 3 by 5. The cells are numbered 0 to 2 vertically and 0 to 4 horizontally. White cells are the active ones. The adjoin cells including the diagonals of a cell is its neighbors.

|    |   |    |   |    |
|----|---|----|---|----|
| 00 | ● | ●  | ● | 04 |
| 10 | ● | 12 | ● | 14 |
| 20 | ● | ●  | ● | 24 |

Figure 2.29: Determination of the Neighborhood [27]

Fig.2.29. demonstrates the active cells and the neighbors of the cell 12. There are three simple rules of the game:

- One inactive cell surrounded by three active cells is become active (born).
- One active cell surrounded by two or three active cells remains active.
- In other cases, the cell dies or remains inactive.

Interpretation of these rules are: a population is needed for birth (3 in this case), that the cells cannot survive to a too wide isolation and too much population (more then 3) will kill them.

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 | 2 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 3 | 2 | 1 |

Figure 2.30: First Generation [27]

|    |    |    |    |    |
|----|----|----|----|----|
| 00 | 01 | 02 | 03 | 04 |
| 10 | 11 | 12 | 13 | 14 |
| 20 | 21 | 22 | 23 | 24 |

Figure 2.31: Second Generation [27]

In this example, three fundamental properties of Cellular Automata is shown [28]:

- **Parallelism:** A system is said to be parallel when its constituents evolve simultaneously and independently. In that case, cells update is performed independently of each other.
- **Locality:** The new state of a cell only depends on its actual state and on the neighborhood.
- **Homogeneity:** The laws are universal; they are common to the whole space of cellular automation.

Cellular Automata applications are diverse and numerous. Fundamentally, Cellular Automation constitutes completely known universes. In a Cellular Automation, laws are simple and completely known. One can then test and analyze the global behavior of a simplified universe, for example:

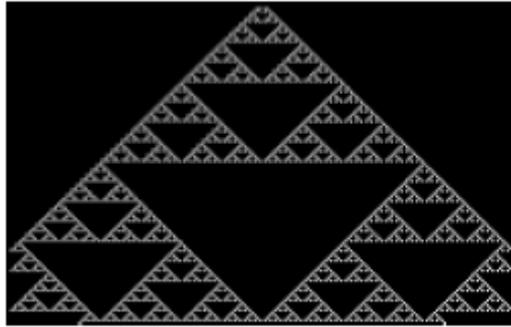


Figure 2.32: Example of 1-D CA Pascal Triangle [27]

- Simulation of gas behavior. A gas is composed of a set of molecules whose behavior depends on the one of neighboring molecules.
- Study of ferromagnetism according to Ising model : this model (1925) represents the material as a network in which each node is in a given magnetic state. This state, in this case one of the two orientations of the spins of certain electrons, depends on the state of the neighboring nodes.
- Simulation of percolation process.
- Simulation of forest fire propagation.
- In a different field, Cellular Automaton can be used as an alternative to differential equations.
- Conception of massive parallel computers.
- Simulation and study of urban development.
- Simulation of crystallization process.

## 2.5. Closure

Out of the micromachining techniques described in the chapter, anisotropic etching is the focus of the past research efforts and commercial programs. Using geometric or cellular automata techniques, this process is being tried to simulate. Because of the speed of the computers available in the market, and flexibility of the mask patterns that can be simulated, cellular automata was chosen to simulate the micromachining processes.

## CHAPTER 3

### WAFER EDITOR

#### 3.1. Introduction

First step in describing the process for simulation in MemsEagle is the creation of the wafer. A wafer editor is designed to create the wafer by entering the dimensions, crystal orientation, doping concentration and type. The details of the editor follow.

#### 3.2. Crystal Orientations

The developed software takes into consideration the wafers widely utilized in MEMS fabrication processes:

1. (100) Wafer flat on (110),
2. (100) Wafer flat on (100),
3. (110) Wafer flat on (100),
4. (111) Wafer flat on (110).

Fig. 3.1 illustrates the crystal lattice and the crystalline structure of common wafers of various orientations while Fig. 3.2 demonstrates the top view of the above-mentioned crystal lattices. To represent these wafers with different crystallographic orientations, a cell containing eighteen Silicon atoms is taken into consideration.

In CA approach, the atoms are not individually represented. Instead, a generic cell representing the mechanic behavior of the silicon atom is facilitated for the sake of convenience. Therefore, in the wafer editor, CA cells having relatively larger dimensions (on the order of  $1\mu\text{m}^3$  for a silicon crystal lattice) are taken

into account. In this approach, two types of CA cells (Type 1 and 2) are utilized based on the arrangement of neighboring CA cells. Notice that the difference between type 1 and type 2 cells arises due to the choice of the origin for the local coordinate system. (i.e. shift of the origin in the z direction) To model the cell structure in the code, an array containing the normalized spatial coordinates of each cell is generated. Table 3.1 tabulates the coordinates of each neighboring cell for a particular wafer. Note that, the local coordinate systems are illustrated in Fig. 3.1 as well.

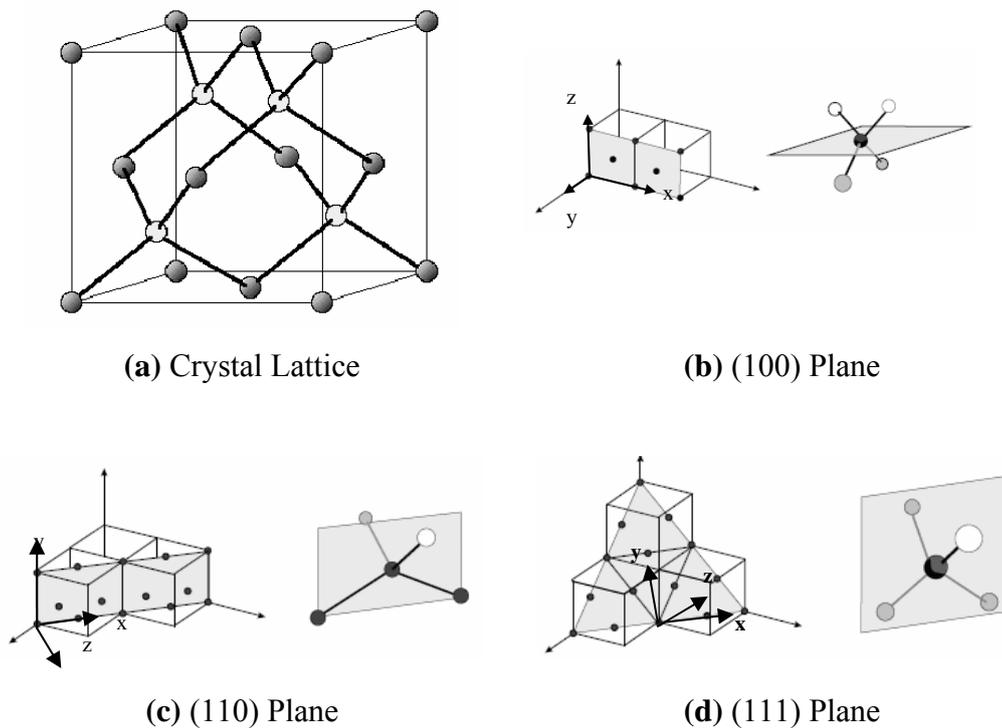
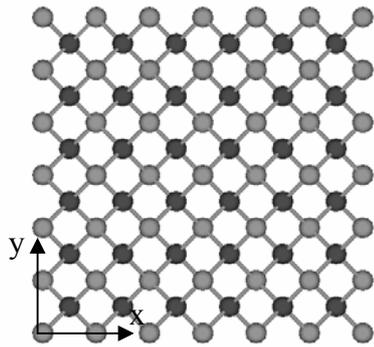


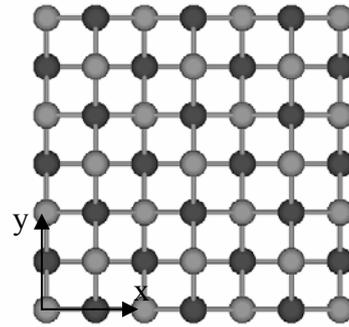
Figure 3.1: Crystal Orientations at different planes

In wafer editor, the user is expected to enter the wafer dimensions through the corresponding dialog box as shown in Fig. 3.3. As can be seen, the orientation of the wafer is set through a set of radio buttons. The last information needed for creating the silicon wafer is the dopant concentration and type. The dopant type is selected via the radio button labeled “p/n”. After selecting the type, the dopant concentration is entered. Notice that the default value for the dopant concentration is  $1 \times 10^{15} \text{ cm}^{-3}$ .

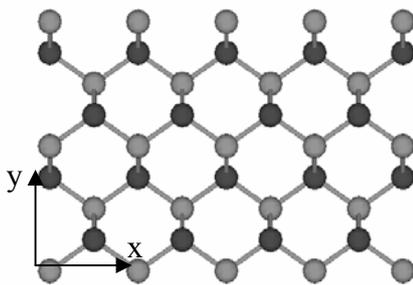
Notice that to accommodate wafers with larger dimensions, a scale factor is applied to the entered dimensions, such that the overall wafer could be efficiently modeled using the limited resources of an ordinary personal computer. (processing power and memory capacity) This subject is further discussed in detail in Section 3.3.



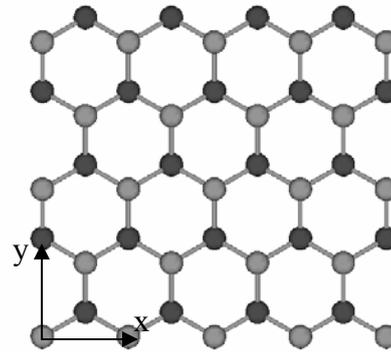
(a) (100) Crystal lattice flat on (100).



(b) (100) Crystal lattice flat on (110).



(c) (110) Crystal lattice.



(d) (111) Crystal lattice.

Figure 3.2: Top view of various crystal lattices with different crystallographic orientations.

Table 3.1: Spatial coordinates of neighboring Silicon atoms for various crystallographic orientations

|                                    |                          | Type 1             |                    |                    | Type 2             |                    |                    |
|------------------------------------|--------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
|                                    |                          | x[ $\mu\text{m}$ ] | y[ $\mu\text{m}$ ] | z[ $\mu\text{m}$ ] | x[ $\mu\text{m}$ ] | y[ $\mu\text{m}$ ] | z[ $\mu\text{m}$ ] |
| (100)<br>Wafer<br>Flat on<br>(100) | 1 <sup>st</sup> Neighbor | +0.25              | +0.25              | +0.25              | +0.25              | +0.25              | -0.25              |
|                                    | 2 <sup>nd</sup> Neighbor | +0.25              | -0.25              | -0.25              | +0.25              | -0.25              | +0.25              |
|                                    | 3 <sup>rd</sup> Neighbor | -0.25              | +0.25              | -0.25              | -0.25              | +0.25              | +0.25              |
|                                    | 4 <sup>th</sup> Neighbor | -0.25              | -0.25              | +0.25              | -0.25              | -0.25              | -0.25              |
| (100)<br>Wafer<br>Flat on<br>(110) | 1 <sup>st</sup> Neighbor | +0.3535            | 0                  | +0.25              | +0.3535            | 0                  | -0.25              |
|                                    | 2 <sup>nd</sup> Neighbor | 0                  | +0.3535            | -0.25              | 0                  | +0.3535            | +0.25              |
|                                    | 3 <sup>rd</sup> Neighbor | -0.3535            | 0                  | +0.25              | -0.3535            | 0                  | -0.25              |
|                                    | 4 <sup>th</sup> Neighbor | 0                  | -0.3535            | -0.25              | 0                  | -0.3535            | +0.25              |
| (110)<br>Wafer<br>Flat on<br>(110) | 1 <sup>st</sup> Neighbor | +0.3535            | +0.25              | 0                  | -0.3535            | -0.25              | 0                  |
|                                    | 2 <sup>nd</sup> Neighbor | -0.3535            | +0.25              | 0                  | +0.3535            | -0.25              | 0                  |
|                                    | 3 <sup>rd</sup> Neighbor | 0                  | -0.25              | -0.3535            | 0                  | +0.25              | +0.3535            |
|                                    | 4 <sup>th</sup> Neighbor | 0                  | -0.25              | +0.3535            | 0                  | +0.25              | -0.3535            |
| (111)<br>Wafer<br>Flat on<br>(110) | 1 <sup>st</sup> Neighbor | +0.3535            | +0.204             | -0.144             | -0.3535            | -0.204             | +0.144             |
|                                    | 2 <sup>nd</sup> Neighbor | -0.3535            | +0.204             | -0.144             | +0.3535            | -0.204             | +0.144             |
|                                    | 3 <sup>rd</sup> Neighbor | 0                  | -0.408             | -0.144             | 0                  | +0.408             | +0.144             |
|                                    | 4 <sup>th</sup> Neighbor | 0                  | 0                  | +0.432             | 0                  | 0                  | -0.432             |

When the command button “Create” is depressed, the code generates the wafer with the chosen type using the following algorithm:

- An array of cells are created using the “Virtual Surface” class, on which the x, y, z location in space, the link type, mass rate, doping concentration, material type, and neighbor information can be stored. This class is further elaborated in the process editor.
- An ordinary CA cell has a pre-determined size. For instance, the distance between the neighboring cells used to model (111) wafer is  $0.432\mu\text{m}$ . Based on the given wafer dimensions, an array of cells contained within the wafer are created based on that characteristic dimension of the CA cell.

- With the given wafer orientation, a two dimensional surface is created and the CA cells on this surface are stored in the VS class.

Except the (110) wafer, all the other wafer types have just one type of CA cell on the surface, whereas the (110) wafer has both of them. Based on the x and y locations of the cells, the type of the corresponding CA cell is determined; the details are further discussed in the section 3.3.

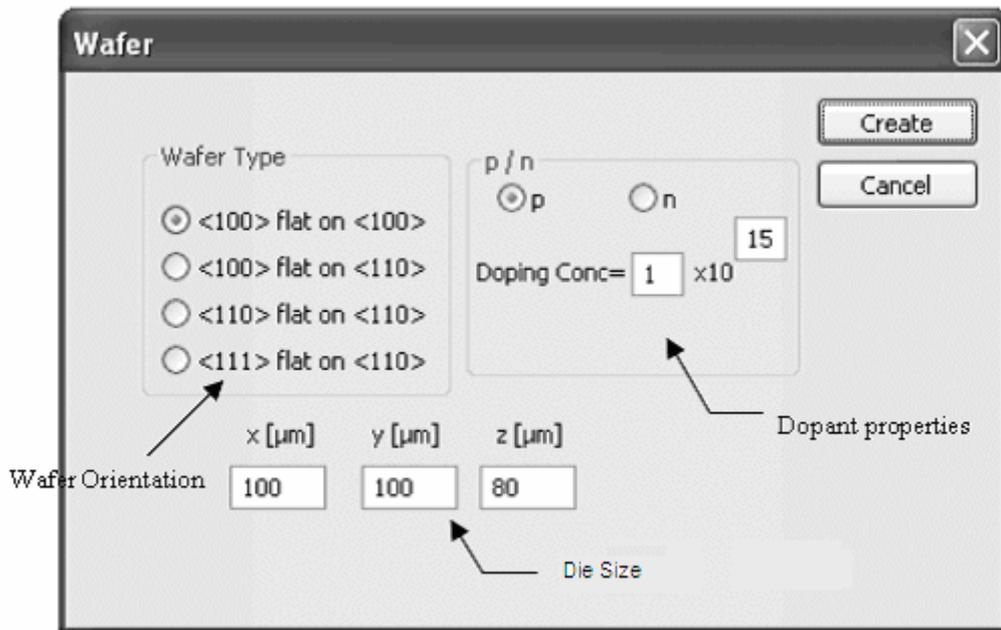


Figure 3.3: Wafer editor

### 3.3. Wafer Size Modifications

Based on the wafer type selected in the wafer editor, MemsEagle software finds the coordinates and the number of CA cells to be generated for simulation. The row and column on which the CA cell lies determines the cell coordinates at the surface of the silicon substrate. The following scheme summarizes this process for different crystal orientations, when no scaling is applied.

For (100) wafer flat on (100) direction:

$$x(i,j) = \begin{cases} x = i + 0.5, & 0 \leq x \leq l_w \wedge j(\bmod)2 = 1 \\ x = i, & 0 \leq x \leq l_w \wedge j(\bmod)2 = 0 \end{cases} \quad (3.1)$$

$$y(i, j) = j \cdot 0.5, \quad 0 \leq y \leq w_w \quad (3.2)$$

For (100) wafer flat on (110) direction:

$$x(i, j) = i \cdot 0.7071, \quad 0 \leq x \leq l_w \quad (3.3)$$

$$y(i, j) = j \cdot 0.7071, \quad 0 \leq y \leq w_w \quad (3.4)$$

For (110) wafer flat on (100) direction:

$$x(i, j) = i \cdot 0.3535, \quad 0 \leq x \leq l_w \quad (3.5)$$

$$y(i,j) = \begin{cases} y = j + 0.25, & 0 \leq y \leq w_w \wedge i(\bmod)2 = 1 \\ y = j, & 0 \leq y \leq w_w \wedge i(\bmod)2 = 0 \end{cases} \quad (3.6)$$

For (111) wafer flat on (110) direction:

$$x(i, j) = i \cdot 0.3535, \quad 0 \leq x \leq l_w \quad (3.7)$$

$$y(i,j) = \begin{cases} y = j \cdot 1.224 + 0.612, & 0 \leq y \leq w_w \wedge i(\bmod)2 = 1 \\ y = j \cdot 1.224, & 0 \leq y \leq w_w \wedge i(\bmod)2 = 0 \end{cases} \quad (3.8)$$

Where  $x, y$  are the cell coordinates,  $j$  is the row number,  $i$  is the column number and  $l_w$  and  $w_w$  are the length and width of the wafer respectively. The number of the surface cells generated is varying for different wafer types. The CA cells generated using the wafer editor for a substrate size  $30 \times 20 \mu\text{m}^2$ , were illustrated by Fig 3.4.

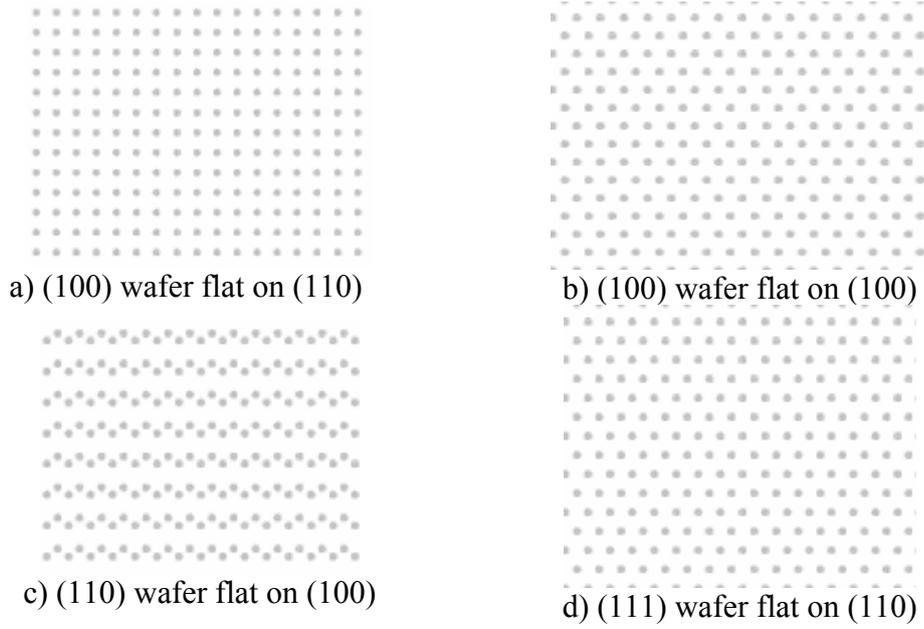


Figure 3.4: Cells generated by Wafer Editor

Although the wafer can be sized by the user arbitrarily, because of the “Cellular Automata” approach used, the wafer is scaled to reduce the corresponding computational cost above a pre-determined limit. In order permit the usage of the software on common personal computers; the maximum operational surface area of the wafer was restricted by  $1500\mu\text{m}^2$ . If the user tries to exceed this limit, the wafer is scaled down and the etch rates are modified accordingly. The following sequence summarizes the modifications done:

$$l_w \cdot w_w = A_w \quad (3.9)$$

$$\sigma = \begin{cases} 1, & A_w \leq 1500 \\ \sqrt{\frac{A_w}{1500}}, & \textit{else} \end{cases} \quad (3.10)$$

$$E_{ef} = \frac{E_r}{\sigma} \quad (3.11)$$

where  $l_w$  is the length,  $w_w$  is the width of the wafer,  $A_w$  is the wafer upper surface area,  $E_r$  is the etch rate for a certain direction,  $E_{ef}$  is the effective etch rate and  $\sigma$  is the scaling factor.

### 3.4. Visualization

When the  $x$ ,  $y$  and  $z$  dimensions of the wafer are entered using the wafer editor, the borders of the wafer are drawn on the screen. Starting from the origin, the lines forming the frame are drawn by using the OpenGL functions. For instance the following OpenGL code draws a line between the points  $P_0(x_0, y_0, z_0)$  and  $P_1(x_1, y_1, z_1)$ :

```
glBegin(GL_LINES);  
    glVertex3f(x0, y0, z0);  
    glVertex3f(x1, y1, z1);  
glEnd();
```

The OpenGL automatically manages all the given coordinates with respect to the current graphic window. Notice that, for switching between window and OpenGL coordinates, `gluProject()` and `gluUnProject()` functions are used. The details about these functions are given in the Appendix B. When the wafer is created; the upper surface is drawn as a filled rectangle (as a filled mask) to be used as the base object of the mask editor. Following sequence is employed for creating polygons (filled) in OpenGL with the given vertices:

```
glBegin(GL_POLYGON);  
    glVertex3f(x0, y0, z0); glVertex3f(x1, y1, z1);  
    glVertex3f(x2, y2, z2); glVertex3f(x3, y3, z3);  
glEnd();
```

Fig. 3.5 illustrates a sample wafer created by the described OpenGL functions. As shown, the upper surface is a filled polygon whereas only the frames of the other surfaces have been drawn.

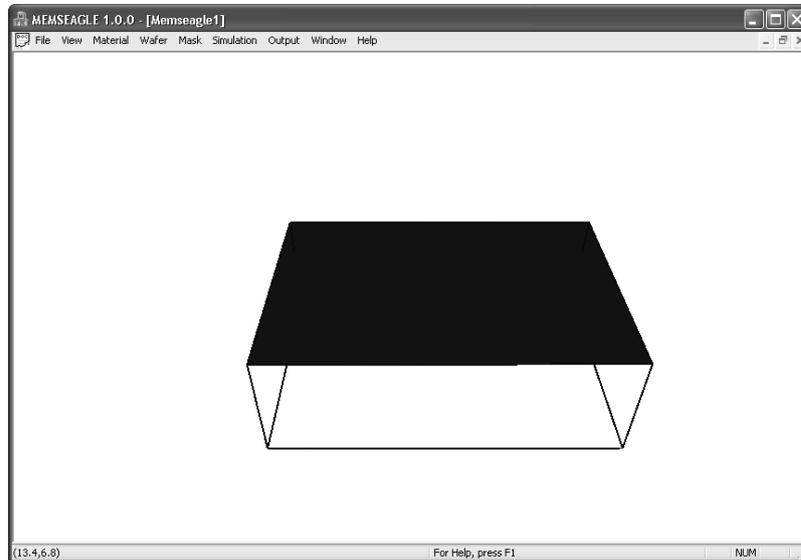


Figure 3.5: Wafer display

### 3.5. Closure

The wafer editor is designed for entering the information needed for creating the silicon wafer through which the MEMS device will be machined. The information entered is stored in the “VS” class created. Through the editor the size, orientation and dopant type and concentration properties of silicon are adjusted for a particular wafer suitable for MEMS device fabrication. Based on the array generated by the wafer editor, a virtual surface, based on the mask information, is produced at the later stages used by Memseagle software.

## **CHAPTER 4**

### **MASK EDITOR**

#### **4.1. Introduction**

After creating the wafer on which the MEMS device will be built, the masks are designed using the mask editor. Using the primitives and the modification options complex shapes can be created, and more than one mask can be designed for different projects. Fundamental principles of the mask editor are summarized in this chapter.

#### **4.2. Mask Editor Features**

During the generation of the substrate, the wafer was created with a mask covering the upper surface completely. The mask pattern was generated using this as a base. The user-interface of the mask editor is illustrated in Fig. 4.1. The shapes are created using the three primitives: rectangle, circle and polygon, as shown in Fig. 4.1. These simple objects can later be modified to have complex shapes that may have holes, obtuse-angle corners or other essential objects of a typical MEMS design. Note that, the shapes can be drawn as empty or filled objects.

Up to 50 masks can be generated for each project, and the masks can be saved or purged during the course of the design. In order to supply a user-friendly drawing environment, grid option is enabled. Notice that, the grid size can be adjusted using the combo box shown in Fig 4.1 and the user has the option to snap onto the grids or not.

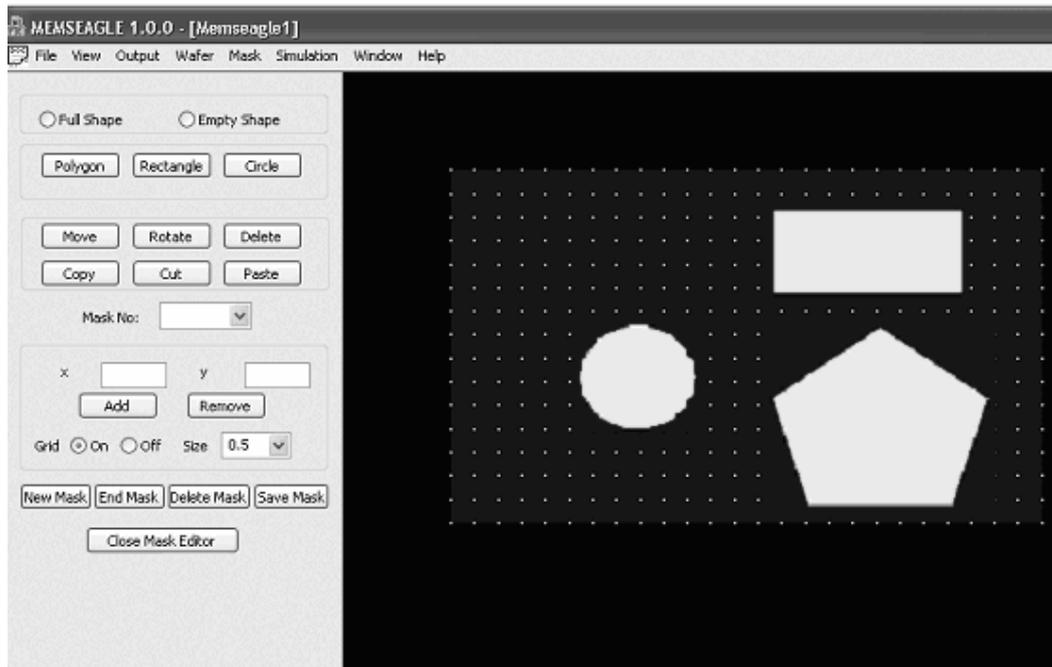


Figure 4.1: Mask Editor Interface

### 4.3. Mask Primitives

The designed mask editor employs various graphical primitives (objects) to create relatively complex mask shapes using the combinations of these. In the mask editor of MemsEagle, there are three primitives used for creating masks: rectangle, polygon, and circle. For the rectangle primitive, the starting corner location, length, and width information are needed whereas the center coordinates and the radius of the circle is the sufficient information for the circle entity. For creating polygons, coordinates of each vertex are to be entered by the user.

In mask editor, two types of shapes can be created: empty and filled. The mask creation process starts out with a filled rectangle, which apparently covers the upper surface of the whole wafer. After that, the user can create a complex mask by adding, purging, and modifying the primitives (either filled or empty) on this

filled base rectangle, using the user-friendly features of the mask editor. The next section discusses these modifications.

For enabling more control over the shapes drawn, the user may enter the grid mode by selecting the “grid on” radio button as shown in Fig 4.1. In grid mode, guide points are drawn on the screen for creating or modifying primitives.

The OpenGL functions used for creating the points are `glVertex3f()` and `glBegin(GL_POINT)`. By entering the `GL_POINT` mode, the code started to draw points on the locations entered by `glVertex3f()` until `glEnd()` function is called. The distance between the grids can be changed using the combo box.

When the grid mode is on, the mouse clicks on the screens are re-adjusted for the nearest grid point and the vertex location is stored as such.

#### **4.4. Modifications**

The created objects can be moved, rotated, resized or purged using the command buttons of the mask editor as shown Fig. 4.1. When the polygon, rectangle or circle buttons clicked, the software generates a new shape and stores it in the allocated array kept for the mask in the memory. On this array, the locations of the vertices along with the type of the shape (empty or filled) are stored. For each mask, a maximum of 50 geometric objects are allowed which hopefully yields a mask pattern accommodating most features of practical MEMS designs.

The modification on the individual graphical primitives can be carried out via the corresponding the modification buttons on the dialog box. To accomplish that, the mask editor employs various OpenGL functions. For instance, a label for each object is generated using `glLoadName()` function of OpenGL during creation phase. Likewise, `glSelectBuffer()` is utilized to select a certain graphical

primitive via mouse by simply returning the index of these objects previously labeled.

After that if the program goes into the modification mode, the buffer type is changed into `GL_RENDER` by using the command `glRenderMode()`. The location of the mouse when the left button is clicked is then checked by the software and if it intersects with one of the shapes, the return value of the `glRenderMode()` function becomes the integer assigned to that shape. Details of the selection process are further explained in detail in section 4.6.

According to the command button clicked, the following actions are taken to modify a particular shape:

- **Move:** The mask type remains the same. The user prompted to select the shape and the x, y displacements of the shape chosen is entered. The vertices of the shape are re-located.
- **Rotate:** The mask type remains the same. The user prompted to select the shape and the rotation angle along with the rotation center is entered. The vertices of the shape are re-located. If the rotation point is omitted, the shape is rotated with respect to the first vertex.
- **Delete:** The user picks up the shape and the selected primitive is deleted from the view screen. In addition, the shape number is decreased by one and the vertex information of that shape is deleted.
- **Copy:** The vertices and the type of the selected shape are stored in the mask array. The 0<sup>th</sup> member of the array is utilized for this action.
- **Cut:** The information is processed just like the copy operation described above; the difference lies in the fact that the selected shape is also deleted.
- **Paste:** The stored shape is drawn on the screen with the first vertex location re-entered.

After the modification has terminated, the code re-enters into the render mode. The difference between the selection mode and the render mode is that, in render mode the shapes entered using the OpenGL functions are displayed on the screen whereas in selection mode these shapes are used for getting information or modification indirectly without any modification on the screen.

#### **4.5. Transfer of Mask Pattern onto the Wafer**

After the drawing of a mask is finished, the user must click the end button. Then, the code starts to transfer the mask information onto the wafer.

The main idea behind locating the empty and filled areas lies in the colors used. The empty areas are yellow colored whereas the filled areas appear blue. By making use of the wafer type selected, the code already determines the location of the silicon cells on the surface between the mask and the wafer. For each cell, the color of the mask should be known.

First, the OpenGL coordinates, which the wafer dimensions and cell locations are entered, should be mapped onto the window coordinates by the `gluProject()` function. The matrices titled `modelview`, `transformation`, and `viewport`, are used to modify the screen size as well as the various viewing attributes.

By making use of the `glReadPixels()` function, the color of the mask on the location of the silicon cells is found. With the information from past processes and the material type, the surface on which the etching, doping or addition will occur is determined.

#### **4.6. Visualization**

The OpenGL functions and C++ algorithms utilized are described in this section. The algorithm used for creating and modifying the primitives, and displaying them on the viewport is summarized.

#### 4.6.1. Creating Primitives (Tessellation)

For creating the rectangle, polygon and circle primitives, the OpenGL function `glVertex3f()` can be used, in conjunction with the `glBegin()` function with `GL_POLYGON` argument. However, it is not possible to use `GL_POLYGON` argument for creating the polygons with obtuse-angle corners, due to possible intersection of the lines that make up the polygons created. For creating polygons with intersecting lines or with holes inside, tessellation functions of OpenGL are employed. OpenGL can directly display only simple convex polygons. A polygon is simple if the edges intersect only at vertices, there are no duplicate vertices, and exactly two edges meet at any vertex. If the user wishes to create concave polygons, polygons containing holes or polygons with intersecting edges, those polygons must first be subdivided into simple convex polygons before they can be displayed. Such subdivision is called tessellation, and OPENGL provides a collection of routines that perform tessellation. [30] In order to use tessellation for a polygon the following procedure should be followed in OpenGL:

1. Create a tessellation object using `gluNewTess()`.
2. Use `gluTessCallback()` several times to register callback functions to perform operations during the tessellation. The trickiest case for a callback function is when the tessellation algorithm detects an intersection and must call the function registered for the `GLU_TESS_COMBINE` callback.
3. Specify tessellation properties using `gluTessProperty()`.
4. Create and render tessellated polygons by specifying the contours of one or more closed polygons.
5. Delete the tessellation object with `gluDeleteTess()`.

Details of these functions are further discussed in Appendix B. Utilizing the pre-defined OpenGL functions, `viewpolygon()` was created for tessellation purposes.

Note that “t” was created using `gluTessCallback()` function. `GLU_TESS_WINDING_POSITIVE` argument was used for determining the interior of the polygon created, while using the user-defined `SetFilling()` function, the interior of the polygon was filled with the color chosen. Notice that, the vertex coordinate information is entered using the `AddVertexArray()` function.

```
void CmemseagleView::viewpolygon (int i,int j)
{
    CGLTesselator t;
    t.StartDef();

    t.gluTessProperty(GLU_TESS_WINDING_POSITIVE);
        t.SetFilling(TRUE);
        t.AddVertexArray(vertices[i][j]);
    t.EndDef();}
```

Fig 4.2 demonstrates the display of a mask shape created using OpenGL tessellation. The color of the filled area can be yellow or blue, with respect to the type of the shape chosen. The un-masked areas are drawn in yellow.

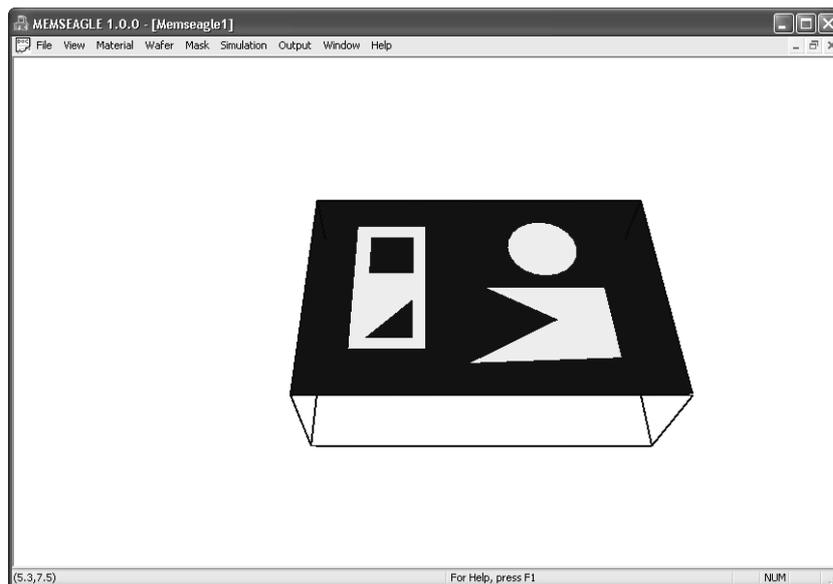


Figure 4.2: Mask Creation using Tessellation

#### 4.6.2. Modifying Primitives (Selection)

In order to modify the objects drawn on the screen, they should be selected first using the mouse. Typically, when using OpenGL's selection mechanism, first, the scene is drawn into the frame buffer and then selection mode is entered and the scene is refreshed. However, while in selection mode, the contents of the frame buffer do not change until selection mode is terminated. When selection mode is finished, OpenGL returns a list of the primitives that intersect the viewing volume. Note that, each primitive that intersects the viewing volume causes a selection hit. The list of primitives is actually returned as an array of integers which stands for the object names and related data "the hit records". This integer corresponds to the current contents of the name stack. The name stack is constructed by loading names onto it as primitives were drawn while in selection mode. Thus, when the list of names is returned, this information can be used to determine which primitives might have been selected on the screen by the user. The steps followed to select an object drawn is listed below:

1. Specify the array to be used for the returned hit records with `glSelectBuffer()`
2. Enter selection mode by specifying `GL_SELECT` with `glRenderMode()`.
3. Initialize the name stack using `glInitNames()` and `glPushName()`.
4. Define the viewing volume to use for selection. (The masked area)
5. Exit selection mode and process the returned selection data (the hit records).

In selection mode, a primitive that intersects with the viewing volume invokes a selection hit. Whenever a name-stack manipulation command is executed or `glRenderMode()` is called; OpenGL writes a hit record into the selection array if there's been a hit since the last time the stack had been manipulated or `glRenderMode()` had been invoked. With this process, objects that share the same name (for example, an object that's composed of more than one primitive) do not generate multiple hit records. Then, the tessellation objects (the convex polygons

that made up the polygon drawn by the user) are processed as one object, rather than multiple shapes to be modified.

#### 4.7. Sample Mask Shapes

The creation of the masks that were used to compare the simulation results with the real cases was displayed below, using different techniques. Desired mask patterns can be formed by modifying the three primitives (circle, polygon and rectangle). The mask patterns created were used in SIMODE for the same verification purposes also.

The mask pattern displayed in Fig 4.3 was created by using the circle primitives. By creating circles centered in point  $O(x,y)$ , the pattern was achieved. There are two filled and two empty circles used during the design. The center of the circle can be entered through the keyboard or by using the grid mode and left mouse button. This mask was later etched in KOH and the progress of this process was discussed in Chapter 7.

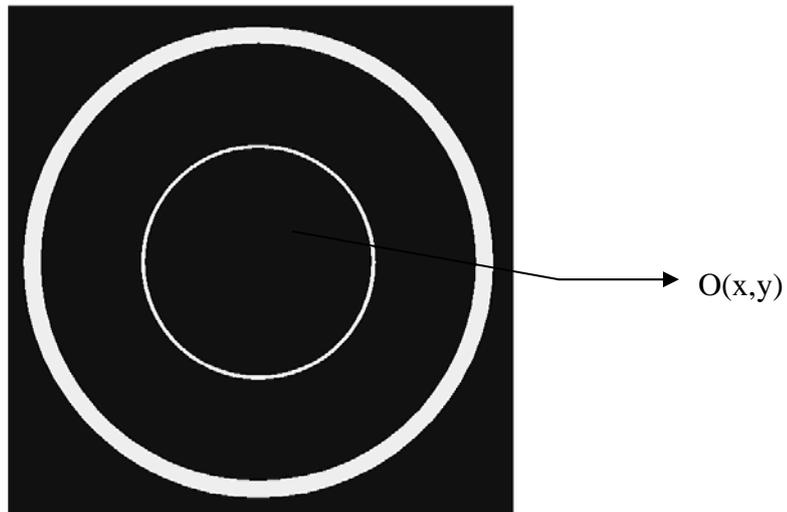


Figure 4.3: Co-centric Rings

The second mask pattern created was used to demonstrate the etch results for polygons that have obtuse angled vertices. The first shape was created using the rectangle primitive and then rotated  $45^\circ$ . The easiest way to create the other

shapes was to enter grid mode. After that by entering the vertices using polygon primitives, the resultant pattern was created.

For the last shape, another way of creation was possible. First using the polygon primitive, the user could create a triangle, then by rotating and copying this triangle, the four sides of the shape was set. The hole inside could be filled with a rectangle and the mask pattern was finished.

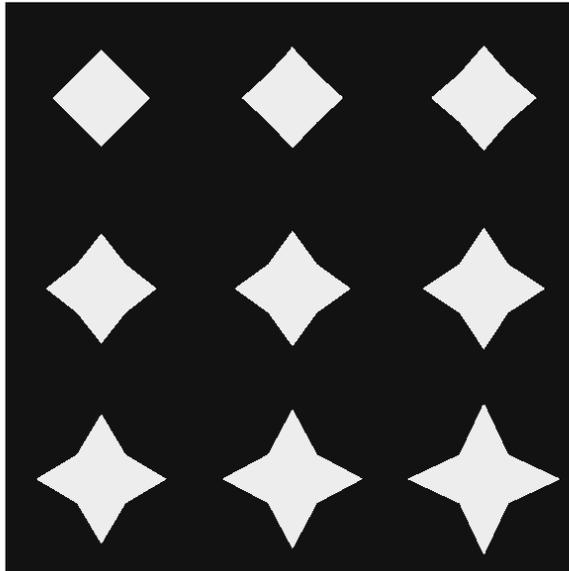


Figure 4.4: Obtuse Angled Shapes

This mask pattern was used for showing the effects of merging planes and the etching of right-angled shapes in wet anisotropic etching. The pattern was created by using the rectangle primitive, by rotating it  $45^\circ$  and copying it to different locations the pattern on Fig. 4.5 was achieved.

The pattern demonstrated in Fig. 4.6 was used to show the effects of the mask misalignments to the resultant etched shape. This pattern can be created in different ways but the simplest approach was to create the shape and then rotating it. The shape was created using the polygon primitive and by copying and rotating the object, the final mask pattern was obtained.



Figure 4.5: Rotated Squares



Figure 4.6: Misaligned Masks

The paddle pattern illustrated in Fig. 4.7 was used to display the compensation of the shapes surrounded by the etchant. In order to achieve this shape first an outer empty rectangle was created. Then there are two ways to follow, entering the grid mode and creating the shape by using a polygon primitive or creating rectangles and copying them.

The following rotated beam displayed in Fig 4.8 was made up of three rectangle primitives. First, the two thin rectangles were created by entering the grid mode and then they were rotated by  $45^\circ$ . The square was added finally and the mask

pattern was obtained. This shape was used to demonstrate the compensation of the rotated beams by the etchant.

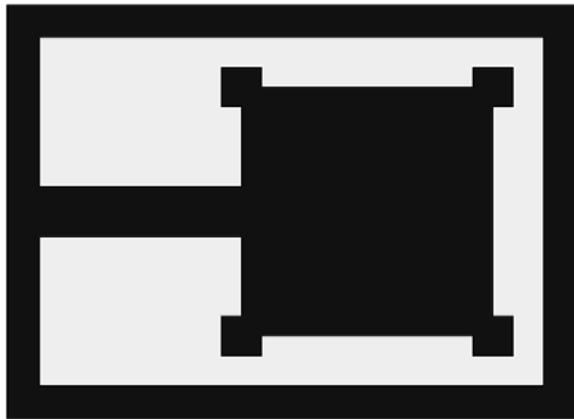


Figure 4.7: Paddle

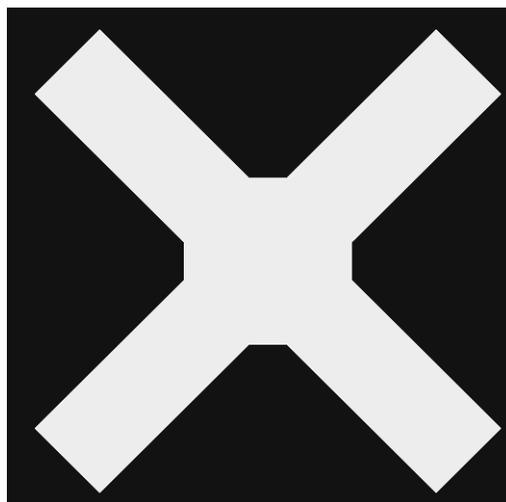


Figure 4.8: 45° Beams

This mask pattern was used to show the compensation of the triangle-cornered shapes in the anisotropic wet etching process. In order to obtain this mask, first the upper or lower half of the pattern was created using the polygon primitive. Then, by copying and rotating the shape by 180° the final shape was formed.

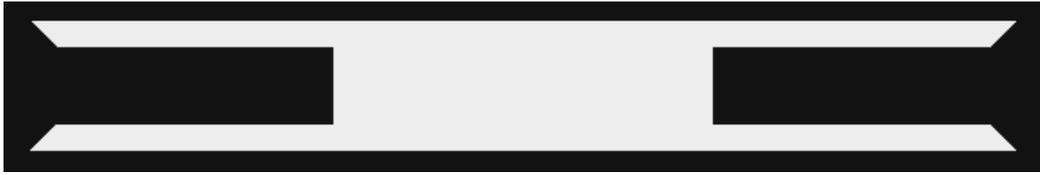


Figure 4.9: Triangle-Cornered Beams

The next sample (Fig 4.10) was used to simulate the compensation of the fingers of a mask pattern by the etchant. If this shape is etched by Deep Reactive Ion Etching or first boron doped and etched, the fingers of a comb drive can be obtained. The simplest way of creating this pattern was to create the one finger sample for each width first. The next step was to create the triangles by using the polygon primitive and copying the fingers upon these triangles.

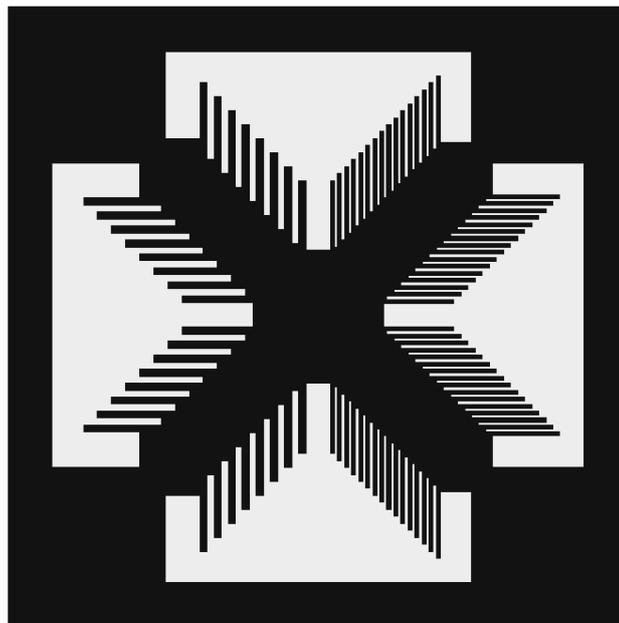


Figure 4.10: Compensation Fingers

This example illustrated in Fig. 4.11 shows a simple spur gear created by MemsEagle mask editor. The mask was generated using the circle and polygon primitives with modification options copy, paste and rotate. Hence, it is possible to obtain relatively complicated MEMS elements using simple objects of the editor.

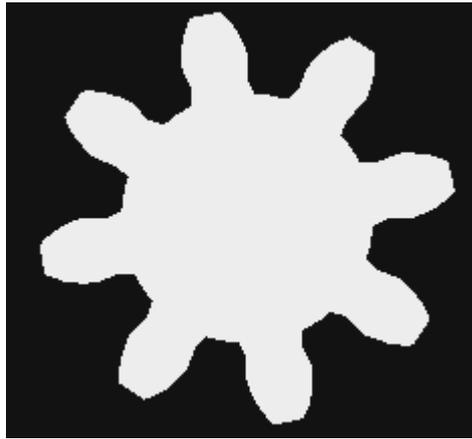


Figure 4.11: Spur Gear Created by the Mask Editor

#### **4.8 Closure**

By using the integrated simple mask editor, the user can draw complex shapes for simulation of the micro-fabrication processes. Up to 50 different masks can be created for a project, and each one can be associated with different processes. There was grid option available, which can be turned off and be chosen for different sizes.

## CHAPTER 5

### PROCESS EDITOR

#### 5.1. Introduction

After creating the wafer and designing the masks, the next step is the selection of the processes to be simulated. MemsEagle includes a process editor where the user selects the process sequences and the associated masks to be used. The micro machining processes that can be simulated by the program and the operating principles of the process editor is described in this chapter.

#### 5.2. Editor Features

The process editor employed in MemsEagle is capable of creating the environment for stand-alone micro-fabrication processes or a complete set of processes for creating a MEMS device by utilizing the project editor. The micro-fabrication processes and the mask patterns are added using the “Add” command button shown in Fig. 5.1. Note that, by utilizing the combo boxes for mask and process selection the necessary dialog boxes are reached for entering the fabrication variables. Figures 5.2 and 5.3 illustrate the dialog boxes used for different micro-fabrication processes simulated.

The processes can later be modified by using the “Choose Process” combo box. When the project is ready for simulation, by exiting using the “OK” command button, the user could return to the main MemsEagle interface for starting the process. Rather than using the project editor, MemsEagle software can simulate single process by simply entering the “process selector” dialog box. Notice that, the fabrication processes to be simulated can be modified through this interface.



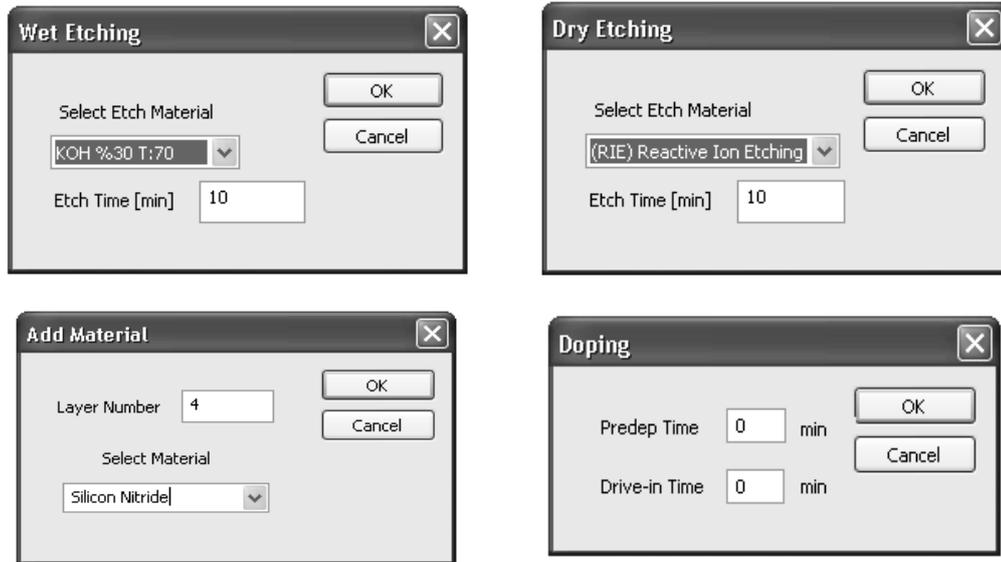


Figure 5.3: Micro-Fabrication Dialog Boxes

### 5.3. Simulated Processes

MemsEagle has mainly concentrated on simulating the etching processes, especially the anisotropic etching. With the etching processes, additive processes like doping and oxidation are also included in the package for self-completeness of MEMS design process.

#### 5.3.1. Wet Etching

Most of the etchants used in wet etching, results in an anisotropic profile for silicon wafers. Based on the cellular automata approach, MemsEagle simulates the resultant wafer shape after the micro-fabrication process applied. The etch rates of the etchants used are given in Table 5.1.

Table 5.1: Etch Rates in  $\mu\text{m}/\text{min}$

| Etchant                     | Planes |       |       |       |
|-----------------------------|--------|-------|-------|-------|
|                             | 100    | 110   | 111   | 311   |
| EDP                         | 0.47   | 0.28  | 0.028 | 0.28  |
| Isotropic (HNA)             | 1      | 1     | 1     | 1     |
| %30KOH %70 H <sub>2</sub> O | 0.797  | 1.455 | 0.005 | 1.436 |
| %40KOH %60 H <sub>2</sub> O | 0.559  | 1.294 | 0.009 | 1.067 |
| %50KOH %50 H <sub>2</sub> O | 0.539  | 0.870 | 0.009 | 0.746 |
| TMAH %20                    | 0.603  | 1.114 | 1.223 | 0.017 |

### 5.3.2. Dry Etching

The only dry-etch process that can be simulated by MemsEagle is “Deep Reactive Ion Etching”. In each step, one layer of cells is removed from the surface of the wafer. Notice that, the etch rate was taken as  $1\mu\text{m}/\text{min}$ .

### 5.3.3. Doping

Doping process is simulated with two restrictions, which will be elaborated in Section 5.5.3. The inputs are the pre-deposition time and diffusion time. For doping concentration dependent etchants, doping concentration is checked by MemsEagle. If the concentration is higher than the threshold value (in certain cases below) the etch stops for that cell. For further information on the etching rates, the user is encouraged to refer to go to Appendix A.

### 5.3.4. Additive Processes

Several materials including Polysilicon, silicon dioxide, silicon nitride, aluminum and gold can be added using the additive process simulation. Only the un-masked areas are exposed to the additive materials and the thickness of the deposited material is needed to simulate the process.

It should be noted that as a general design principle, surface micro-machining has to be applied as the last fabrication process owing to the fact that the deposited films on the surface degrades a great deal when they are exposed to common bulk micromachining chemical agents. Therefore, based on this principle, MemsEagle is designed to deposit directly several stacked patterned layers over the surface shaped by bulk micromachining. The crystal orientations of these materials are different from that of the substrate. Hence, they are indicated by a different color. However, in order to transform the new virtual surface, these new cells are to be merged with the existing CA cells.

#### 5.4. Virtual Surface Creation

First step in the simulation procedure is, to decide on which CA cells are to be exposed to the etchant. The mask-wafer interface information is supplied by the mask editor, whereas the cells under or above this interface should be checked for determining the cells on the “virtual surface”. In order to store all the information harvested, a user-defined “VS” class was generated using Visual C++. The information that can be stored under this class is summarized in Table 5.2.

Table 5.2: VS Class Members

| <b>VS(Virtual Surface) Class</b> | <b>Type</b> | <b>Range</b>         | <b>Description</b>                       |
|----------------------------------|-------------|----------------------|--|
| Neg [4]                          | int         | -1,0,1               | Neighbor Information                     |
| NegVs [4]                        | int         | 0,1                  | Neighbors on the etchant-wafer interface |
| M                                | Float       | 0..1                 | State of the cell                        |
| Material                         | int         | 1..6                 | Material of the cell                     |
| Plane                            | int         | 1..4                 | Plane on which the cell lies             |
| x                                | Float       | 0..1500              | x location of the cell                   |
| y                                | Float       | 0..1500              | y location of the cell                   |
| z                                | Float       | 0..1500              | z location of the cell                   |
| tc                               | Float       | 0..1                 | Time compensation value                  |
| dc                               | Float       | 0..5x10 <sup>5</sup> | Doping Concentration                     |
| Selection                        | int         | 0,1                  | Whether the cell is on the VS or not     |

As mentioned before, there are four neighbors for each cell. The neighboring cell information is stored under the “neg[4]” and “neg\_vs[4]” variables on VS class. On these arrays, the three possible states of a neighbor are as follows:

- 1: There is a non-etched neighbor on the predefined location. The locations of the neighboring cells were given in Table 3.1. This cell lies on the virtual surface.
- 0: The neighbor cell on this location has been etched in previous steps or out of wafer range.
- -1: There is a cell on this location but it is not on the virtual surface, in other words it is not exposed to etchant.

To determine the virtual surface, the software first starts out with the information supplied by the mask editor. By checking the cells lying on the un-masked areas, the neighboring cells, which have the state “0”, are sought. These cells simply represent the holes and cavities formed in previous steps and the neighbors of these cells should be added to the virtual surface. When there is no more neighboring cells left with state “0”, the virtual surface creation is finished. This procedure is repeated before each time step, because there may be holes that are under the masked areas and not exposed to etchant in previous steps. Some undercut process may be observed around the perimeter of unmasked areas or some cavities might be bridged. As a result, consequently those cells will also be exposed to etchant in the next steps. Fig. 5.4 illustrates the flowchart of this process. Notice that in the flow chart; M, which is an element of VS class, denotes the state of a particular CA cell.

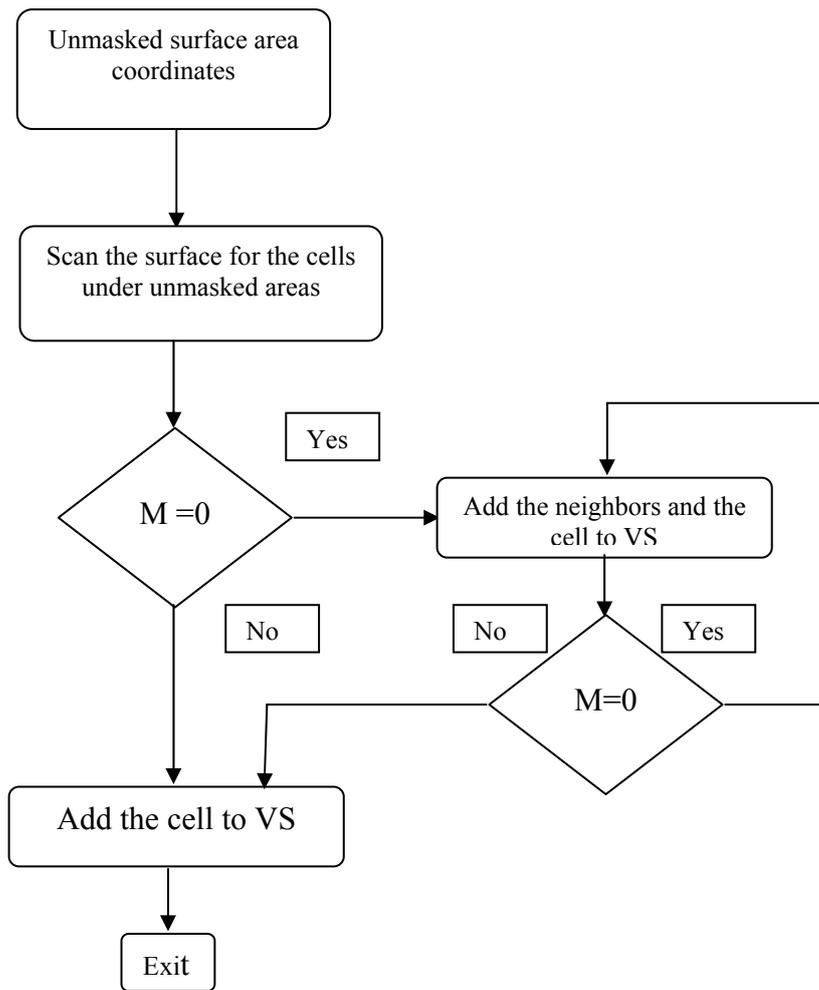


Figure 5.4: Virtual Surface Creation Flowchart

### 5.5. Plane of the Cells

After deciding on which cells should be processed in the next time step, to apply the etch algorithm, the plane of the cells should be determined. There is a function called `Plane()` on `VS` class, that was created for assessing the plane of the cells. Simply by finding the number of the neighboring cells and checking whether they lie on the virtual surface or not; the function finds the plane for each cell. The flowchart of the `Plane()` function is given in Fig. 5.5.

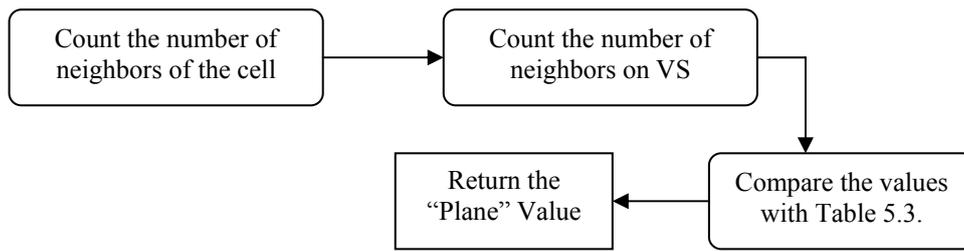


Figure 5.5: Plane Function Flow

Table 5.3: Number of Neighbors for Different Planes

|  | <100> | <110> | <111> |
|--|-------|-------|-------|
| Number of Neighbors                              | 2     | 3     | 3     |
| Number of Neighbors lying on the virtual surface | -     | 2     | -     |

Since each cell location and state information are stored in an array, simply scanning this array yields the information needed by the Plane() function.

## 5.6. Process Application

After the virtual surface is set and the planes of the cells are found, the cells undergo certain modifications based on the process selected.

### 5.6.1. Wet Etching Modifications

Using the etchant information obtained, each cell state is modified according to the etch rate of the chosen etchant for the plane of the cell. The initial states of the cells are “1” and when the state value reaches zero, the cell is removed. The main problem in applying the etch rate occurs if the state of the cell decreases below zero.

This situation is undesirable since, when the time passed between the state of the cell reaches zero, the final value of the state (which appears to be negative), is lost. To facilitate this time, the neighbors of the cell etched, will be exposed to

the etchant for the next step time (1 minute for MemsEagle) plus the lost time when the state of the etched cell dropped below zero. Note that, a similar approach entitled “time compensation” was known to be utilized by the developers of the ACES program.

After the etchant is applied to the cells, if the state of the cell is equal to zero, it is removed. The neighbors of the cell that do not lie on the virtual surface enter the etchant-wafer interface. The added cell inherits the conjugate of the type for cell being removed. (Type 1 $\leftrightarrow$ Type 2) The progress of the wet etching is illustrated in Fig. 5.6 where  $i$  refers to the time index and  $M$  denotes the state of a particular cell.

### **5.6.2. Dry Etching (Deep Reactive Ion Etch) Modifications**

Most of the steps of simulation are same as the wet etching for the dry etch process also. Since, only the cells in the un-masked areas should be removed, certain checks should be carried out before modifying the state of a cell.

First, during the creation of the mask possible cell locations on the un-masked areas are stored in an array. This is different than just storing the cell locations on the mask surface because the  $x$  and  $y$  locations of the cells under the surface of the wafer is varying. So all possible  $x$ ,  $y$  locations are stored in the memory for using in the doping, dry etching and additive processes. Fig. 5.7 illustrates the generation of the CA cell locations for different wafer types.

The steps used in wet etching are followed till the modification of the state values. Here, if the cell location does not match any of the locations stored in the array mentioned, the state value is not modified. Thus, only the cells that are under the un-masked areas are etched away. Addition of the neighbors is just the same as wet etching.

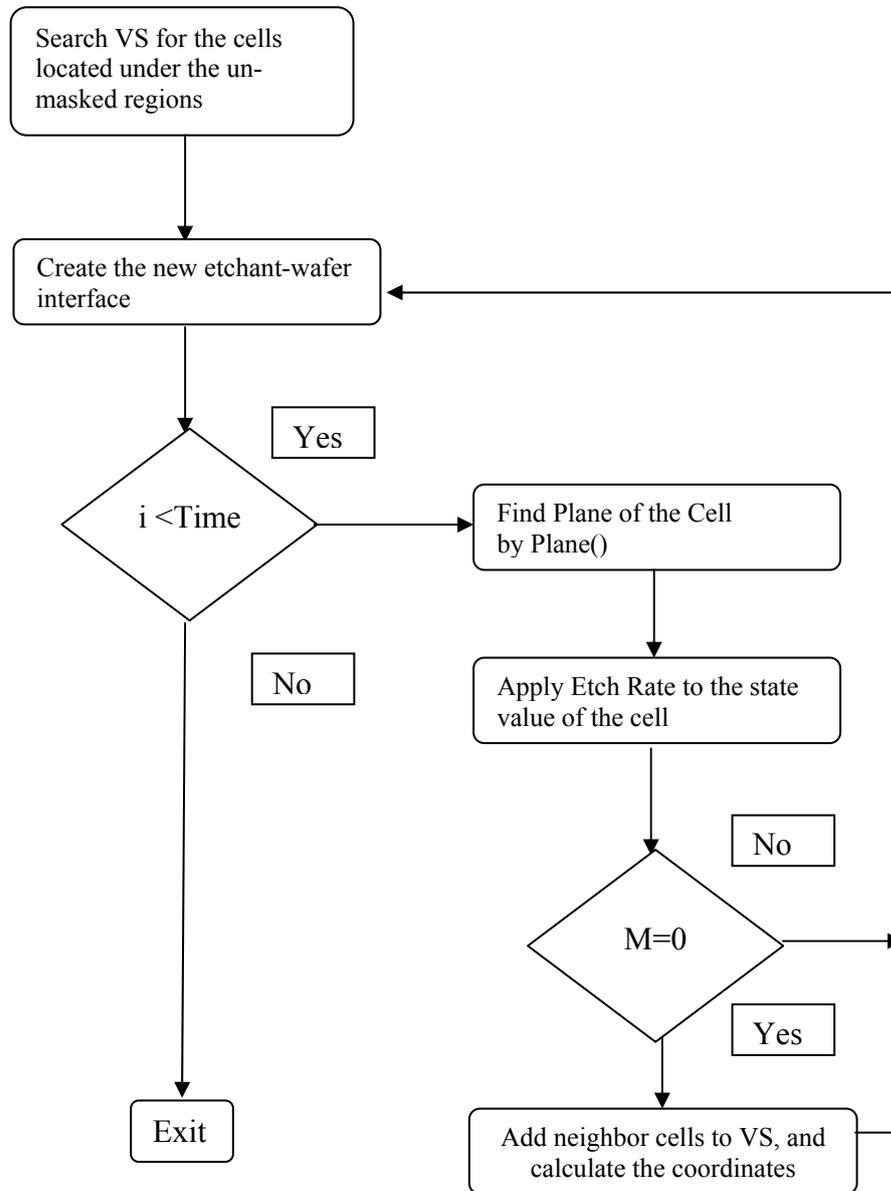


Figure 5.6: Wet Etching Flowchart

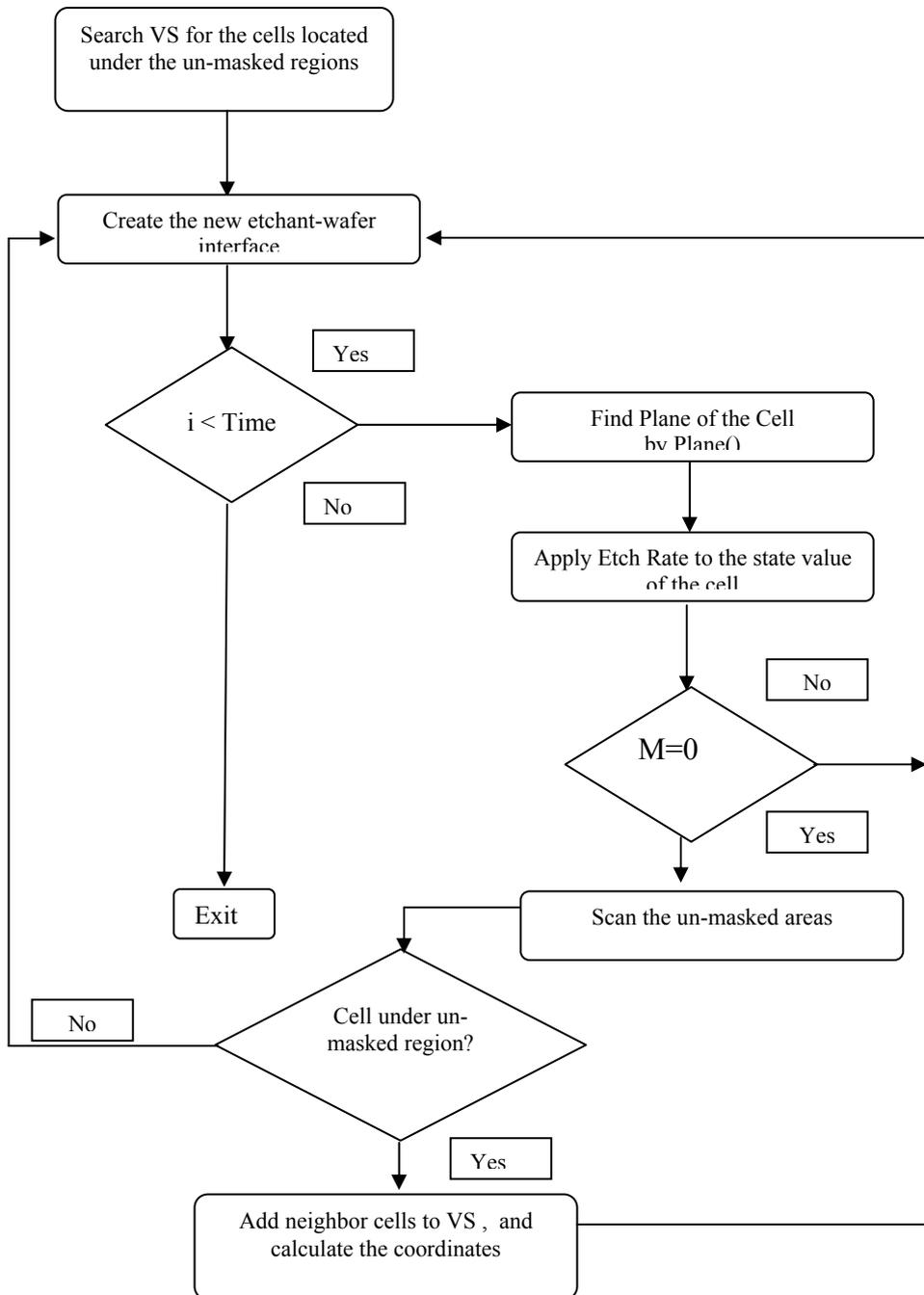


Figure 5.7: Dry Etching Flowchart

For example, for the rectangular mask in Fig.5.8, the cell locations are stored first. Notice that this is a (100) wafer flat on (100) direction. The (x,y)

coordinates of the surface cells and their neighboring cells are stored in the array generated. As illustrated in Fig.5.9, for each particular cell, the following cell locations should also be stored in the array if they are still in the un-masked areas:  $P_1(x+0.25, y+0.25)$ ,  $P_2(x-0.25, y-0.25)$ ,  $P_3(x+0.25, y-0.25)$ ,  $P_4(x-0.25, y+0.25)$ ,  $P_5(x+0.5, y)$ ,  $P_6(x-0.5, y)$ ,  $P_7(x, y+0.5)$ ,  $P_8(x, y-0.5)$ . By just comparing the new cells' x and y coordinates, the software decides whether the new cell located under the un-masked area and should be etched or not.

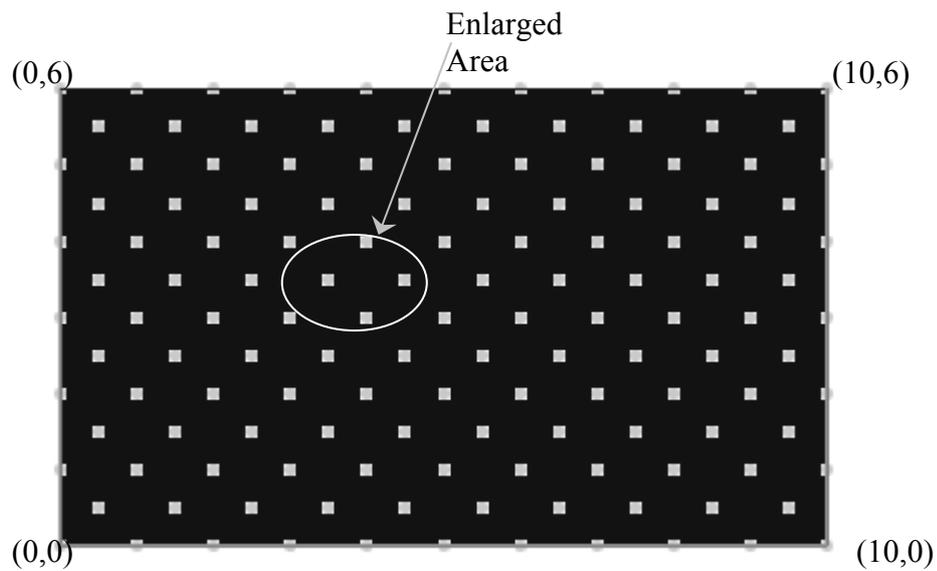


Figure 5.8: Dry Etch Modification

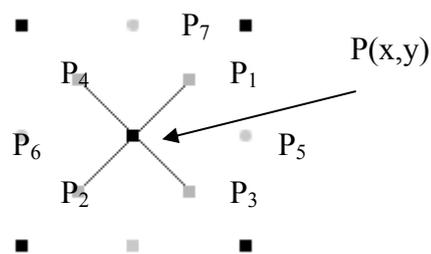


Figure 5.9: Enlarged Area

### 5.6.3. Doping Modifications

Doping is introduction of impurities into the silicon wafer, thus in theory the silicon cell lattice has to be modified to accommodate the extra atoms diffused to

the silicon crystal. In MemsEagle, doping is simulated by modifying the doping concentration property associated with a particular cell without ever changing the cell array.

Another assumption inherently used in the simulation is that: only the unmasked areas of the wafer are doped. The doping does not continue in the lateral directions covering the mask portions of the wafer. Within this framework, doping simulation in MemsEagle is the one used to slow out the etching process.

The “doping concentration” values of each cell is modified during the doping process. Just as the dry etch process, only the un-masked areas are modified. Using the drive-in and pre-deposition times entered by the user, the code finds out the penetration of the impurities using the following expressions.

$$Q = 1.13N_0\sqrt{D_1t_1} \quad (5.1)$$

$$N(z,t) = \left[ \frac{Q}{\pi Dt} \right] e^{-\frac{z^2}{4Dt}} \quad (5.2)$$

where  $Q[\text{cm}^{-2}]$  is the total impurity dose,  $D_1[\text{cm}^2/\text{s}]$  is pre-deposition coefficient and  $t_1[\text{s}]$  is pre-deposition time;  $N(z,t)[\text{cm}^{-3}]$  is the concentration profile,  $D[\text{cm}^2/\text{s}]$  is diffusion coefficient,  $t[\text{s}]$  is diffusion time and  $z[\mu\text{m}]$  is the depth from the surface. The temperature of the doping process simulated is  $1200^\circ\text{C}$ .

When a new cell is added to the virtual surface, its x, y coordinates are checked and if it matches with the un-masked region used during the doping, the doping value of the cell is modified during the creation phase.

#### **5.6.4. Additive Process Modifications**

Final micro-fabrication step that can be simulated in MemsEagle is the additive processes. MemsEagle just displays the resultant material layer deposited based on the number of layers entered by the user.

The steps followed for finding the areas on which the material is to be deposited are similar to the DRIE process. The software scans the unmasked areas and finds out the upper-most CA cells on those regions. After that, if the state value of the cell is not equal to 0, its neighbors located in the upper side of the wafer are added to the virtual surface. As an example, for wafer type (100) flat on (110), neighbors 1 and 3 should be added for type 1 cells whereas neighbors 2 and 4 should be added for type 2 cells. Similar to DRIE, the new cells should be checked whether they are located under the un-masked areas or not. The flowchart for this process is shown in Fig.5.10.

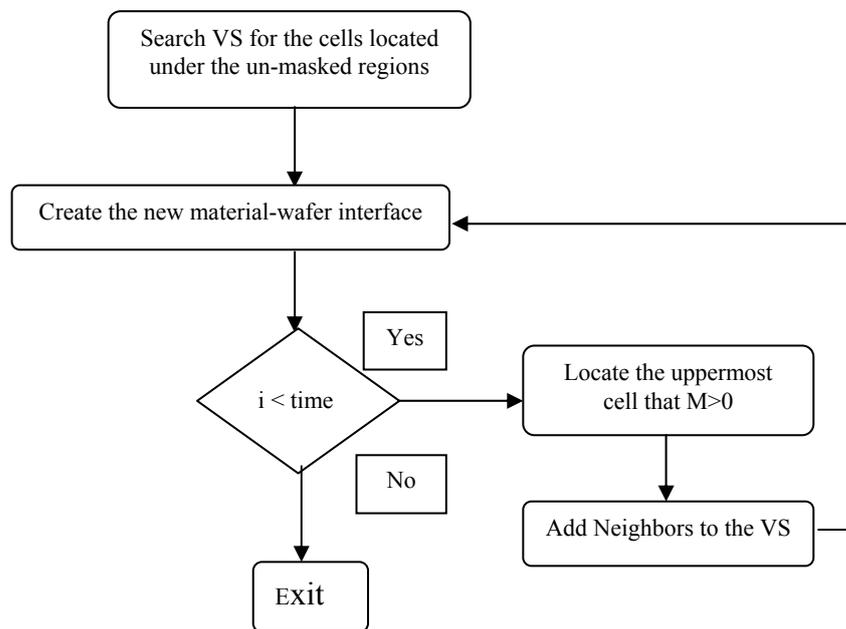


Figure 5.10: Additive Process Flowchart

### 5.7. Visualization of data

After the process is finished the plane, coordinates, state, doping concentration and material information is stored in “VS\_Draw” array. This information is later used with OpenGL functions for displaying the simulation results. Displaying the process results are done by drawing the cells on the screen with respect to their material, plane type or doping concentration. The simple OpenGL sequence:

```
glBegin(GL_POINT);  
    glVertex3f(x0, y0, z0);  
glEnd();
```

where  $(x_0, y_0, z_0)$  represents the cell location in space is used for drawing the cells on the screen. Because of the perspective view used, the cells and lines nearer to the screen are drawn bigger than the ones located deeper into the screen. Using OpenGL functions the size and shape of the points can be modified for easy viewing of the surfaces generated by the process. Table 5.4 shows the color map used for representing different materials, planes and doping concentrations in Memseagle. Notice that, a simulation result utilizing the plane view is illustrated in Fig. 5.11.

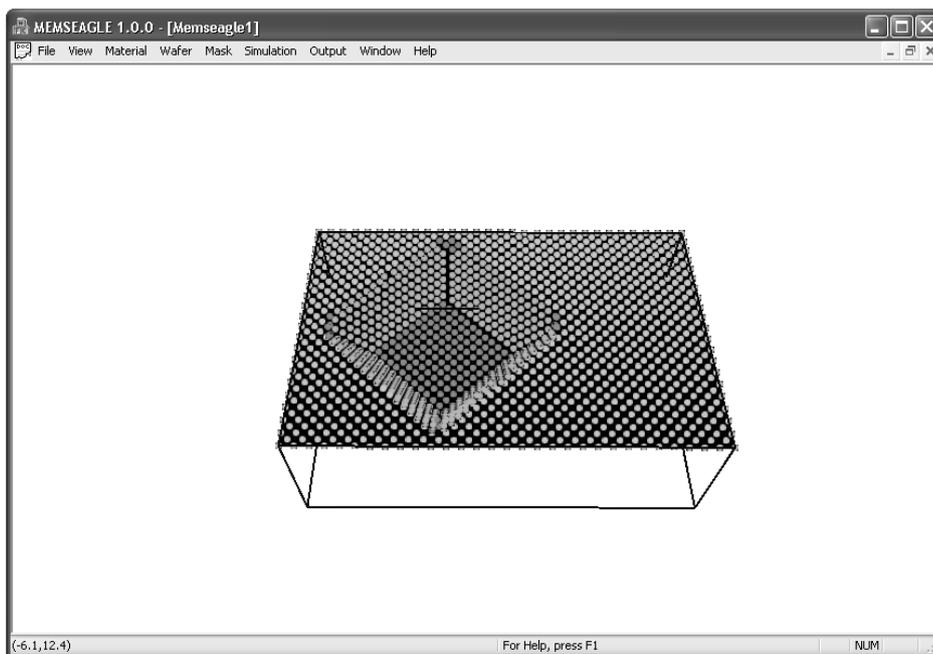
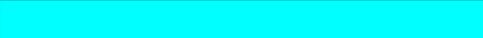


Figure 5.11: Displaying Results

Table 5.4: Color Map in MemEagle

|               | Type       | Colors   |
|---------------|------------|--|
| Plane View    | (100)      |  |
|               | (110)      |  |
|               | (111)      |  |
|               | (311)      |  |
| Material View | Si         |  |
|               | SiO2       |  |
|               | Si3Ni4     |  |
|               | Al         |  |
|               | Poly Si    |  |
|               | Gold       |  |
| Doping        | $>10^{15}$ |  |
|               | $>10^{16}$ |  |
|               | $>10^{17}$ |  |
|               | $>10^{18}$ |  |
|               | $>10^{19}$ |  |

### 5.7.1. Viewing Transformations

A perspective view is used for displaying the results in MemEagle. The OpenGL function `gluPerspective(angle, aspect_ratio, znear, zfar)` is utilized, where the first argument is the field of view angle, the third and fourth arguments are the distance from the viewer to the near clipping plane (always positive) and the distance from the viewer to the far clipping plane (always positive) respectively, is used for defining the view. The position of the viewer is set by the OpenGL function `gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)` where `eyex, eyey, eyez` are the position of the eye point; `centerx, centery, centerz` are the position of the reference point and `upx, upy, upz` are the direction of the up vector. Figures 5.12 to 5.14 illustrate views of a micro-fabrication process from different angles, locations and scaling.

There are three viewing transformations used in MemEagle: move, rotate and zoom. Moving is done by transforming the objects drawn in the x, y plane by clicking and dragging the left mouse button. The OpenGL function `glTranslatef(x_value, y_value)` is used for moving.

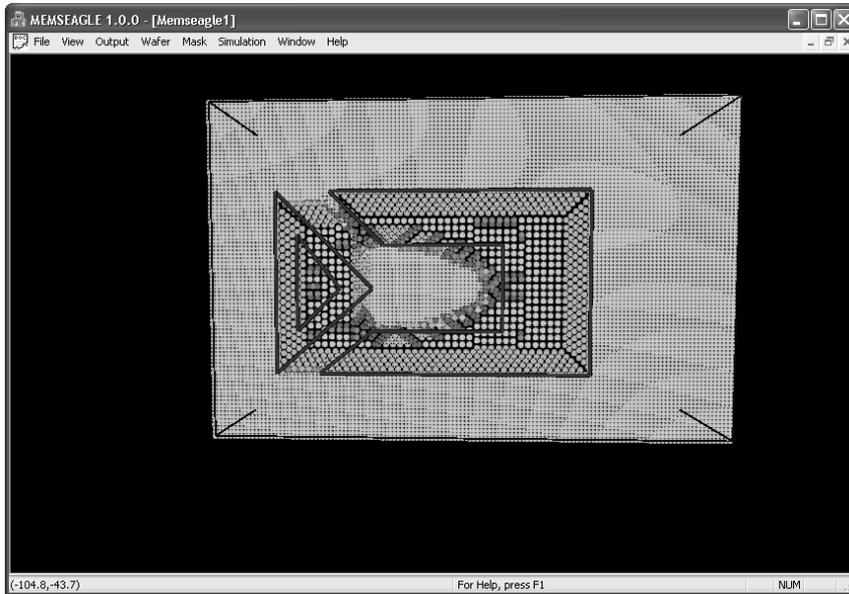


Figure 5.12: Top View of the etch result

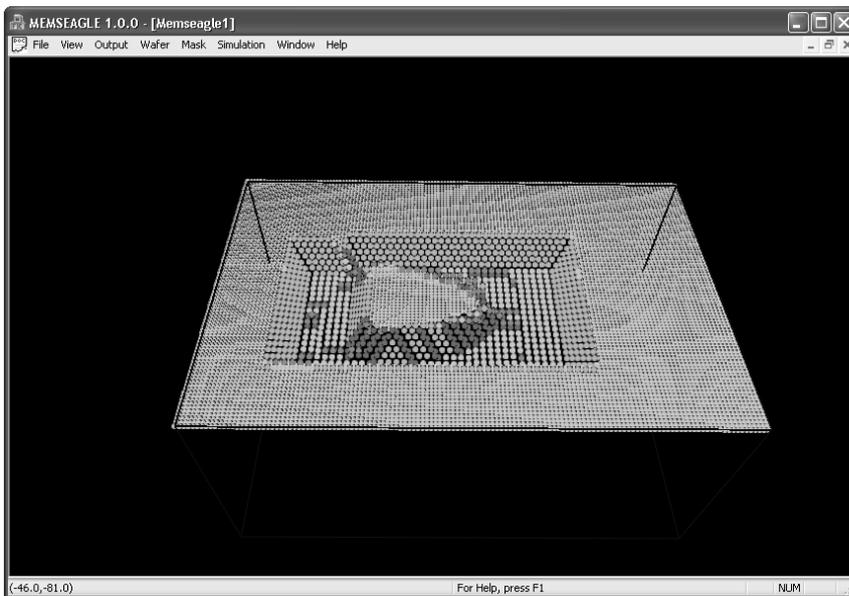


Figure 5.13: Etch Result without mask from a different angle

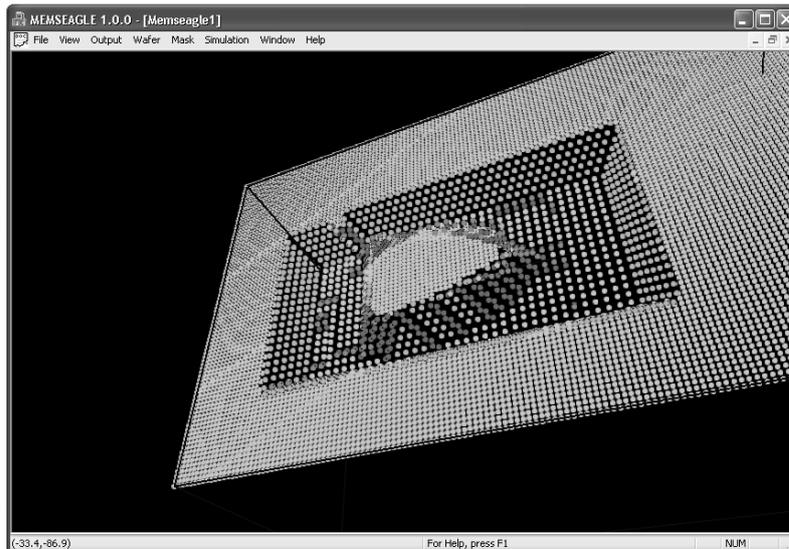


Figure 5.14: Etch Result From different angle and scaling

Take note that, rotation is carried out by the function `glRotatef(angle, x, y, z)` where angle is the angle of rotation and  $x, y, z$  are defining the direction of the vector from the origin to be used for rotating the objects around. The zoom option is done by the help of the function `glScalef(x, y, z)` where  $x, y$  and  $z$  are scale factors along the  $x, y$ , and  $z$  axes, respectively.

## 5.8. Closure

Process editor is the heart of the MemsEagle software. All the information gathered is used for simulation of the micro-fabrication processes through this editor, and the functions under it. Wet and dry etching processes, doping and other additive operations can be simulated in an integrated environment. All the information obtained through process editor is sent to OpenGL function "OnDraw" for displaying the results.

## CHAPTER 6

### PROGRAM FEATURES

#### 6.1. Introduction

The mechanics of the wafer creation, mask generation and process simulation was explained in the previous chapters. This chapter is dedicated to the user-interface of MemEagle and how this mechanism are activated using the menu and dialog controls.

#### 6.2. MemEagle User Interface

MemEagle is a menu-driven MFC (Microsoft Foundation Class) software, using OpenGL functions and dialog bars for simulating micro-fabrication processes and displaying them. Notice that, the three editors previously discussed can be accessed through the menu commands in the program. Fig. 6.1 displays the interface of MemEagle, and the wafer, mask and simulation menus can be seen which are used for reaching those editors.

In the preceding sections, the editors are explained using a micro-fabrication example. Notice that, there is an “Output” menu shown in Fig. 6.1, which was not mentioned up to now. Through this menu, the distance between two points, point coordinates; depth and doping concentration of particular points can be found.

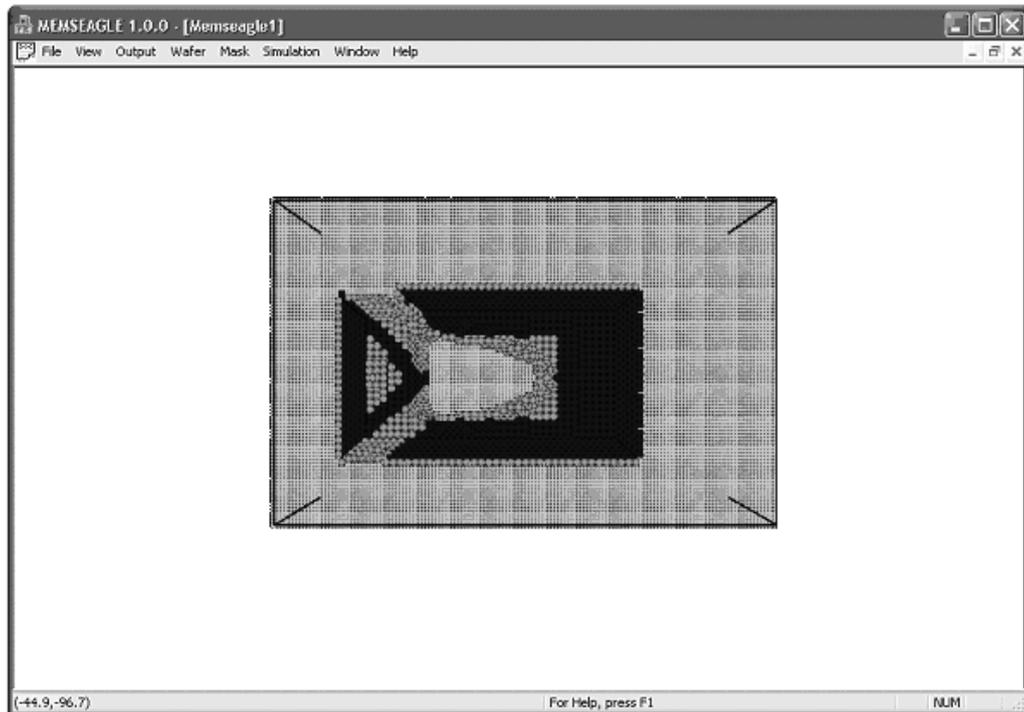


Figure 6.1: MemsEagle Interface

### 6.3. Wafer Editor

The editor was accessed through the “Wafer” menu and from the “Wafer Editor...” menu item. Using the editor dialog box, as illustrated in Fig 6.2, the user can choose, the wafer crystal orientation, doping type and concentration and the wafer size.

As mentioned earlier, the substrate was generated as a wire frame except the upper surface. This surface is created as a mask covering the entire area, and from this upper surface the micro-fabrication processes is to be initiated. A sample wafer was generated and illustrated in Fig. 6.2.

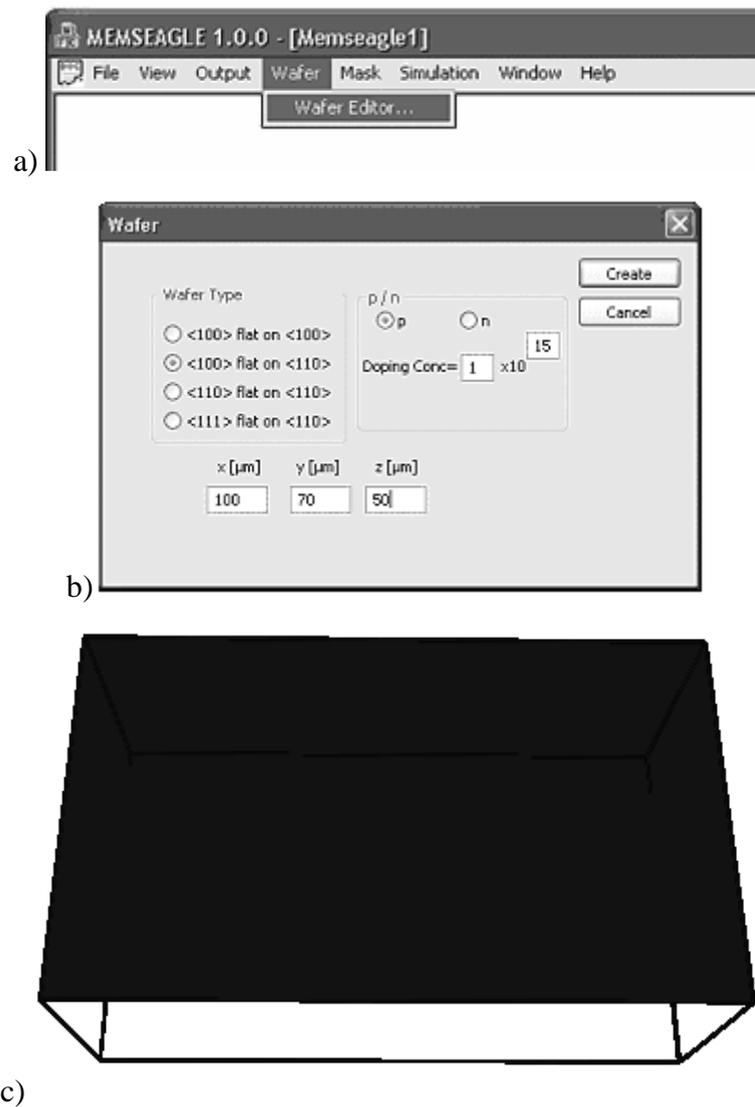


Figure 6.2: Wafer Editor

#### 6.4. Mask Editor

Mask editor was reached from the mask menu similar to the wafer editor. When the mask editor is entered, just the upper surface of the substrate is drawn on the screen. This procedure is illustrated in Fig. 6.3.

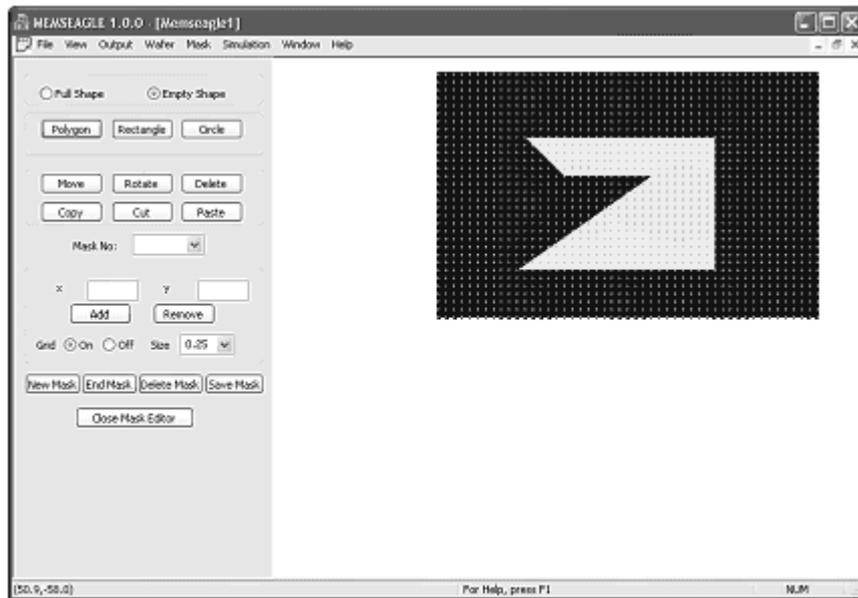
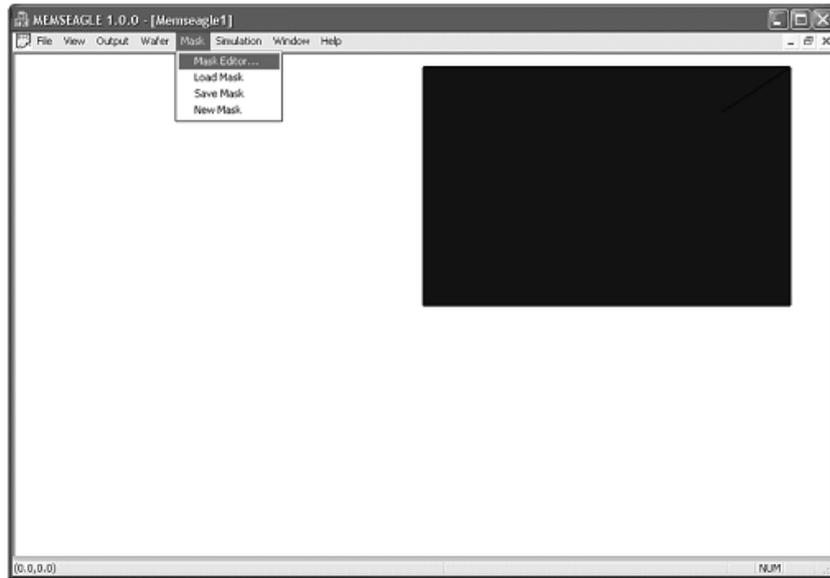


Figure 6.3: Mask Editor

In the example, a polygon is initiated in grid mode. As seen in Fig 6.4 the grid and empty radio buttons are selected. Using the left mouse button alone, the pattern can be generated easily with the help of the coordinate display. The last shape was created selecting the full shape button so a masked area created inside the triangle.

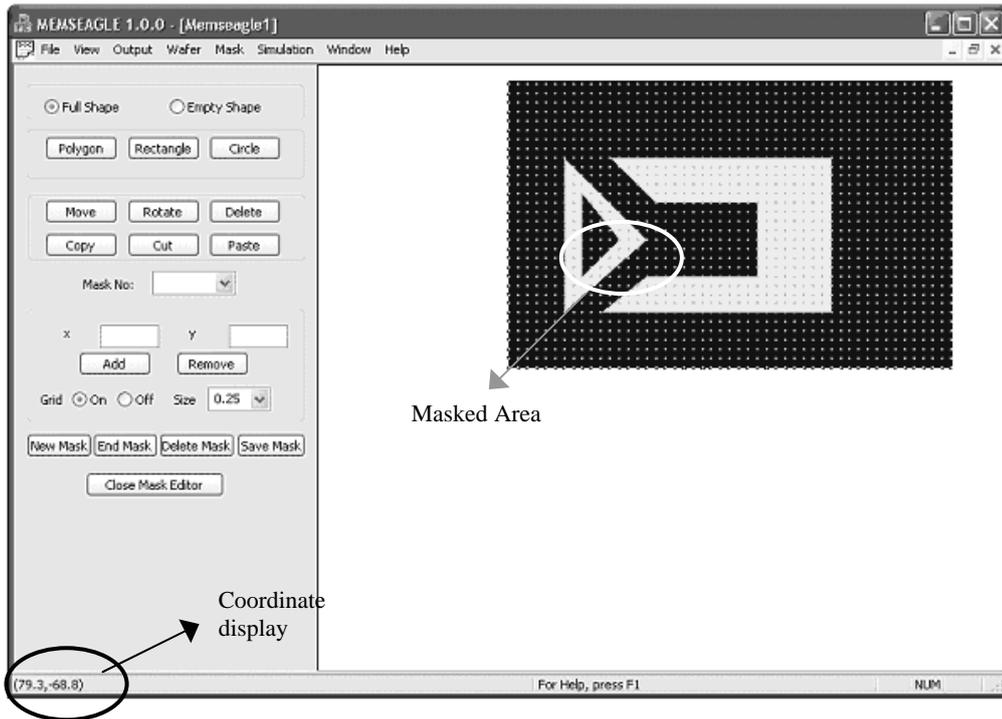


Figure 6.4: Drawing Mask

When creation of a particular mask is finished, the grid must be turned-off and the “end” command button must be clicked. By pressing the “end” button, the user activates the scanning process, which generates the necessary mask information for the process editor.

### 6.5. Process Editor

When the process editor was initialized via the simulation menu, a dialog box is appears on the screen as illustrated in Fig. 6.5. The user is prompted to select the micro-fabrication type and the mask number from the combo boxes. As shown in this example, the user has selected a wet etching process and the corresponding mask.

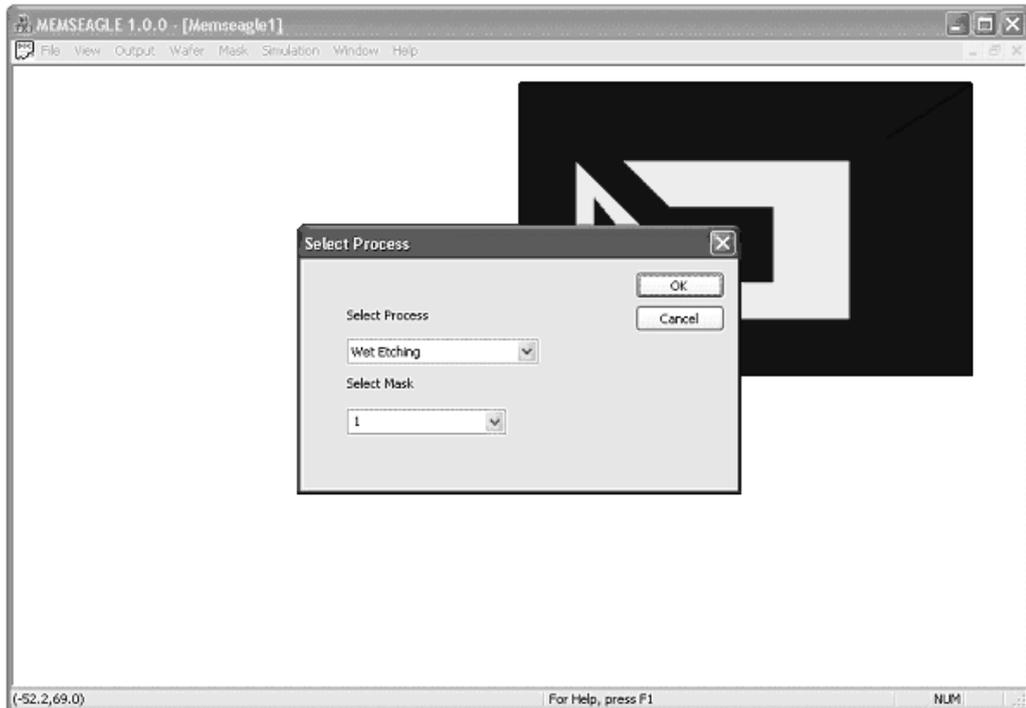


Figure 6.5: Process Selection

The next step is the selection of the etchant and the process time. Note that, in MemsEagle minimum process time is 1 minute. When the selections are complete, the user must enter the “OK” command button to store the information in the memory. As shown in Fig. 6.6, the user has chosen KOH with a concentration of 30% at a constant temperature of 70°C as the etchant, and the simulation time is selected as 10min.

MemsEagle provides not only a stand-alone process simulation but also projects composed of a large number of sequential processes. In order to create such a project, the project editor should be used. As explained in Chapter 5, the micro-fabrication steps and the masks can be defined using this editor. Hence, the whole project can be simulated in one-step. The interface used for the project editor is illustrated in Fig. 6.7. As can be seen, all the relevant information entered can be seen from a list control provided.

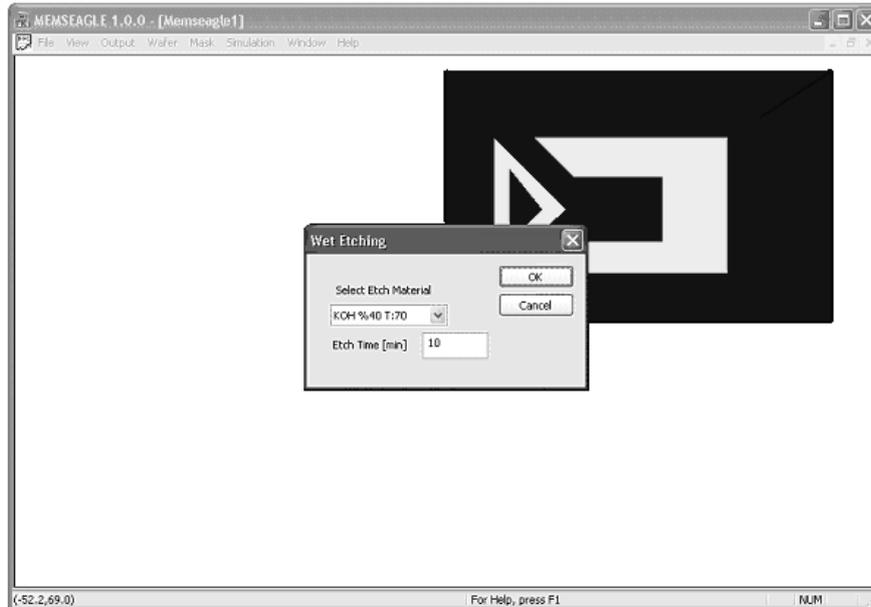


Figure 6.6: Wet Etching Dialog

The result of the simulation process defined is displayed in Fig. 6.8. Once the results are obtained, the next step should be harvesting the information. This can be done by using the output menu or by visual inspection with the help of viewport modifications.

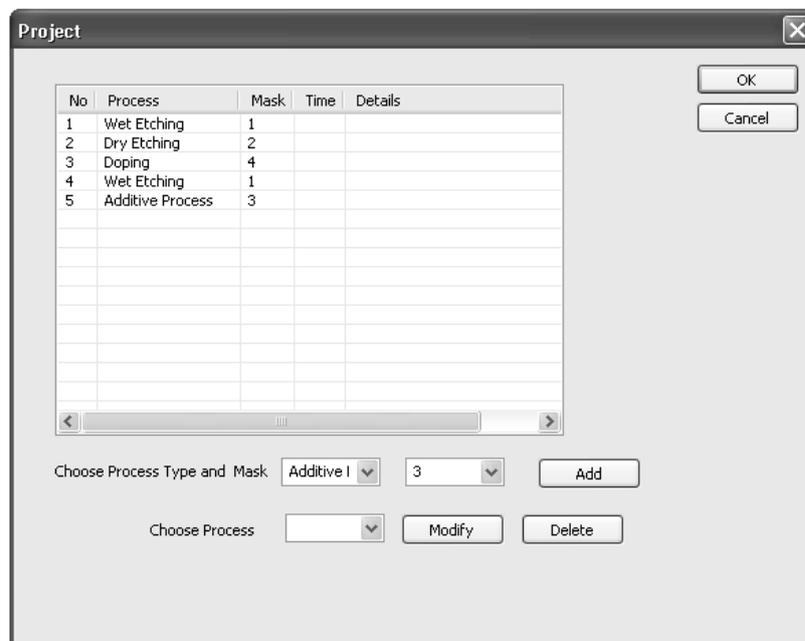


Figure 6.7: Project Editor

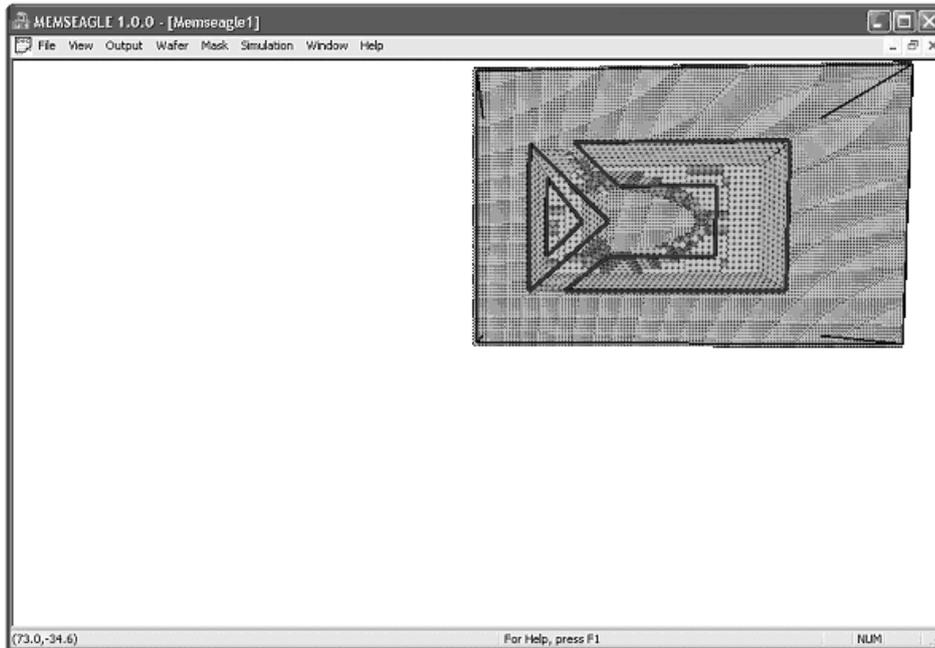


Figure 6.8: Simulation Result

## 6.6. Viewing

As the simulation of the micro-fabrication process is finished, the resulting shape is displayed on the screen, as the CA cells with colors represents not only different planes but also materials with different doping concentrations. Through the view menu, the display options can be entered as illustrated in Fig. 6.9. The user is able to do the following actions:

- Rotate, zoom, move the object
- Display/hide the un-etched cells
- Show/hide the mask
- Change the coloring scheme for gathering particular information on the simulation results.

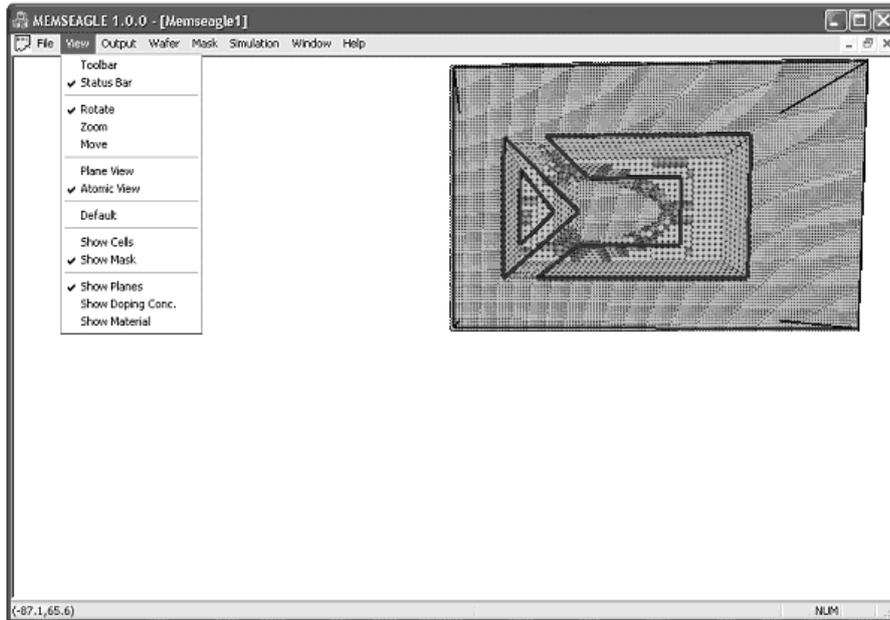


Figure 6.9: View Menu

Different views of the result have already been shown through Figs 6.11 to 6.13. Here color scheme for the doping concentration is illustrated by Fig. 6.10. Note that the dark colored areas are the heavily doped regions. (The substrate was doped before) Now that the visual inspection is done on the results, the next step should be taking measurements on the simulated design.

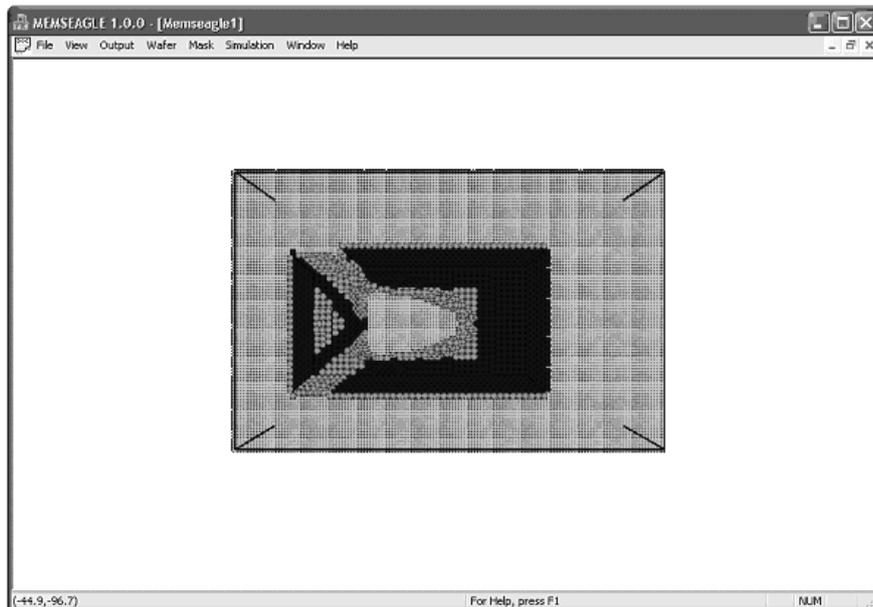


Figure 6.10: Viewing Doping Concentration

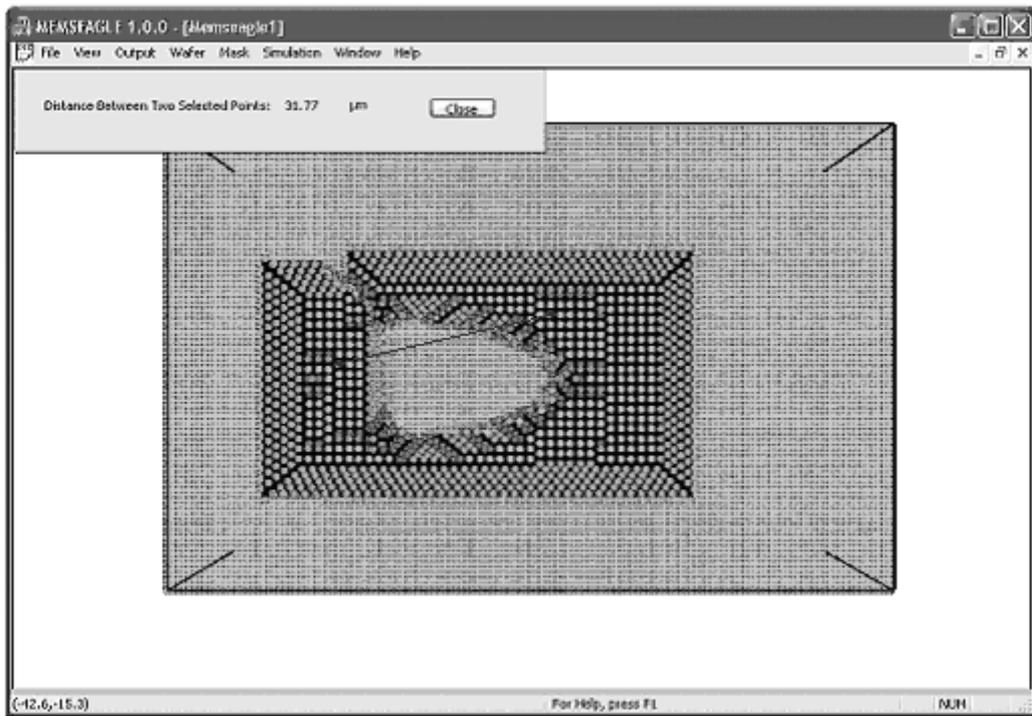


Figure 6.11: Distance between two points

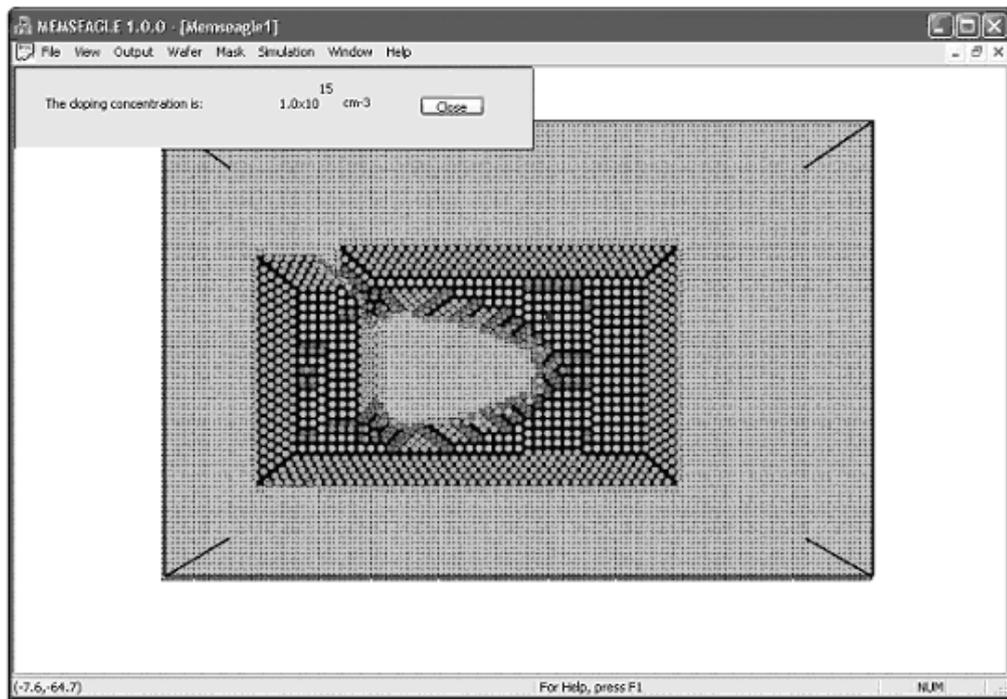


Figure 6.12: Viewing doping concentration of a point

## 6.7. Taking Measurements

After the output has been generated; location, depth, doping concentration and plane information of the generated CA cells can be accessed via the output menu. In addition, distance between two points can be determined. The selection algorithm used was the same as for the mask modifications and explained in detail in Chapter 4. By selecting the desired information and choosing the point via mouse, the information sought is displayed on the screen as shown in Figures 6.11 to 6.13.

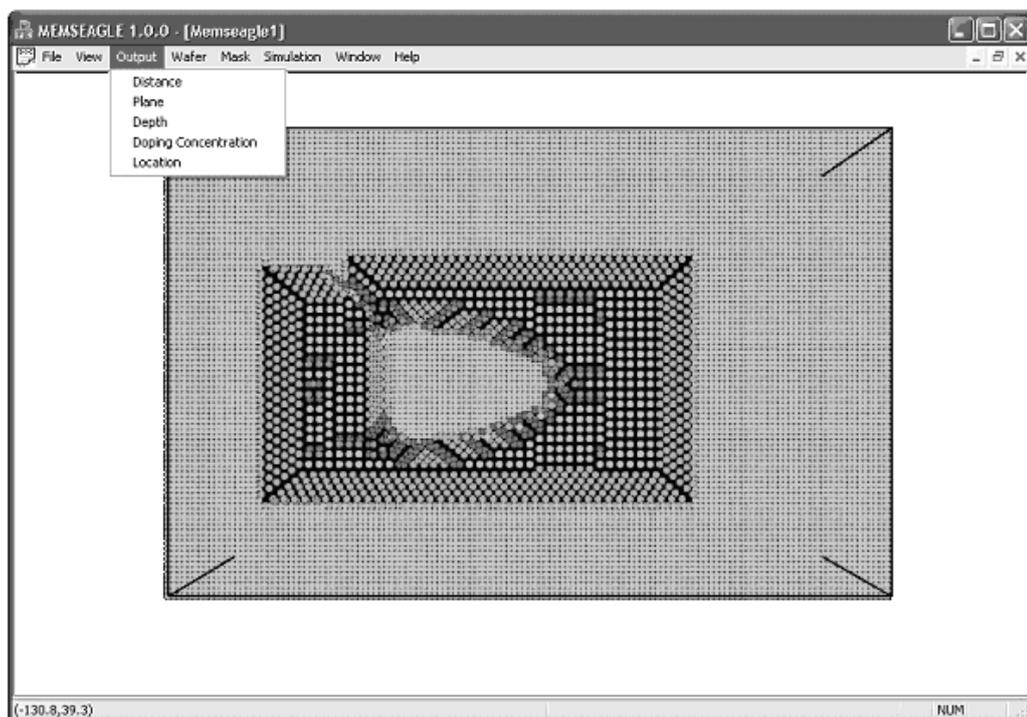


Figure 6.13: Output Menu

## 6.8. Closure

The user interface of Memseagle has been explained in detail. The capabilities of the software and the information that can be accessed through different menus and dialog boxes have been shown. By making use of a wet etching example, a simple simulation scheme was elaborated.

## CHAPTER 7

### VERIFICATION OF SIMULATION RESULTS

#### 7.1. Introduction

To verify the validity for the output of the developed program, the simulated MEMS fabrication processes have to be compared to those of the real cases. Hence, the object of this chapter is to study the performance of MemsEagle through a bunch of real-world cases.

#### 7.2. Simulation Results

To assess the validity of the simulated results given by MemsEagle, four simulations are conducted. The simulated cases are as follows:

1. Anisotropic wet etching of complex shapes on silicon wafer
2. Deep Reactive Ion Etch of high aspect ratio structures on silicon wafer
3. Creation of complex structures via doping
4. Surface micromachining

Details about these simulations follow:

##### 7.2.1. Anisotropic Wet Etching

In this simulation, the  $\langle 100 \rangle$  silicon wafer which is flat on  $\langle 110 \rangle$  plane is used as the substrate and the etchant selected is 30% KOH at 80°C. Notice that, the cases to be studied are directly taken from the manual of SIMODE [24]. These cases are essentially used to confirm the results given by the SIMODE code. By simply comparing the simulation results to the features of the actual MEMS structures being fabricated for this purpose. Therefore the masks designed for SIMODE verification are directly utilized in MemsEagle. The creation of these

masks is also explained in mask editor section. The process results at different time steps are also shown in the preceding sections. The simulated wet etching cases are as follows:

1. Etching a rectangular pattern in  $\langle 100 \rangle$  wafers flat on  $\langle 100 \rangle$
2. Etching a rectangular pattern in  $\langle 100 \rangle$  wafers flat on  $\langle 110 \rangle$
3. Etching co-centered rings in  $\langle 100 \rangle$  wafers flat on  $\langle 110 \rangle$
4. Wet etching of diamond shape in  $\langle 100 \rangle$  wafers flat on  $\langle 110 \rangle$
5. Etching of a paddle shape
6. Etching of fingers
7. Etching simulation of a “Tee” shaped pattern
8. Wet etching of complex shapes with various mask (mis)alignment angles

These cases are elaborated sequentially in the following sections.

#### **7.2.1.1. Case 1 for wet etching**

In order to test the etching simulation capabilities of MemsEagle for of  $\langle 100 \rangle$  wafers flat on  $\langle 100 \rangle$ , a simple rectangular mask pattern is used first. EDP is employed as the etchant. After 30 minutes, the resulting shape consisting of (111) planes starting from the (110) planes, which are inclined by  $45^\circ$ , appears as expected. Because the etch rate of (111) planes is slower ( $0.028\mu\text{m}/\text{min}$ ) compared to (100) and (110) planes, where the corresponding etching rates of these planes are  $0.047\mu\text{m}/\text{min}$  and  $0.28\mu\text{m}/\text{min}$  [13] respectively, etching tends to slow down to a near stop at (111) planes. Figure 7.1 confirms this observation. If the process is continued, the (111) planes will merge at the bottom and a pyramid whose base is aligned at (110) planes is formed as shown in Fig. 7.2. Similarly, Fig. 7.3 shows the fabrication of this textbook structure, which in turn verifies the results of the MemsEagle.

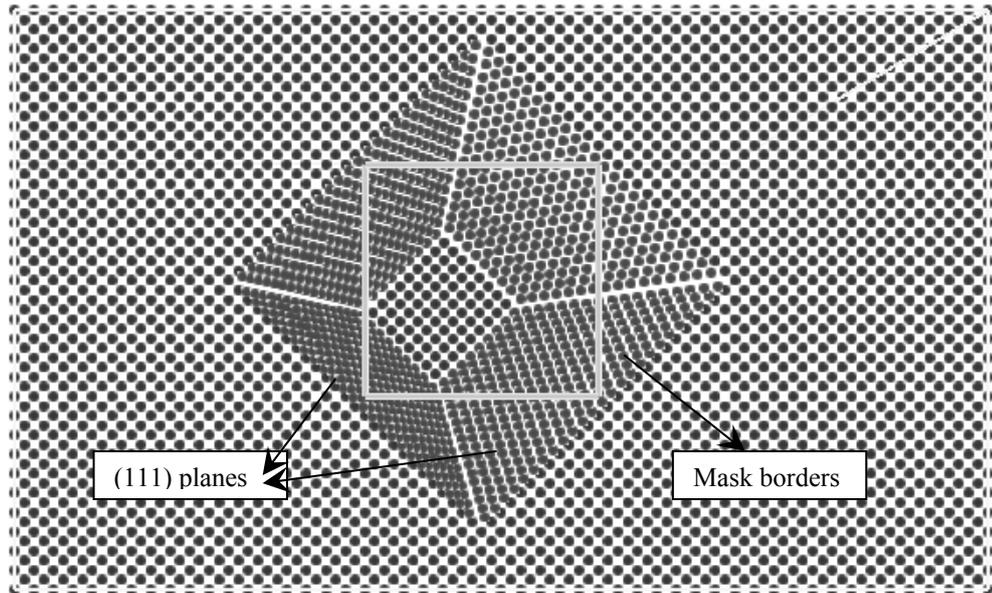


Figure 7.1: EDP Etch View

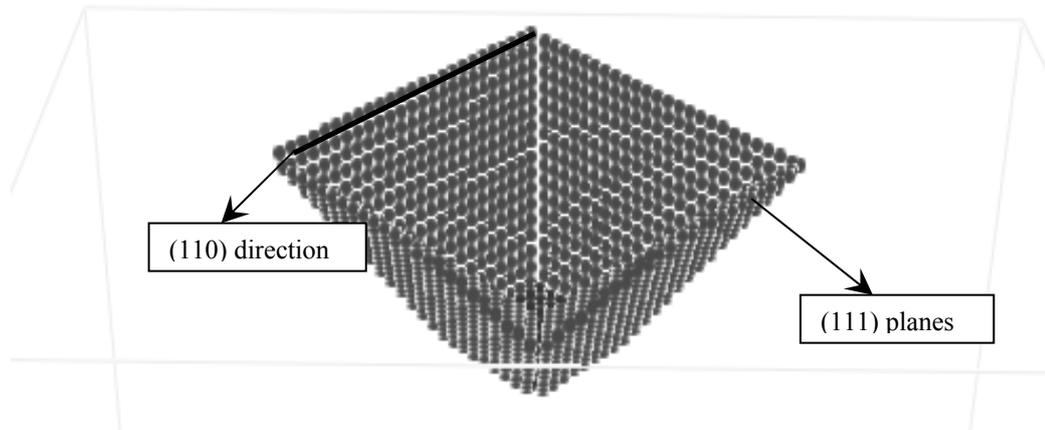


Figure 7.2: EDP Etch 3-D View



Figure 7.3: EDP Etch of (100) Silicon Wafer [31]

### 7.2.1.2. Case 2 for wet etching

As discussed in the previous case, the etching of (100) wafers stops at lines parallel to (110) planes. In order to show this, and to check the accuracy of the software for <100> wafers flat on <110>; etching of a rectangular mask pattern on such a wafer has to be studied. The resultant etch profile in agreement with the experimental results. Figures 7.4 and 7.5 illustrate the formation of (111) planes and the termination of etching through (110) directions.

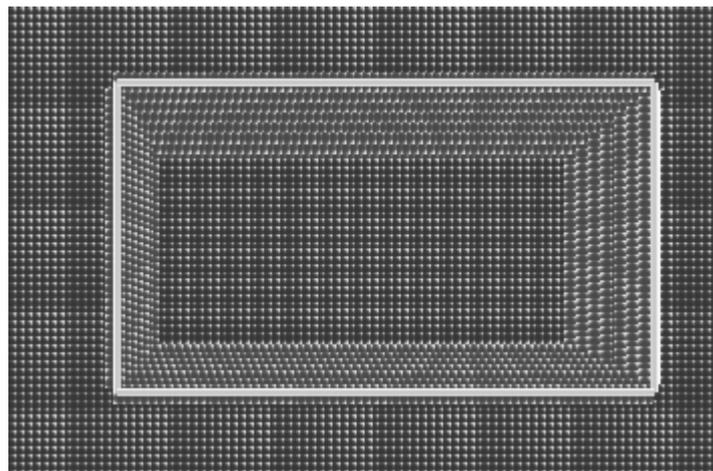


Figure 7.4: Etch profile of <100> wafer flat on <110>

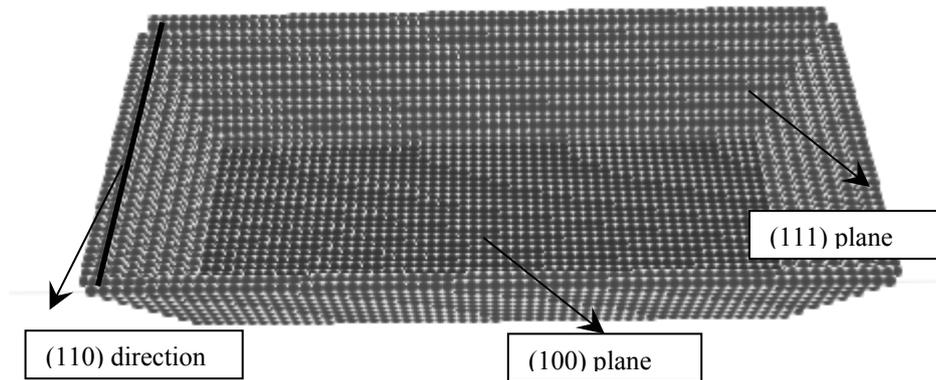


Figure 7.5: 3-D Etch profile of  $\langle 100 \rangle$  wafer flat on  $\langle 110 \rangle$

The actual shape formed through such a fabrication scheme is very similar to Fig. 7.3 and is well studied in literature [4], [24]. Therefore, these simulation results will not be discussed any further.

### 7.2.1.3. Case 3 of wet etching

As a more complex case, co-centered circles are taken into consideration. The resulting mask, which is shown in Fig 7.6, is created using the mask editor of MemsEagle. Due to the geometric restrictions imposed by the mask editor, the circular mask patterns have a stepwise circular pattern correlated with the resulting resolution of the mask editor. The wafer is exposed to KOH (at  $80^{\circ}\text{C}$ ) for 100 minutes and the resultant shape (only one quadrant) was illustrated in Fig 7.7. Similarly, the actual fabrication result is demonstrated in Fig 7.8. Not surprisingly, the simulation result and the actual one agree well.

Notice that, the (100) plane formed at region 1 shown in Fig 7.7, has a measured depth of  $105\mu\text{m}$  whereas the result of the experiment shows  $109\mu\text{m}$ . As expected (111) planes formed at regions 2 and 3, has an inclination angle of  $54.7^{\circ}$ . Both the simulation results and the experimental ones show that the (111) planes

merge each other at right angles, and form the base of a rectangular pyramid as the fabrication continues.

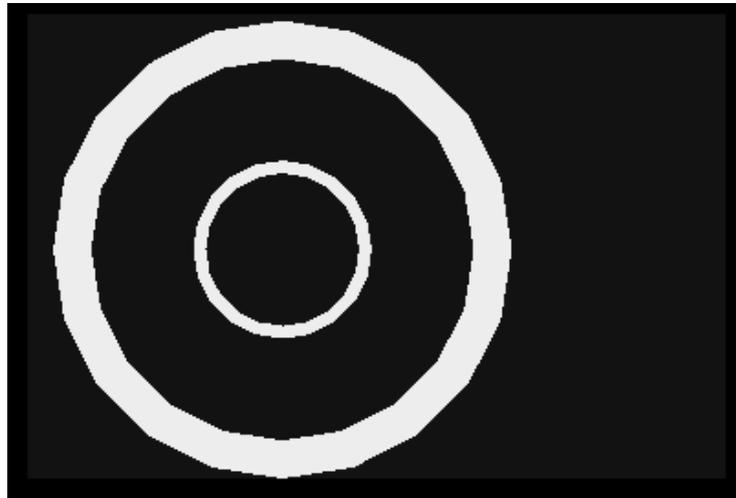


Figure 7.6: Co-centered Circular Mask Pattern

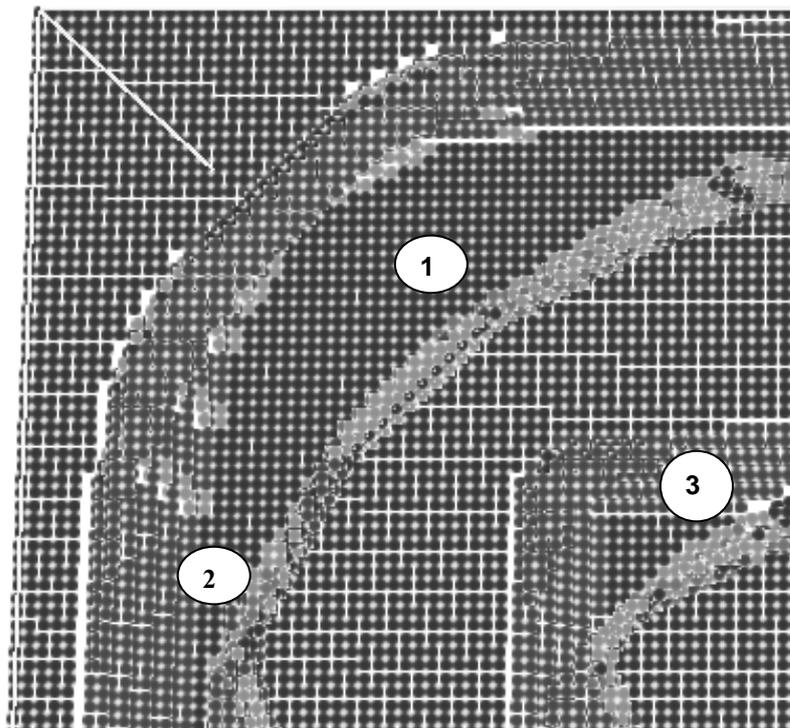


Figure 7.7: MemsEagle Result for Co-centered Circular Mask Pattern

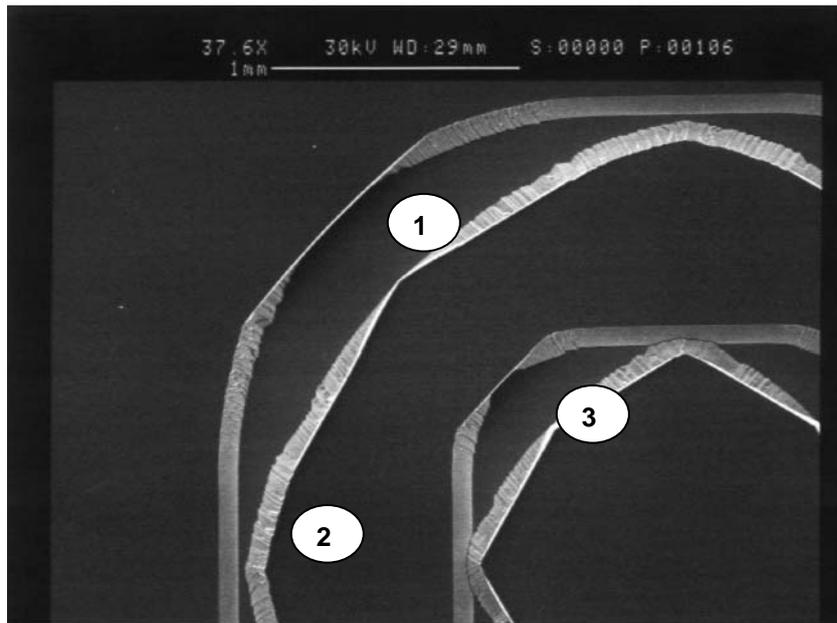


Figure 7.8: Experiment Result for Co-centered Circular Mask Pattern [24]

#### 7.2.1.4. Case 4 of Wet Etching

The next case concentrates on a much-sophisticated shape, which is an array of rectangular patterns as shown in Fig.7.9. A  $\langle 100 \rangle$  wafer is used as the substrate, which is flat on  $\langle 110 \rangle$  plane. The etchant was KOH 30% (at  $80^{\circ}\text{C}$ ) and the etching time is set to be 150 minutes. The simulation result and the experiment result after 100 minutes were displayed in Fig. 7.10 and Fig. 7.11. Since, the pits formed under the unmasked square shapes have not merged yet, the results are very similar to those obtained in article 7.2.1.2.

After 150 minutes, these shapes are beginning to merge and a situation, which creates a difficult problem faced during the simulation of micro-fabrication processes by the geometric methods discussed in Chapter 2, is observed. As shown on Fig. 7.12, the simulation result of MemsEagle agrees well with the experimental result shown in Fig. 7.13.



Figure 7.9: Mask Pattern of Merging Shapes

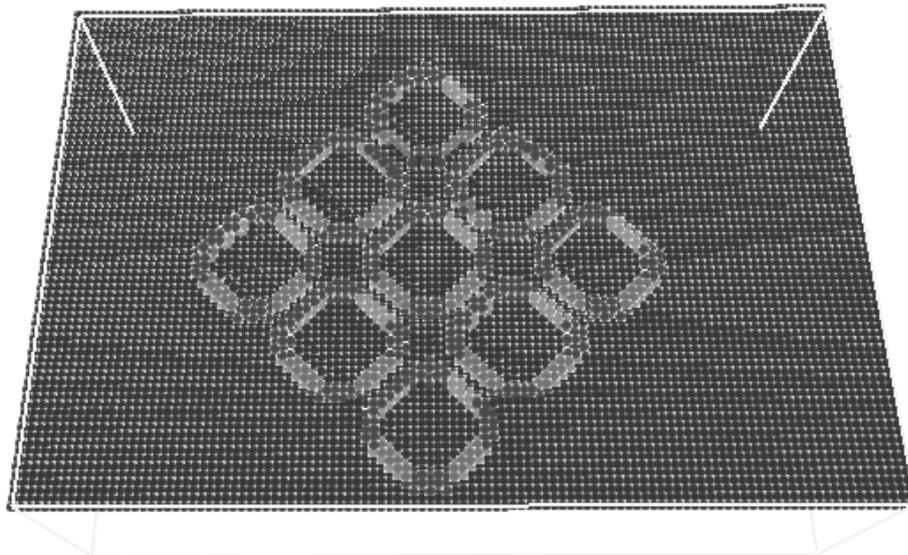


Figure 7.10: MemsEagle Simulation Result of the mask (Fig. 7.9)

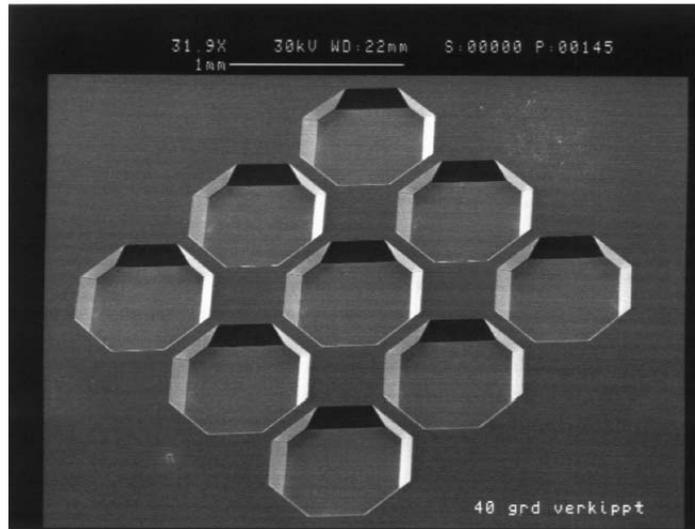


Figure 7.11: Experimental Result for the Mask (Fig. 7.9)[24]

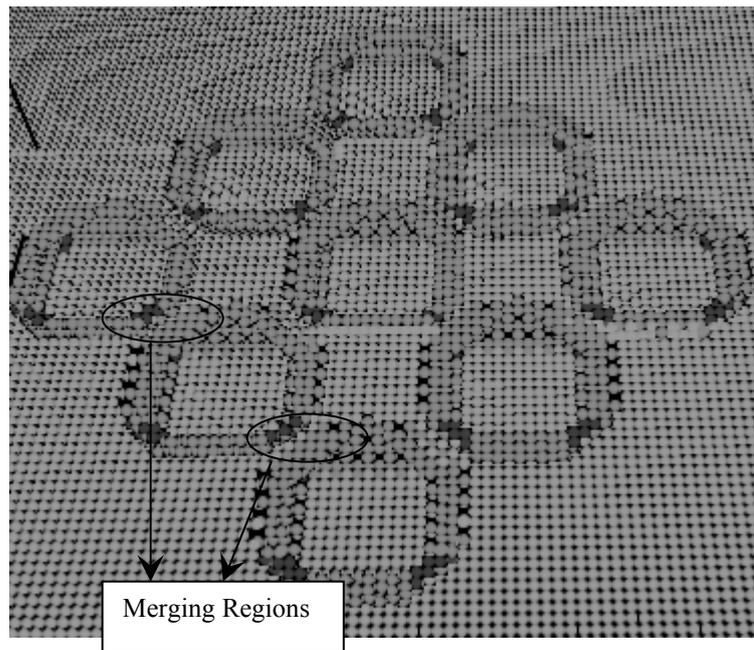


Figure 7.12: Etch Result After 150 minutes

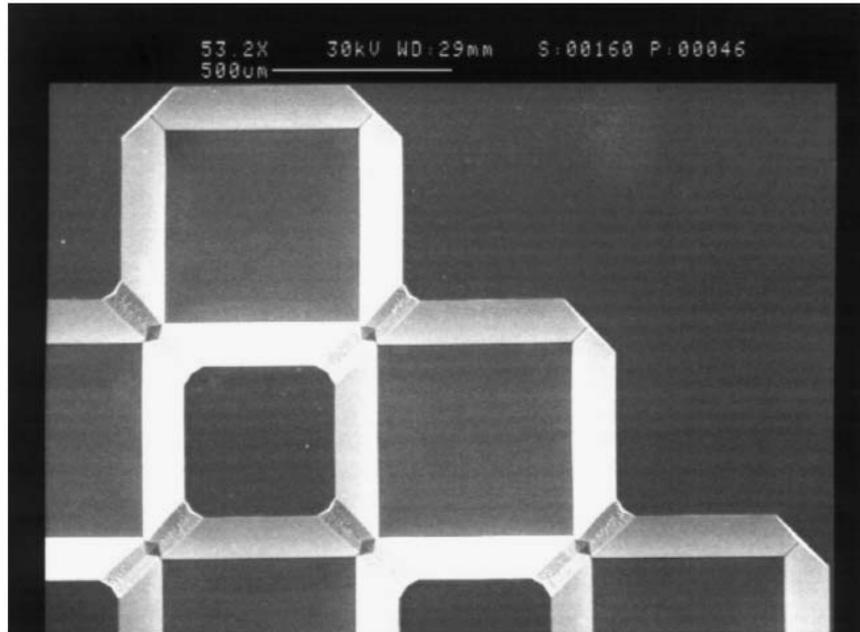


Figure 7.13: Experiment Result after 150 min. [24]

#### 7.2.1.5. Case 5 of Wet Etching

The next simulation illustrated focuses on the chemical etching process taking place under a paddle shaped mask pattern. As displayed in Fig.7.14, a paddle like mask was generated and etched in 30% KOH at (80°C).

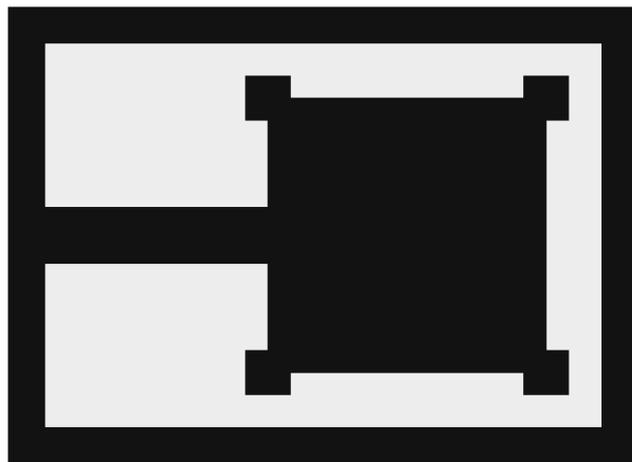


Figure 7.14: Paddle Mask

Fig. 7.15 shows the simulation results for this case. After 100 minutes, the etch stops, as expected, at regions one, two, three and four while in region five the mask was undercut until it reaches the (110) directions. The obtuse angled mask patterns are undercut through these corners in anisotropic silicon etching, this example illustrates this concept. This property can be used to have released shapes, using doping or other etch stop techniques. Finally, fig. 7.16 shows the corresponding experimental results, which are in agreement with those given by MemsEagle.

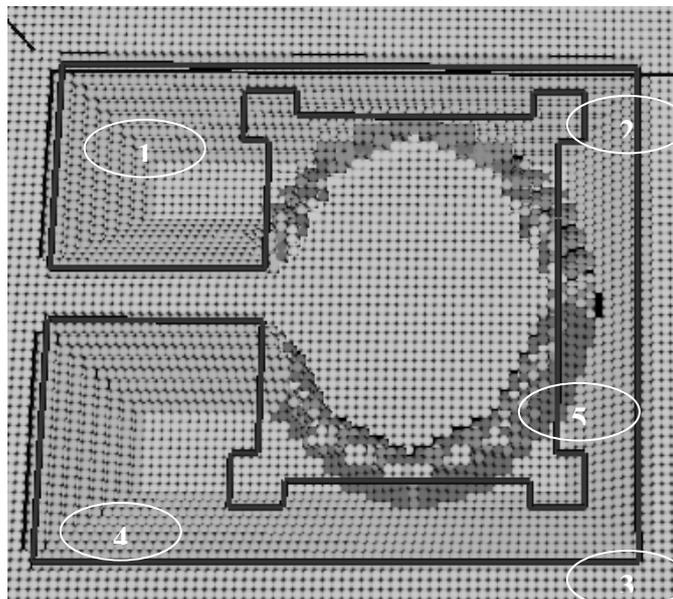


Figure 7.15: Simulation Result after 100min

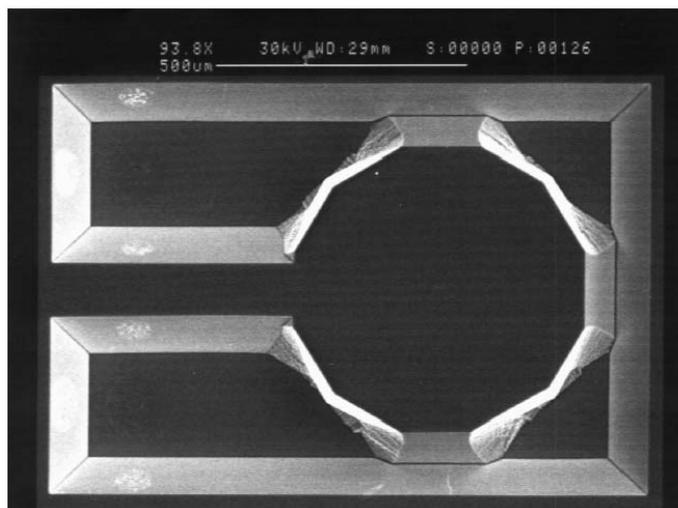


Figure 7.16: Experiment Result after 100min. [24]

### 7.2.2.6 Case 6 of Wet Etching

The following simulation illustrates the etching of the fingers of a mask pattern due to mask-undercut behavior of anisotropic etchants. As shown in Fig. 7.17 the fingers erode starting from the free tip of the finger. After 50 minutes, all the fingers were etched away. As can be seen from fig. 7.18, the simulation results match well with the experimental ones.

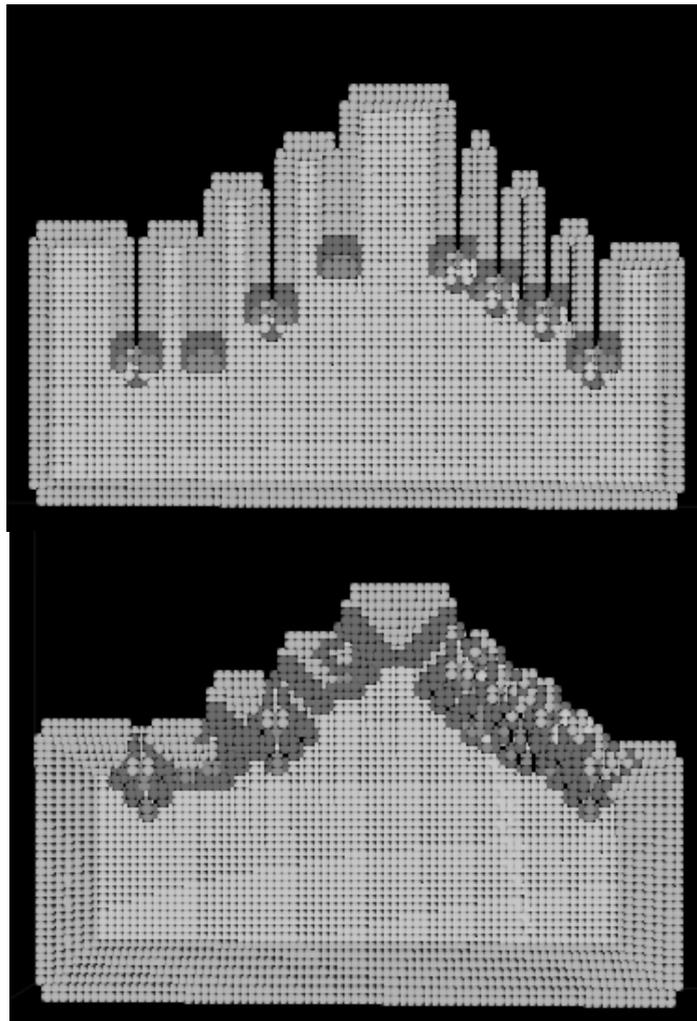


Figure 7.17: Simulation Results after 30min. and 50min.

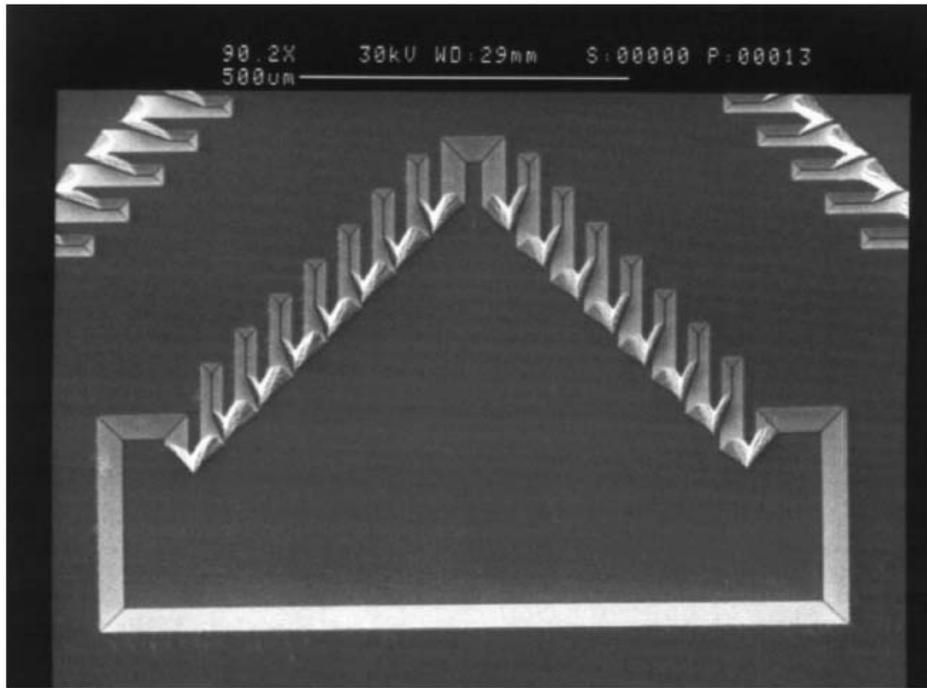


Figure 7.18: Experiment Results after 50min. [24]

Notice that, in this simulation the number of fingers along with the overall dimensions of the mask is reduced, owing to the fact that the smaller features on the mask yield huge number of CA cells used in the simulation. Thus, the simulation time becomes unnecessarily long e.g. one day of computation on a modest PC. Therefore, the presented simulation case is intended to demonstrate the behavioral properties of MemsEagle for all practical purposes.

#### 7.2.2.7. Case 7 of Wet Etching

In this case the wet etching of a complex shape referred to as “Tee” is considered. Fig. 7.19 shows not only the corresponding mask shape but also the result of the etching process of a  $\langle 100 \rangle$  wafer, exposed to 30% KOH (at  $80^\circ\text{C}$ ) for a duration of 50 minutes. Due to the rectangular mask shapes (which are aligned to (110) directions), except for the “Tee”, there is no mask-undercut. All the rectangular patterns are etched to have a pyramidal pit with  $54.7^\circ$  inclination angle, and etching slows down at (111) planes.

Notice that, having obtuse-angle corners the mask starts to undercut from the “Tee”. This undercut region (region 1), expands through the directions shown on the simulation result. Note that, if the wafer were exposed to the etchant for a sufficient time, the material under the mask would be totally undercut to yield a pyramidal form.

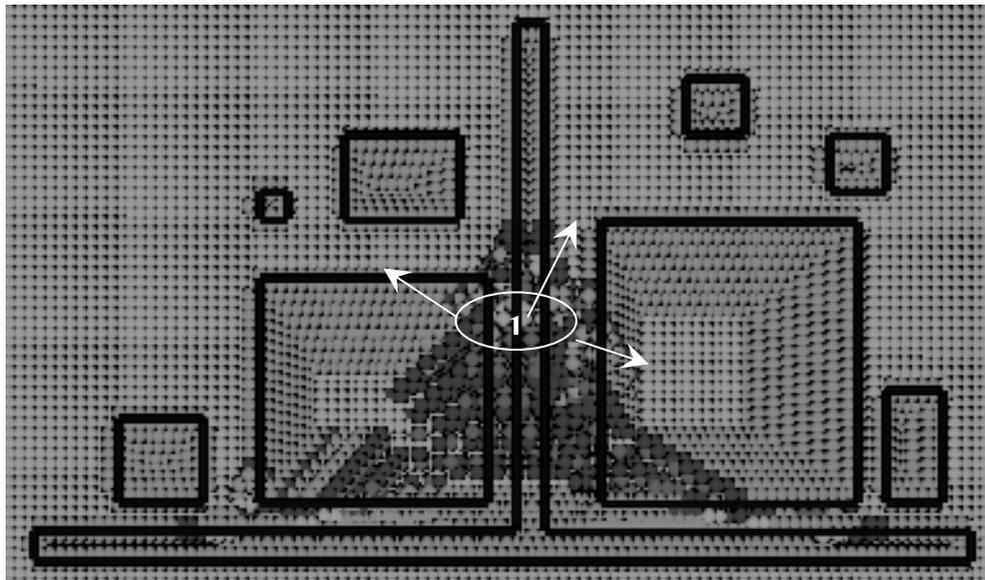


Figure 7.19: MemsEagle Simulation for “Tee”

Notice that, simulation results qualitatively match with those of the experimental results demonstrated on Fig. 7.20. All indicated observations in the simulation could be extended to the experiment as well.

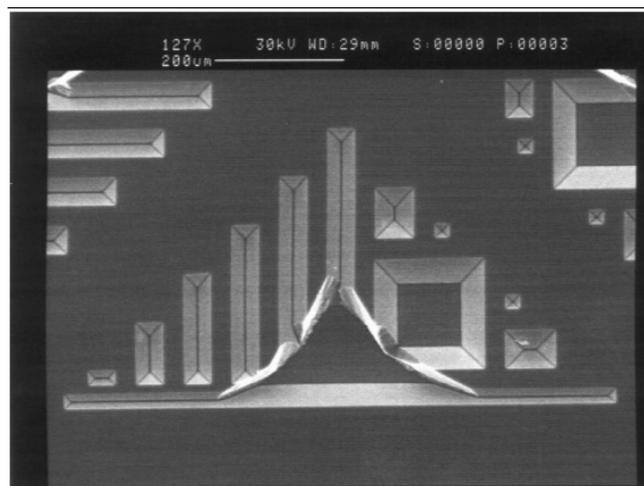


Figure 7.20: Experiment Result for “Tee” [24]

### 7.2.1.8. Case 8 of Wet Etching

Most of the etching problems occurred due to mask misalignments. In order to visualize the effect of misaligned masks, and to confirm the simulation capability of MemsEagle for such cases the mask patterns illustrated by Fig. 7.21 are used.

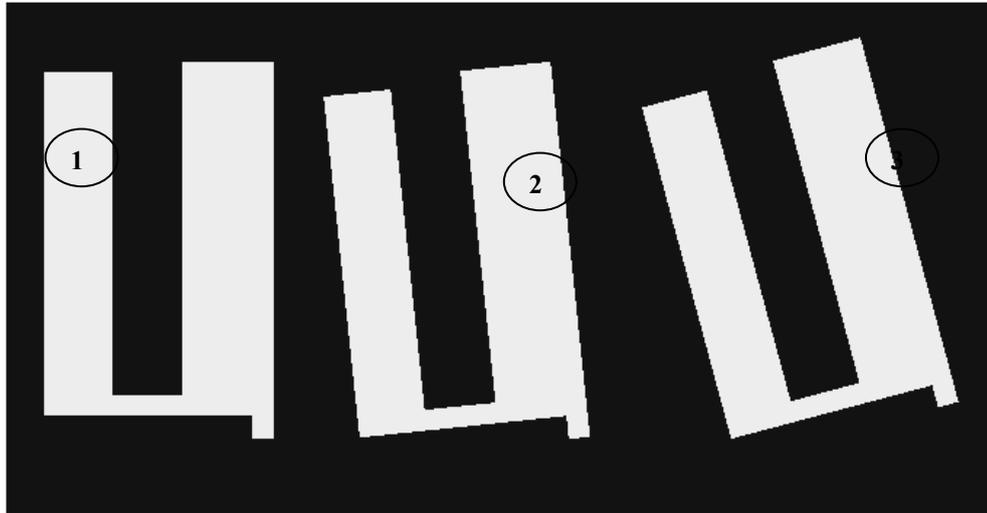


Figure 7.21: Misaligned Mask Pattern [24]

The first shape is created using the mask editor while the second and the third shapes are rotated by  $5^\circ$  and  $15^\circ$  respectively to mimic the mask misalignment effects. The same etchant and wafer type are used as with the previous cases. The results of the simulation after 50 minutes are compared to the experimental results. Fig. 7.22 illustrates the resultant shape when the first mask was used. Note that, the maximum depth is measured as  $48\mu\text{m}$  whereas the corresponding depth in the experiment happens to be  $56\mu\text{m}$ . As expected, for regions one and two, etching stopped at (111) planes through the (110) directions. Through regions three and four, because the (111) planes did not end at (110) planes, the material under the masked areas are undercut. Should the etch continue, these regions will also merge at (111) planes aligned with (110) directions. The experimental results generally agree well with the simulation results as displayed in Fig. 7.23.

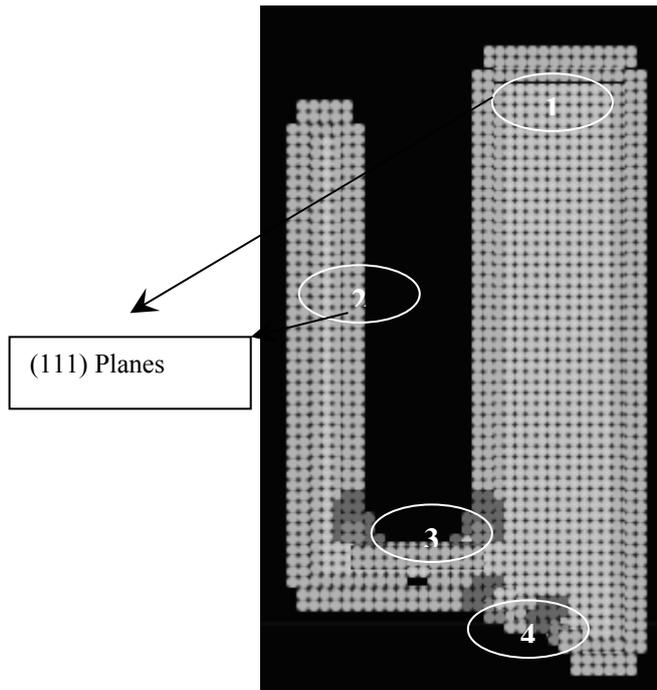


Figure 7.22: MemsEagle Result after 50min.

Notice that, there exists a discrepancy of  $56-48=8\mu\text{m}$  in the depth measurement between the experiment and the simulation. This is due to the fact that, the mask employs an isometric scaling factor of 10:1 that is a  $10\mu\text{m}$  feature in the experiment is scaled down to  $1\mu\text{m}$  in MemsEagle. Hence, the discretization effect manifests itself as a slight error in depth throughout the simulation.

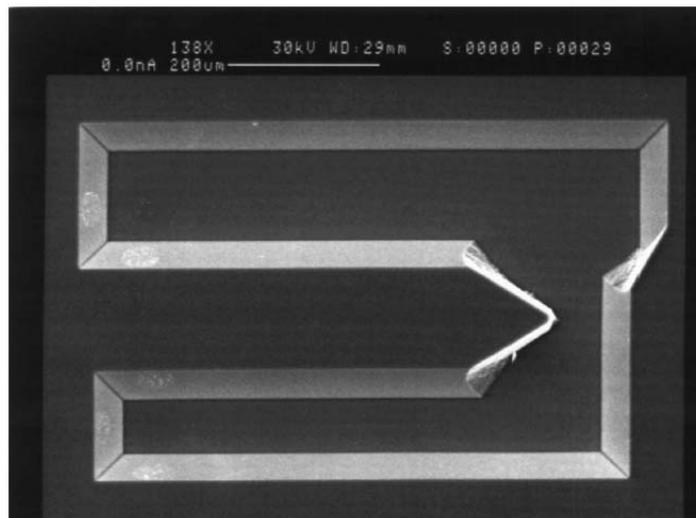


Figure 7.23: Experiment Result after 50min. [24]

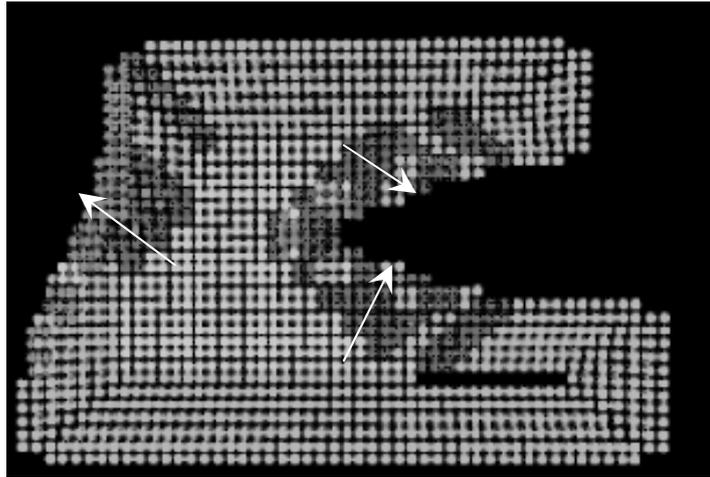


Figure 7.24: MemscEagle Result after 100min.

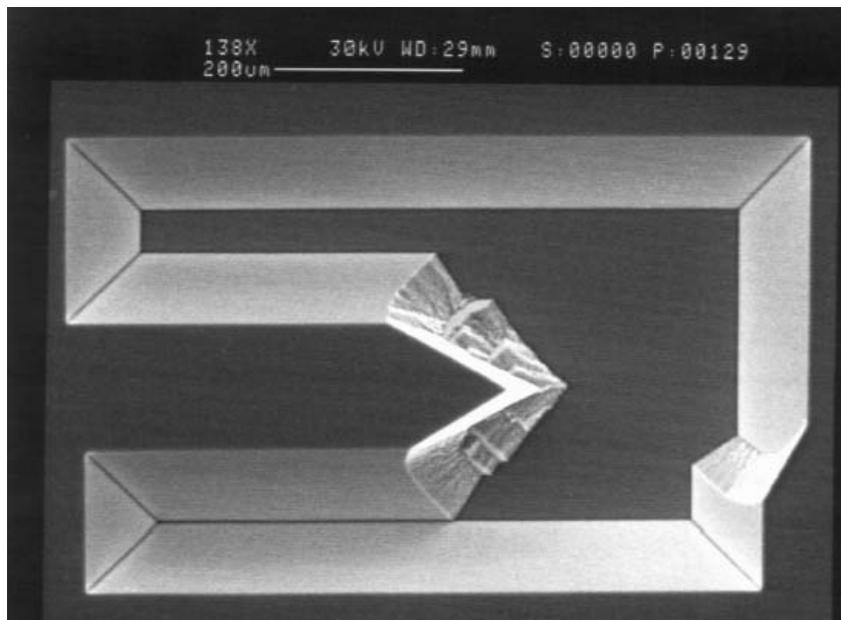


Figure 7.25: Experiment Result after 100min. [24]

When the etch continues as shown in Fig. 7.24 and 7.25, the third and fourth regions continued to expand in the directions shown by arrows in Fig 7.24. After sufficient time elapses, the resultant shape will be a pyramid whose base is a rectangle.

If the mask pattern were rotated (aka. misaligned)  $5^\circ$ , as illustrated in Fig 7.26, the regions one and two (refer to Fig.7.22) have also continued to be etched as they are no longer aligned with (110) direction. As confirmed by the experiment

results (Fig 7.28), the resultant shape would be a pyramid with a larger rectangular base.

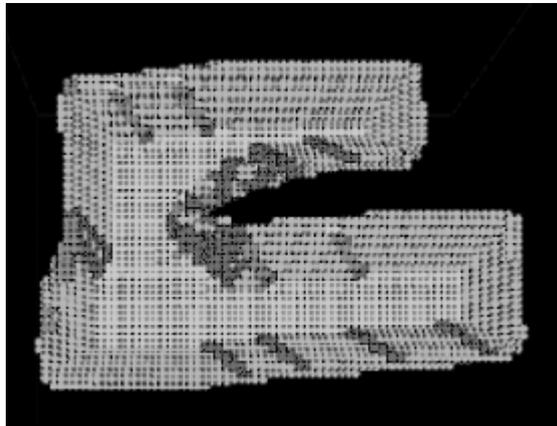


Figure 7.26: Simulation of the 5°-rotated mask

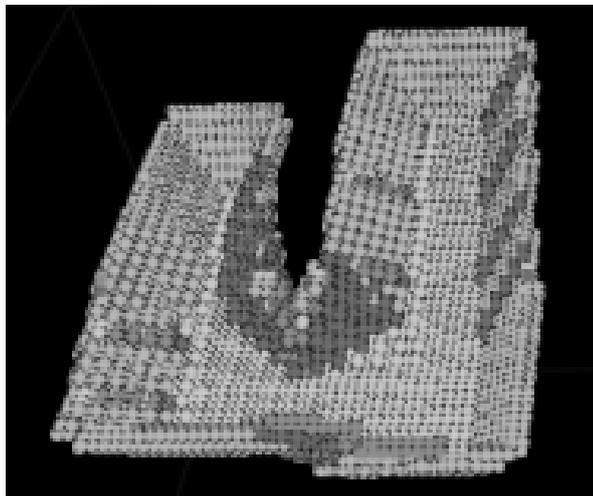


Figure 7.27: 3-D View of Simulation Result

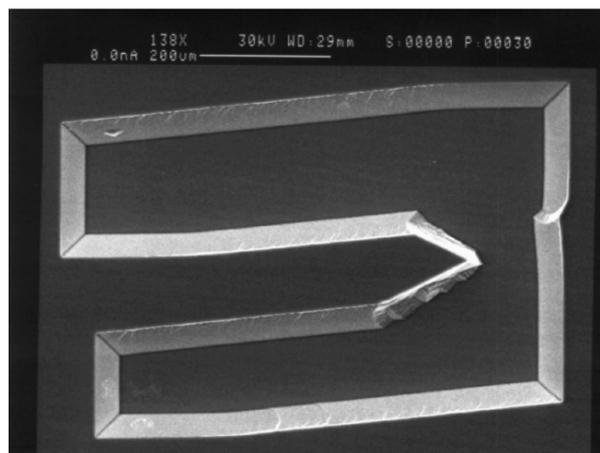


Figure 7.28: Experiment Result of the 5° rotated mask [24]

Finally, the  $15^\circ$  rotated mask pattern is simulated. The result of this rotation yields also a larger rectangular base. (Fig 7.31) The width and the length of this rectangle are based on the maximum and minimum vertical and horizontal points of the mask border as shown in Fig 7.29.

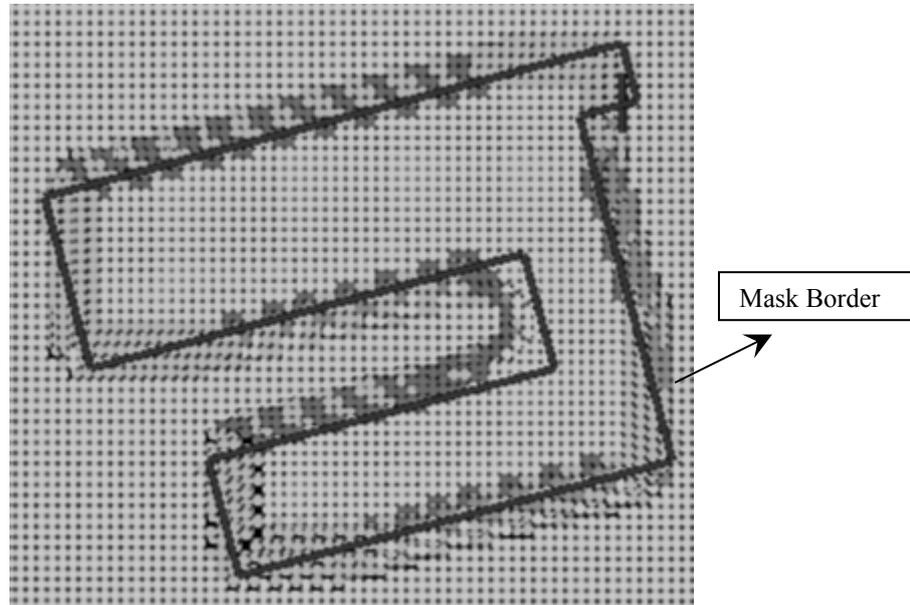


Figure 7.29: Simulation of the  $15^\circ$  rotated mask

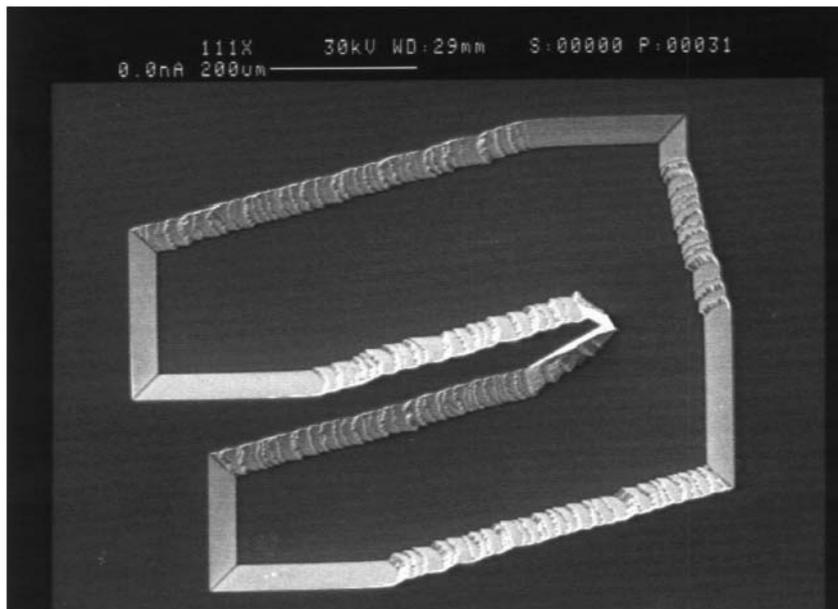


Figure 7.30: Experiment Result of the  $15^\circ$  rotated mask [24]

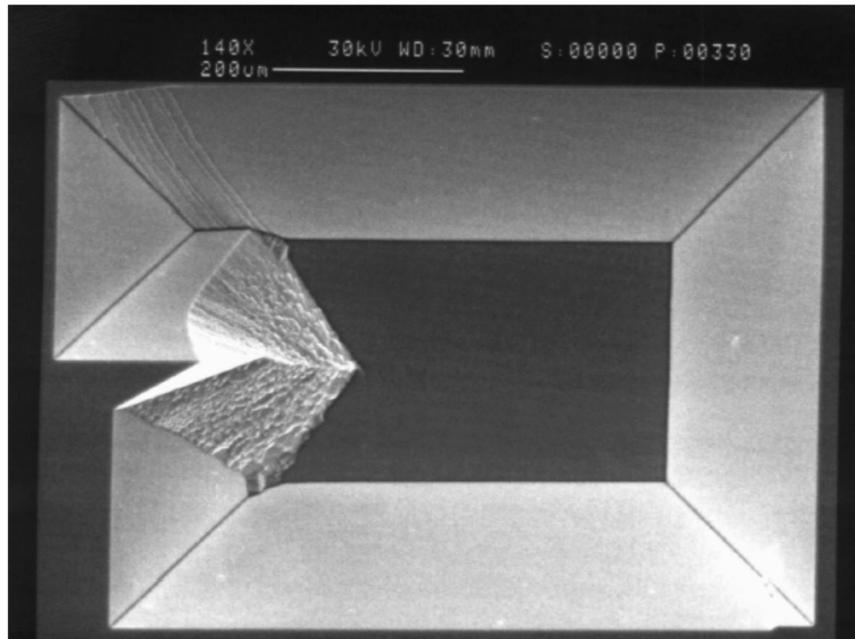


Figure 7.31: Experiment Result after 200min. [24]

The next section explores the performance of MemEagle on Deep Reactive Ion Etching (DRIE).

### 7.2.2. Deep Reactive Ion Etching

The following sample shows the dry etching capabilities of MemEagle. The software simulates Deep Reactive Ion Etching by neglecting the effect of the size of the shape to be etched. When dealing with openings with different sizes, a difference in etching rate is observed [32]. This effect is called Aspect Ratio Dependent Etching (ARDE). Fig 7.32 clearly demonstrates the ARDE effect where the depths of the holes vary with opening size.

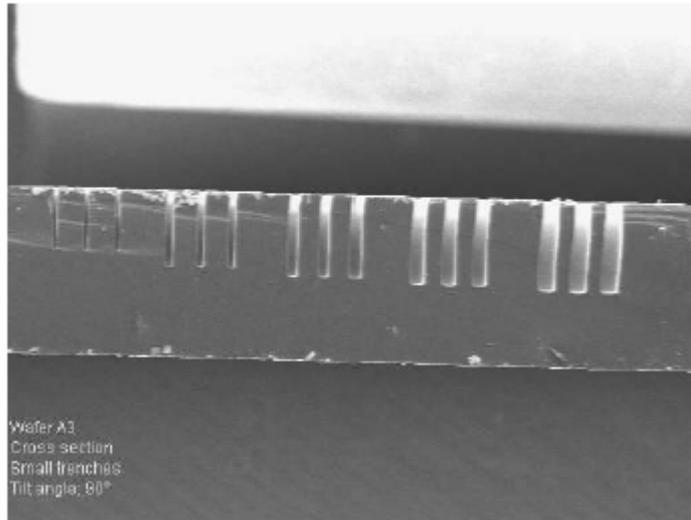


Figure 7.32: Deep Reactive Ion Etching ARDE Effect [32]

Using a mask pattern composed of circles with different diameters as illustrated in Fig 7.33, the Deep Reactive Ion Etch (DRIE) is simulated. Since the ARDE effect is neglected in the simulation, all holes exactly have the same depth as demonstrated in Fig 7.34. However, the ARDE effect can be incorporated to MemsEagle by enhancing the algorithm that will scan dynamically and locate the neighboring cells on the virtual surface, and could define the holes in which the CA cells are located. However, enhancing the algorithm this way requires considerable software development efforts.

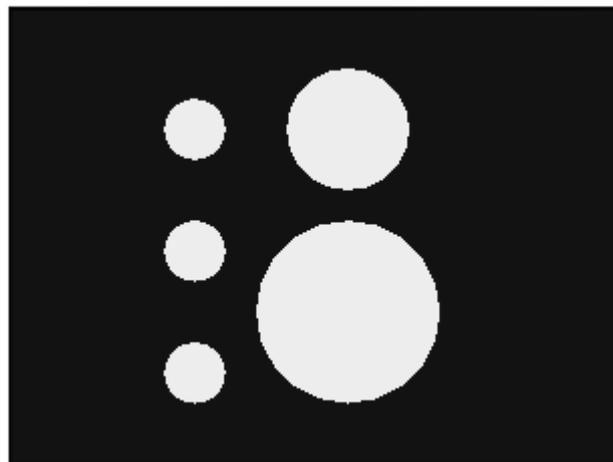


Figure 7.33: Mask Pattern for DRIE

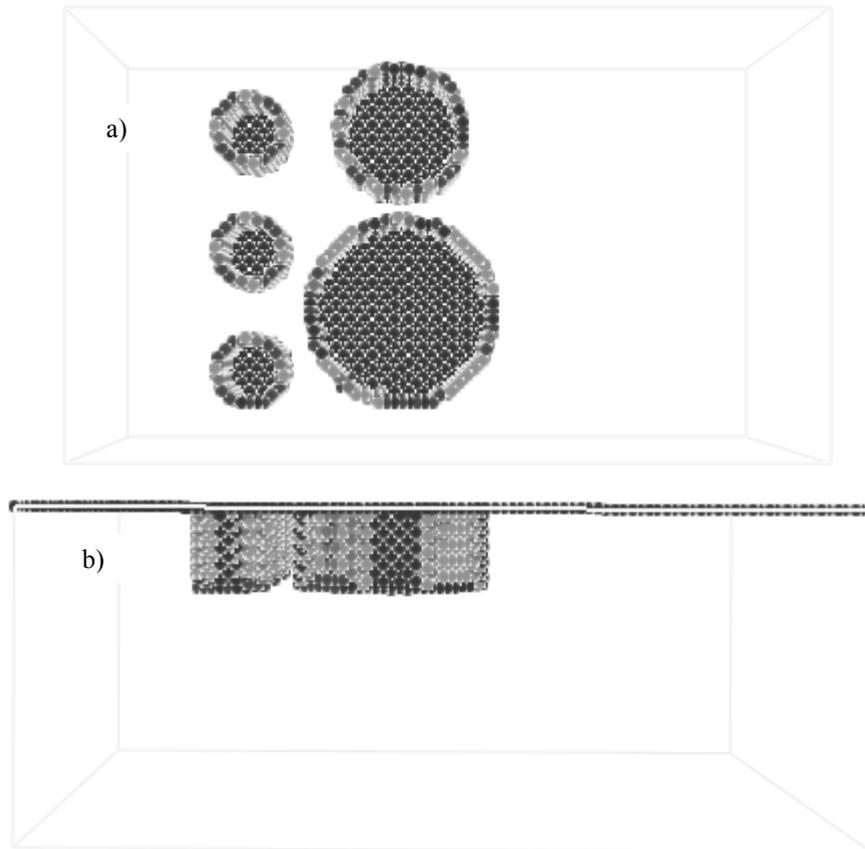


Figure 7.34: DRIE Simulation Result (a) top view, (b) front view

### 7.2.3. Doping

Doping is mostly utilized in MEMS fabrication processes for the purpose of stopping the etch as well as creating independent machine elements and structures on the surface through bulk micromachining processes.

In this case, an anisotropic etching is performed on a doped shape which is to be released at the end of the process, through the use of the dopant as the etch stop. First, two rectangular masks are created using the mask editor. The first mask, whose surface area is smaller than the other, was used for doping. The  $\langle 100 \rangle$  wafer is doped for a pre-deposition time of 10min and a drive-in time of 30min while the temperature of the doping process is  $1200^{\circ}\text{C}$ . Thus, a shallow heavily doped region is to be formed. After this step, the second mask is applied to etch

the substrate by a doping concentration dependant etchant for releasing the heavily doped region. Fig. 7.35 illustrates the result of the simulation, the wafer is etched totally around the heavily doped region and that part is released as expected. Note that, the lateral penetration of the dopant is neglected for all practical purposes. Only the un-masked areas are doped during the process.

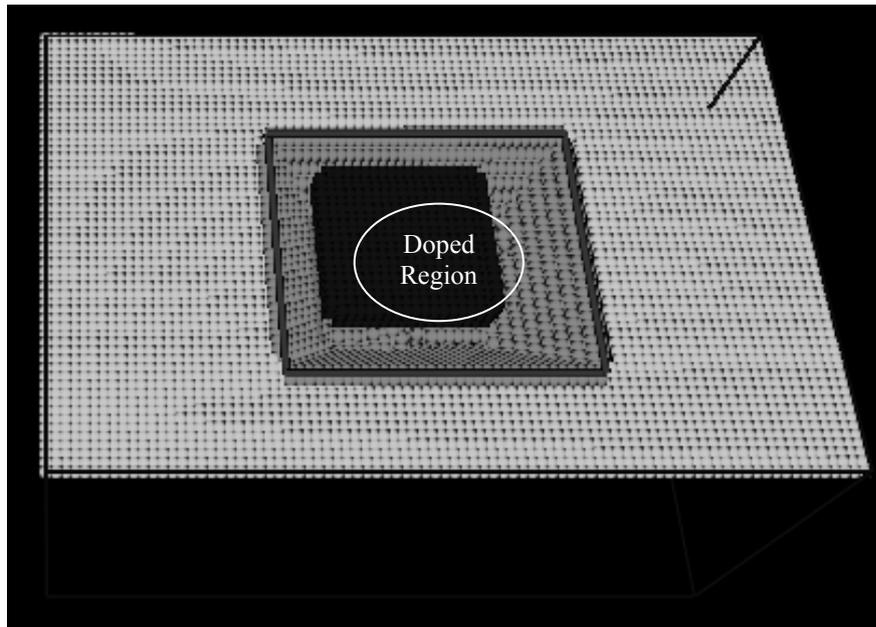


Figure 7.35: Released Part

As a final example for doping, a cantilever beam which is to be used as a micro-switch in many MEMS applications is considered. As can be seen in Fig. 7.36 the desired cantilever beam can be formed with the application of two mask sequentially. One is for the doping and the other one is for etching of the surrounding of the doped region.

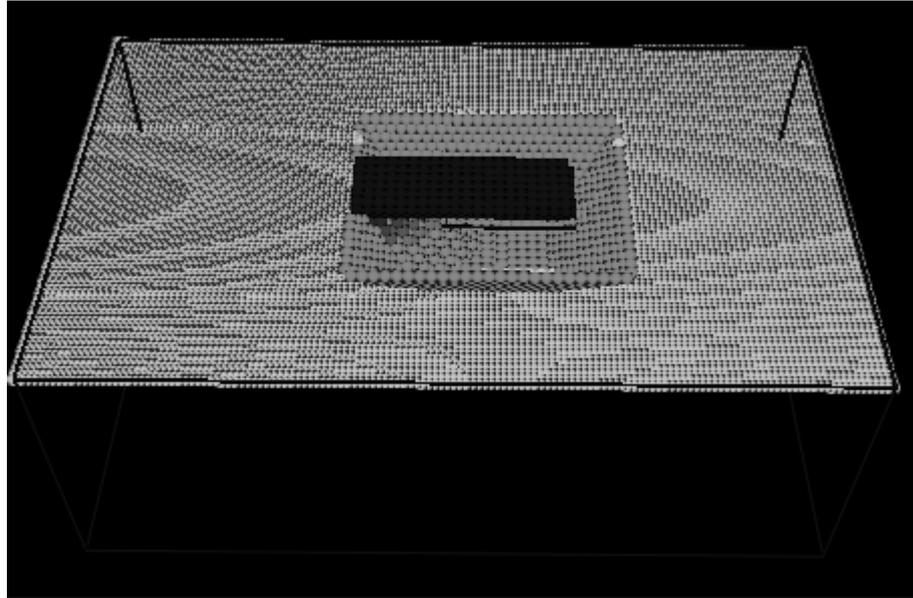


Figure 7.36: Cantilever Beam

#### 7.2.4 Additive Process

In MemsEagle surface micromachining processes can be easily simulated, with the following reservations:

1. Micromachining is the last application process to be applied on the wafer.
2. The mask shape can be directly transferred to the material layer being patterned. No dimensional errors are occurred in the lateral directions.

Based on these assumptions MemsEagle directly deposit the patterned mask shape onto the substrate and the other patterned layers follows the topology of the preceding ones. Figures. 7.37 and 7.38. show a layer of Silicon Nitride deposited on the wafer.

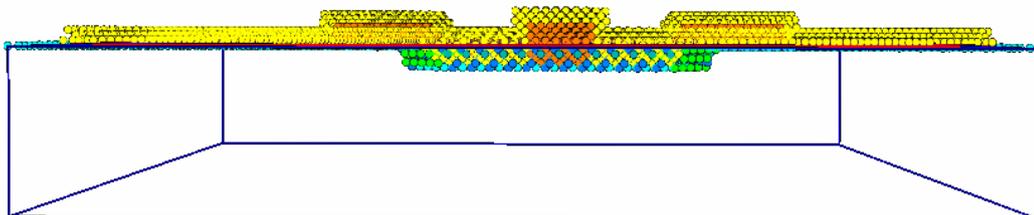


Figure 7.37: Silicon Nitride Deposition

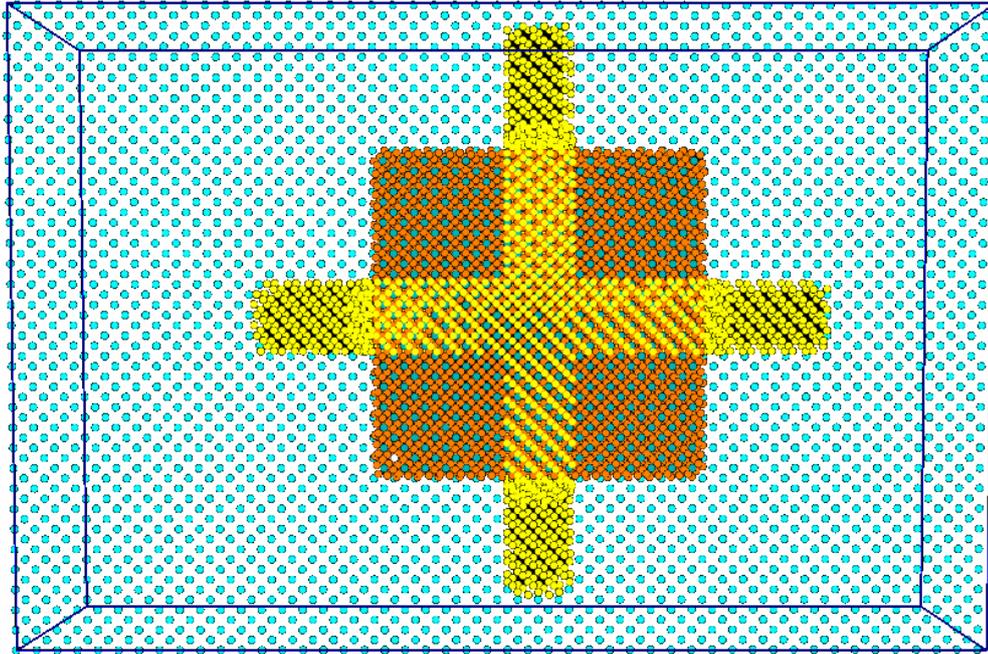


Figure 7.38: Silicon Nitride Deposition Top View

### 7.3. Etch Rate Verification

Calculation of the “Effective Etch Rate” was summarized in this section. Based upon this information, a spoke pattern was used to verify the etch rates. In order to convert the etch rates obtained from previous researches, an algorithm is devised for MemEagle. As mentioned before, a cell may have a state value between 0 and 1. In addition, when a cell is etched, its neighbors, which are  $0.432\mu\text{m}$  away, are added to the wafer-etchant interface. Note that, in order to simulate the processes accurately, the etch rates are divided by this value to have the “Effective Etch Rate” to be used on the state value of the cell.

The second approach used for determining the etch rates is to calculate the etch rates for scaled wafers. As mentioned, wafers with surface area above then  $1500\mu\text{m}^2$  are scaled down to have a surface area of  $1500\mu\text{m}^2$ . The scale factor calculated is also used to determine the “Effective Etch Rate”. The etch rates obtained is divided by this value and the resultant value is applied to the cells.

Since no figures on the etch rates could be directly obtained from the literature, the verification of the etch rates has to be qualitatively obtained by taking a look at various a spoke patterns when different etchants (KOH, EDP) are used. As can be seen from Figures 7.39 and 7.40, the simulated results as well as the experiment results are in good agreement with each other. Thus, the etching performance of MemsEagle has been found realistic for the simulation of most practical MEMS designs.

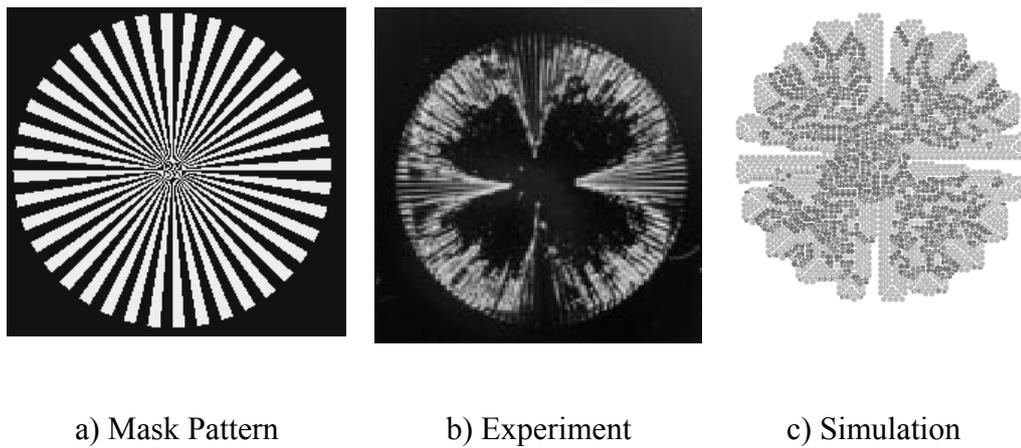


Figure 7.39: Spoke Pattern etched by EDP

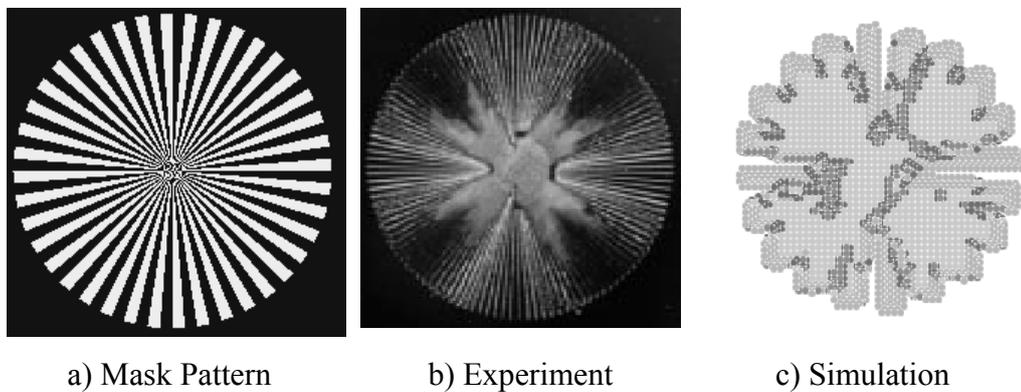


Figure 7.40: Spoke Pattern etched by KOH

#### 7.4. Program Performance

In order to simulate sequentially, the etching process for different wafer types with various etchants, all the information obtained from each process is stored in

the “VS (Virtual Surface)” array created. The number of cells stored could be up to 150.000 for certain complex shapes. But, even with these sizes of arrays, the simulation could be carried on most personal computers within reasonable durations. The computation time for a shape consisting of 60.000 cells and 40 time steps take 25-35 minutes on a common PC (256MB of RAM, 1400Mhz Processor)

### 7.5. Quantitative Analysis on the Simulation Results

To assess the accuracy of the simulation results, a quantitative analysis on geometric errors resulting from various for bulk micromachining processes, is conducted. To accomplish that, the previous cases where the experimental results are also available through [24] are taken into consideration. Unfortunately, the exact geometry of the masks being used for the experimental studies are not specified. Therefore, in the simulation with MemsEagle, the masks are designed by taking rough measurements on the resultant shapes formed after the experimentation. Table 7.1 demonstrates the results for the afore mentioned cases 3 to 8. In this table, the etching conditions including the etchant type, temperature, process time are given. The columns, experimental and simulation illustrates the maximum depth of the etch surfaces obtained through experimentation and simulation.

Table 7.1: Maximum Depth of the Etched Surfaces

| Process | Etchant | Temp. [°C] | Time[min] | Experimental [µm] | Simulation [µm] | Relative Error |
|---------|---------|------------|-----------|-------------------|-----------------|----------------|
| Case 3  | KOH %30 | 80         | 100       | 109               | 105             | 3,67%          |
| Case 4  | KOH %30 | 80         | 100       | 110               | 105             | 4,55%          |
| Case 4  | KOH %30 | 80         | 150       | 166               | 155             | 6,63%          |
| Case 5  | KOH %30 | 80         | 100       | 110               | 105             | 4,55%          |
| Case 6  | KOH %30 | 80         | 50        | 55                | 52              | 5,45%          |
| Case 6  | KOH %30 | 80         | 100       | 110               | 105             | 4,55%          |
| Case 7  | KOH %30 | 80         | 50        | 56                | 52              | 7,14%          |
| Case 7  | KOH %30 | 80         | 100       | 110               | 105             | 4,55%          |
| Case 8  | KOH %30 | 80         | 50        | 56                | 52              | 7,14%          |
| Case 8  | KOH %30 | 80         | 100       | 110               | 105             | 4,55%          |

Notice that the relative error (E%) is defined as:

$$E\% = \frac{d_{\text{exp}} - d}{d_{\text{exp}}} \cdot 100 \quad (7.1)$$

where  $d_{\text{exp}}$ ,  $d$  refer to the depth values obtained through experiments and simulations respectively. As can be seen the maximum relative error is about 7%, despite the inaccuracies introduced by wafer scaling, mask design as well as uncertainties in the etch rates. Results of MemsEagle are in excellent agreement with the experimental results, thanks to the realistic simulation properties offered by the CA method.

## 7.6. Closure

As the program based on the “Cellular Automata” approach, the die on the wafer has to be discretized. Due to this feature, using larger size arrays could improve the accuracy of the results. To test the performance of the device software, a number of simulations were carried out. The experimental results show that, the simulation results are in good agreement with the actual cases.

Even though the simulation accuracy is sufficient for the simulation of most practical MEMS designs, there exists a significant improvement opportunity in MemsEagle. Higher order planes such as (211) and (411) could also be added to the etching algorithm and a more-packed silicon lattice can be used to enhance the simulation accuracy of the program.

## CHAPTER 8

### CONCLUSION AND FUTURE WORK

In this thesis a software code for simulating micro-machining processes of silicon wafers that are used in Micro-Electro-Mechanical-Systems (MEMS) have been developed. The wet and dry etching including anisotropic and isotropic etching, doping and additive processes could be simulated.

The outputs of the program includes the depth of the resulting shape, 3-D model of the etch result and the dimensions of the final product. The user can design the mask by the integrated mask design tool in MemsEagle. Both inside and outside of the mask shape can be etched, or exposed to the process selected.

Main advantage of MemsEagle is the integrated design environment it provides to the users. Different from the other codes, there is a mask design tool and different properties of each cell can be accessed. Besides the anisotropic etching, MemsEagle can be utilized for simulation of additive processes like doping. Table 8.1 demonstrates the features of the softwares developed during past researches and MemsEagle.

Table 8.1: Features of the Available Softwares

| Program          | Simulation Capability            |                                   |               | Design Capability  |                         |                      |
|------------------|----------------------------------|-----------------------------------|---------------|--------------------|-------------------------|----------------------|
|                  | <i>Bulk <math>\mu</math>-fab</i> | <i>Surf. <math>\mu</math>-fab</i> | <i>Doping</i> | <i>Mask Design</i> | <i>Project Creation</i> | <i>Visualization</i> |
| <b>ASEP</b>      | Yes                              | No                                | No            | No                 | Yes                     | Planes               |
| <b>SEGS</b>      | Yes                              | No                                | No            | No                 | No                      | Planes               |
| <b>Suzana</b>    | Yes                              | No                                | No            | No                 | No                      | 3-D Solid            |
| <b>AnisE</b>     | Yes                              | No                                | No            | No                 | Yes                     | Cells                |
| <b>ACES</b>      | Yes                              | No                                | Yes           | No                 | Partial                 | Cells                |
| <b>SIMODE</b>    | Yes                              | No                                | No            | No                 | Yes                     | Planes               |
| <b>MemsEagle</b> | Yes                              | Yes                               | Yes           | Yes                | Yes                     | Cells                |

Notice that in this table the visualization column refers to how the simulation results are presented. For instance, the word “planes” in this column denotes the simulation results are displayed as a collection of planes while “cells” indicate that only the spatial location of various surface elements are shown. Similarly, “3-D Solid” refers that all results are demonstrated as assembly of 3-dimensional objects such as cones, cylinders, planes etc. Hence, visual interpretation of such data is much more convenient if compared to others. As can be seen from the Table 8.1, MemsEagle integrates many of the unavailable simulation features of the other softwares. Furthermore, integrated design capability of MemsEagle is seemingly superior than most of the available freeware packages.

Apart from advanced design features, the simulation capabilities of MemsEagle (especially for bulk micromachining) have proven to be distinctive. That is, the outputs of the program are in excellent agreement with the experimental results presented by various researchers. Furthermore, with the advancements in the microprocessor technology, most complex MEMS fabrication processes can be realistically simulated by MemsEagle running on even a modest personal computer within reasonable time periods.

### **8.1. Future work**

Even though MemsEagle has a potential to be an integrated design environment, the development efforts are far from over for this software. The software needs to have various nice to have features so as to be a complete and user-friendly environment for MEMS designs. The features to be added on this software are as follows:

- A mesh generation tool for the resulting shapes has to be developed. By providing this, the results of MemsEagle can further be processed via commercial codes like Ansys, Matlab etc.
- Masks designed by other programs like AutoCad, can be imported by means of modifications in the mask design algorithm.

- The wafer types and etch material database can be extended for detailed designs.
- In surface micromachining, removal of sacrificial layers has not been taken into consideration in MemsEagle yet. The CA algorithm has to be extended to cover the etching of such thin sacrificial layers.
- Higher order planes like (211), (540) can be included in etch algorithm for more accurate results.
- Etch concentration changes as a function of time in detail for more accurate results.
- ARDE effect on Deep Reactive Ion Etching should be modeled.
- Lateral penetration of Doping has to be modeled.
- Final achievement for MemsEagle could be implementing other processes like wafer bonding and having design procedures like MUMPS.

By having a software code like MemsEagle, especially the design time for a MEMS product will be lower, and accurate results can be obtained without needing trial-error usage of mask shapes.

## APPENDIX A

### ETCH RATES

#### A.1. KOH Etch Rates

The KOH etch rate is strongly effected by the crystallographic orientation of the Silicon. Table 1 relates silicon orientation-dependent etch rates ( $\mu\text{m}\cdot\text{min}^{-1}$ ) of KOH to crystal orientation with an etching temperature of  $70^{\circ}\text{C}$ . Table A.1 is taken directly from [33]. In parentheses are normalized values relative to (110).

Table A.1: KOH Etch Rates

| Crystallographic Orientation | Rates at different KOH Concentration at $70^{\circ}\text{C}$ |               |               |
|------------------------------|--|---------------|---------------|
|                              | 30%  | 40%           | 50%           |
| (100)                        | 0.797 (0.548)  | 0.599 (0.463) | 0.539 (0.619) |
| (110)                        | 1.455 (1.000)  | 1.294 (1.000) | 0.870 (1.000) |
| (210)                        | 1.561 (1.072)  | 1.233 (0.953) | 0.959 (1.103) |
| (211)                        | 1.319 (0.906)  | 0.950 (0.734) | 0.621 (0.714) |
| (221)                        | 0.714 (0.491)  | 0.544 (0.420) | 0.322 (0.371) |
| (310)                        | 1.456 (1.000)  | 1.088 (0.841) | 0.757 (0.871) |
| (311)                        | 1.436 (0.987)  | 1.067 (0.824) | 0.746 (0.858) |
| (320)                        | 1.543 (1.060)  | 1.287 (0.995) | 1.013 (1.165) |
| (331)                        | 1.160 (0.797)  | 0.800 (0.619) | 0.489 (0.563) |
| (530)                        | 1.556 (1.069)  | 1.280 (0.989) | 1.033 (1.188) |
| (540)                        | 1.512 (1.039)  | 1.287 (0.994) | 0.914 (1.051) |
| (111)                        | 0.005 (0.004)  | 0.009 (0.007) | 0.009 (0.010) |

Table A.2 relates silicon orientation-dependent etch rates of KOH to percent composition, temperature, and orientation. This table is taken from [34]. As with all wet-chemical etching solutions, the dissolution rate is a strong function of temperature. Significantly faster etch rates at higher temperatures are typical, but less ideal etch behavior is also common with more aggressive etch rates. Also,

heavy boron doping can significantly harden the silicon and sharply reduce the etch rate.

Table A.2: KOH Etch Rates vs. Composition and Temperature

| Etchant                                       | Temperature °C | Direction (Plane) | Etch Rate [ $\mu\text{m}/\text{min}$ ] |
|---|----------------|-------------------|--|
| 20% KOH:80% H <sub>2</sub> O                  | 20             | (100)             | 0,025                                  |
|   | 40             | (100)             | 0,188                                  |
|   | 60             | (100)             | 0,45                                   |
|   | 80             | (100)             | 1,4                                    |
|   | 100            | (100)             | 4,1                                    |
| 30% KOH:70% H <sub>2</sub> O                  | 20             | (100)             | 0,024                                  |
|   | 40             | (100)             | 0,108                                  |
|   | 60             | (100)             | 0,41                                   |
|   | 80             | (100)             | 1,3                                    |
|   | 100            | (100)             | 3,8                                    |
|   | 20             | (110)             | 0,035                                  |
|   | 40             | (110)             | 0,16                                   |
|   | 60             | (110)             | 0,62                                   |
|   | 80             | (110)             | 2                                      |
|   | 100            | (110)             | 5,8                                    |
| 40% KOH:60% H <sub>2</sub> O                  | 20             | (100)             | 0,02                                   |
|   | 40             | (100)             | 0,088                                  |
|   | 60             | (100)             | 0,33                                   |
|   | 80             | (100)             | 1,1                                    |
|   | 100            | (100)             | 3,1                                    |
| 20% KOH:80% 4H <sub>2</sub> O: 1 IPA          | 20             | (100)             | 0,015                                  |
|   | 40             | (100)             | 0,071                                  |
|   | 60             | (100)             | 0,28                                   |
|   | 80             | (100)             | 0,96                                   |
|   | 100            | (100)             | 2,9                                    |
| 44% KOH: 56% H <sub>2</sub> O                 | 120            | (100)             | 5,8                                    |
|   | 120            | (110)             | 11,7                                   |
|   | 120            | (111)             | 0,02                                   |
| 23.4%KOH: 63.3%H <sub>2</sub> O:<br>13.3% IPA | 80             | (100)             | 1                                      |
|   | 80             | (110)             | 0,06                                   |

## A.2. TMAH Etch Rates

The orientation dependence of the TMAH etch rate is similar to KOH and varies similarly in accordance to the atomic organization of the crystallographic plane. Table A.3 [35] relates silicon orientation-dependent etch rates of TMAH (20.0wt%, 79.8°C) to orientation.

Table A.3: TMAH Etching Rates vs. Orientation

| Crystallographic Orientation | Etching rate [ $\mu\text{m}/\text{min}$ ] | Etching rate ratio |               |
|------------------------------|---|--------------------|---------------|
|                              |   | (i j k)/(100)      | (i j k)/(111) |
| (100)                        | 0.603                                     | 1.000              | 37            |
| (110)                        | 1.114                                     | 1.847              | 68            |
| (210)                        | 1.154                                     | 1.914              | 70            |
| (211)                        | 1.132                                     | 1.877              | 69            |
| (221)                        | 1.142                                     | 1.894              | 69            |
| (310)                        | 1.184                                     | 1.964              | 72            |
| (311)                        | 1.223                                     | 2.028              | 74            |
| (320)                        | 1.211                                     | 2.008              | 73            |
| (331)                        | 1.099                                     | 1.823              | 67            |
| (530)                        | 1.097                                     | 1.819              | 66            |
| (540)                        | 1.135                                     | 1.882              | 69            |
| (111)                        | 0.017                                     | 0.027              | 1             |

Similar to KOH, the TMAH etch rate varies exponentially with temperature. Table A.4 [34] relates silicon orientation-dependent etch rates of TMAH to percent composition, temperature, and orientation.

Table A.4: TMAH Etch Rates vs. Composition and Temperature

| Etchant  | Temperature °C | Direction(Plane) | Etch Rate [mμ/min] |
|--|----------------|------------------|--------------------|
| 5% TMAH::95% H <sub>2</sub> O                  | 60             | (100)            | 0,33               |
|  | 70             | (100)            | 0,48               |
|  | 80             | (100)            | 0,87               |
|  | 90             | (100)            | 1,4                |
|  | 60             | (110)            | 0,64               |
|  | 70             | (110)            | 0,74               |
|  | 80             | (110)            | 1,4                |
|  | 90             | (110)            | 1,8                |
|  | 60             | (111)            | 0,026              |
|  | 90             | (111)            | 0,034              |
| 10% TMAH:90% H <sub>2</sub> O                  | 60             | (100)            | 0,28               |
|  | 70             | (100)            | 0,41               |
|  | 80             | (100)            | 0,72               |
|  | 90             | (100)            | 1,2                |
| 2% TMAH:98% H <sub>2</sub> O                   | 80             | (100)            | 0,65               |
|  | 80             | (111)            | 0,41               |
| 5% TMAH:95% H <sub>2</sub> O                   | 80             | (100)            | 0,63               |
|  | 80             | (111)            | 0,013              |
| 10% TMAH:90% H <sub>2</sub> O                  | 80             | (100)            | 0,57               |
|  | 80             | (111)            | 0,014              |
| 22% TMAH in H <sub>2</sub> O                   | 90             | (100)            | 0,9                |
|  | 90             | (110)            | 1,8                |
|  | 90             | (111)            | 0,018              |
| 22% TMAH in H <sub>2</sub> O + 0.5% surfactant | 90             | (100)            | 0,6                |
|  | 90             | (110)            | 0,12               |
|  | 90             | (111)            | 0,01               |
| 22% TMAH in H <sub>2</sub> O + 1% surfactant   | 90             | (100)            | 0,6                |
|  | 90             | (110)            | 0,1                |
|  | 90             | (111)            | 0,009              |

### A.3. EDP Etch Rates

Similar to KOH, EDP is often used for fast removal and silicon micromachining. Table A.5 [13] relates silicon orientation-dependent etch rates in EDP solutions to Temperature and Orientation.

Table A.5: EDP Etch Rates vs. Composition and Temperature

| Etchant  | Temperature °C | Direction(Plane) | Etch Rate [ $\mu\text{m}/\text{min}$ ] |
|--|----------------|------------------|--|
| 500 ml $\text{NH}_2(\text{CH}_2)_2\text{NH}_2$ :88g<br>$\text{C}_6\text{H}_4(\text{OH})_2$ : 234 ml $\text{H}_2\text{O}$   | 110            | (100)            | 0,47                                   |
|  | 110            | (110)            | 0,28                                   |
|  | 110            | (111)            | 0,028                                  |
| 500 ml $\text{NH}_2(\text{CH}_2)_2\text{NH}_2$ :160g<br>$\text{C}_6\text{H}_4(\text{OH})_2$ : 160 ml $\text{H}_2\text{O}$  | 115            | (100)            | 0,45                                   |
| 500 ml $\text{NH}_2(\text{CH}_2)_2\text{NH}_2$ :160g<br>$\text{C}_6\text{H}_4(\text{OH})_2$ : 160 ml $\text{H}_2\text{O}$ 3.0g<br>$\text{C}_6\text{H}_4\text{N}_2$ | 115            | (100)            | 0,65                                   |
| 500 ml $\text{NH}_2(\text{CH}_2)_2\text{NH}_2$ :80g<br>$\text{C}_6\text{H}_4(\text{OH})_2$ : 66 ml $\text{H}_2\text{O}$ 3.6g<br>$\text{C}_6\text{H}_4\text{N}_2$   | 50             | (100)            | 0,075                                  |
|  | 75             | (100)            | 0,22                                   |
|  | 95             | (100)            | 0,43                                   |
|  | 105            | (100)            | 0,57                                   |
|  | 110            | (100)            | 0,75                                   |

## APPENDIX B

### OPENGL FUNCTIONS

The OpenGL Functions mentioned in the previous chapters are explained in detail.

#### B.1. glBegin – glEnd functions

```
void glBegin( GLenum mode )
```

##### PARAMETERS

*mode* : Specifies the primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd. Ten symbolic constants are accepted: GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_LINE\_LOOP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_QUADS, GL\_QUAD\_STRIP, and GL\_POLYGON.

```
void glEnd( void )
```

##### DESCRIPTION

glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies which of ten ways the vertices are interpreted. Taking  $n$  as an integer count starting at one, and  $N$  as the total number of vertices specified, the interpretations are as follows:

GL\_POINTS: Treats each vertex as a single point. Vertex  $n$  defines point  $n$ .  $N$  points are drawn.

GL\_LINES: Treats each pair of vertices as an independent line segment. Vertices  $2n-1$  and  $2n$  define line  $n$ .  $N/2$  lines are drawn.

GL\_LINE\_STRIP: Draws a connected group of line segments from the first vertex to the last. Vertices  $n$  and  $n+1$  define line  $n$ .  $N-1$  lines drawn.

GL\_LINE\_LOOP: Draws a connected group of line segments from the first vertex to the last, then back to the first. Vertices  $n$  and  $n+1$  define line  $n$ . The last line, however, is defined by vertices  $N$  and  $1$ .  $N$  lines are drawn.

GL\_TRIANGLES: Treats each triplet of vertices as an independent triangle. Vertices  $3n-2$ ,  $3n-1$ , and  $3n$  define triangle  $n$ .  $N/3$  triangles are drawn.

GL\_TRIANGLE\_STRIP: Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. For odd  $n$ , vertices  $n$ ,  $n+1$ , and  $n+2$  define triangle  $n$ . For even  $n$ , vertices  $n+1$ ,  $n$ , and  $n+2$  define triangle  $n$ .  $N-2$  triangles are drawn.

GL\_TRIANGLE\_FAN: Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. Vertices  $1$ ,  $n+1$ , and  $n+2$  define triangle  $n$ .  $N-2$  triangles are drawn.

GL\_QUADS: Treats each group of four vertices as an independent quadrilateral. Vertices  $4n-3$ ,  $4n-2$ ,  $4n-1$ , and  $4n$  define quadrilateral  $n$ .  $N/4$  quadrilaterals are drawn.

GL\_QUAD\_STRIP: Draws a connected group of quadrilaterals. One quadrilateral is defined for each pair of vertices presented after the first pair. Vertices  $2n-1$ ,  $2n$ ,  $n+2$ , and  $2n+1$  define quadrilateral  $n$ .  $N/2-1$  quadrilaterals are drawn. Note that the order in which vertices are used to construct a quadrilateral from strip data is different from that used with independent data.

GL\_POLYGON: Draws a single, convex polygon. Vertices  $1$  through  $N$  define this polygon.

## B.2. glVertex Function

### PARAMETERS

$x, y, z, w$ : Specify  $x, y, z,$  and  $w$  coordinates of a vertex. Not all parameters are present in all forms of the command.

### DESCRIPTION

glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When only  $x$  and  $y$  are specified,  $z$  defaults to 0.0 and  $w$  defaults to 1.0. When  $x, y,$  and  $z$  are specified,  $w$  defaults to 1.0.

## B.3. glLoadName Function

The name stack is used during selection mode to allow sets of rendering commands to be uniquely identified. It consists of an ordered set of unsigned integers. glLoadName causes *name* to replace the value on the top of the name stack, which is initially empty. The name stack is always empty while the render mode is not GL\_SELECT. Calls to glLoadName while the render mode is not GL\_SELECT are ignored.

## B.4. glSelectBuffer Function

```
void glSelectBuffer( GLsizei size, GLuint *buffer )
```

### PARAMETERS

*size*: Specifies the size of *buffer*.

*buffer*: Returns the selection data.

`glSelectBuffer` has two arguments: *buffer* is a pointer to an array of unsigned integers, and *size* indicates the size of the array. *buffer* returns values from the name stack when the rendering mode is `GL_SELECT`. `glSelectBuffer` must be issued before selection mode is enabled, and it must not be issued while the rendering mode is `GL_SELECT`. Selection is used by a programmer to determine which primitives are drawn into some region of a window. The region is defined by the current modelview and perspective matrices.

In selection mode, no pixel fragments are produced from rasterization. Instead, if a primitive intersects the clipping volume defined by the viewing frustum and the user-defined clipping planes, this primitive causes a selection hit. (With polygons, no hit occurs if the polygon is culled.) When a change is made to the name stack, or when `glRenderMode` is called, a hit record is copied to *buffer* if any hits have occurred since the last such event (name stack change or `glRenderMode` call). The hit record consists of the number of names in the name stack at the time of the event, followed by the minimum and maximum depth values of all vertices that hit since the previous event, followed by the name stack contents, bottom name first.

Returned depth values are mapped such that the largest unsigned integer value corresponds to window coordinate depth 1.0, and zero corresponds to window coordinate depth 0.0.

An internal index into *buffer* is reset to zero whenever selection mode is entered. Each time a hit record is copied into *buffer*, the index is incremented to point to the cell just past the end of the block of names - that is, to the next available cell. If the hit record is larger than the number of remaining locations in *buffer*, as much data as can fit is copied, and the overflow flag is set. If the name stack is empty when a hit record is copied, that record consists of zero followed by the minimum and maximum depth values.

Selection mode is exited by calling `glRenderMode` with an argument other than `GL_SELECT`. Whenever `glRenderMode` is called while the render mode is `GL_SELECT`, it returns the number of hit records copied to *buffer*, resets the overflow flag and the selection buffer pointer, and initializes the name stack to be empty. If the overflow bit was set when `glRenderMode` was called, a negative hit record count is returned.

### **B.5. `glRenderMode` Function**

`GLint glRenderMode( GLenum mode )`

*mode*: Specifies the rasterization mode. Three values are accepted: `GL_RENDER`, `GL_SELECT`, and `GL_FEEDBACK`. The default value is `GL_RENDER`.

`glRenderMode` sets the rasterization mode. It takes one argument, *mode*, which can assume one of three predefined values:

`GL_RENDER`: Render mode. Primitives are rasterized, producing pixel fragments, which are written into the frame buffer. This is the normal mode and also the default mode.

`GL_SELECT`: Selection mode. No pixel fragments are produced, and no change to the frame buffer contents is made. Instead, a record of the names of primitives that would have been drawn if the render mode were `GL_RENDER` is returned in a select buffer, which must be created before selection mode is entered.

`GL_FEEDBACK`: Feedback mode. No pixel fragments are produced, and no change to the frame buffer contents is made. Instead, the coordinates and attributes of vertices that would have been drawn had the render mode been `GL_RENDER` is returned in a feedback buffer, which must be created before feedback mode is entered.

The return value of `glRenderMode` is determined by the render mode at the time `glRenderMode` is called, rather than by *mode*. The values returned for the three render modes are as follows:

`GL_RENDER`: 0

`GL_SELECT`: The number of hit records transferred to the select buffer.

`GL_FEEDBACK`: The number of values (not vertices) transferred to the feedback buffer.

### B.6. `gluProject` Function

```
int gluProject( GLdouble objx, GLdouble objy, GLdouble objz, const GLdouble
modelMatrix[16], const GLdouble projMatrix[16], const GLint viewport[4],
GLdouble *winx, GLdouble *winy, GLdouble *winz )
```

#### PARAMETERS

*objx, objy, objz*: Specify the object coordinates.

*modelMatrix*: Specifies the current modelview matrix.

*projMatrix*: Specifies the current projection matrix.

*viewport*: Specifies the current viewport.

*winx, winy, winz*: Return the computed window coordinates.

`gluProject` transforms the specified object coordinates into window coordinates using *modelMatrix*, *projMatrix*, and *viewport*. The result is stored in *winx*, *winy*, and *winz*. A return value of `GL_TRUE` indicates success, and `GL_FALSE` indicates failure.

### B.6. `glReadPixel`() Function

```
void glReadPixels( GLint x, GLint y, GLsizei width, GLsizei height, GLenum
format, GLenum type, GLvoid *pixels )
```

## PARAMETERS

*x, y*: Specify the window coordinates of the first pixel that is read from the frame buffer. This location is the lower left corner of a rectangular block of pixels.

*width, height*: Specify the dimensions of the pixel rectangle. *width* and *height* of one correspond to a single pixel.

*format*: Specifies the format of the pixel data. The following symbolic values are accepted: `GL_COLOR_INDEX`, `GL_STENCIL_INDEX`, `GL_DEPTH_COMPONENT`, `GL_RED`, `GL_GREEN`, `GL_BLUE`, `GL_ALPHA`, `GL_RGB`, `GL_RGBA`, `GL_LUMINANCE`, and `GL_LUMINANCE_ALPHA`.

*type*: Specifies the data type of the pixel data. Must be one of `GL_UNSIGNED_BYTE`, `GL_BYTE`, `GL_BITMAP`, `GL_UNSIGNED_SHORT`, `GL_SHORT`, `GL_UNSIGNED_INT`, `GL_INT`, or `GL_FLOAT`.

*Pixels*: Returns the pixel data.

`glReadPixels` returns pixel data from the frame buffer, starting with the pixel whose lower left corner is at location  $(x, y)$ , into client memory starting at location *pixels*. Several parameters control the processing of the pixel data before it is placed into client memory. These parameters are set with three commands: `glPixelStore`, `glPixelTransfer`, and `glPixelMap`. This reference page describes the effects on `glReadPixels` of most, but not all of the parameters specified by these three commands. `glReadPixels` returns values from each pixel with lower left-hand corner at  $(x + i, y + j)$  for  $0 \leq i < \text{width}$  and  $0 \leq j < \text{height}$ . This pixel is said to be the *i*th pixel in the *j*th row. Pixels are returned in row order from the lowest to the highest row, left to right in each row.

## REFERENCES

1. Bustillo, J.E., Howe, R.T., "Surface Micromachining for Micro-electro-mechanical Systems", *Proceedings of the IEEE*, Vol. 86, No 8, Aug. 1998
2. Najafi, K., "Lecture Notes on Photolithography, Center for Wireless Integrated MicroSystems", *University of Michigan*
3. Madou, Marc J., "Fundamentals of Microfabrication", *CRC Press.*, Boca Raton, Fla, 1997
4. Kovacs, G.T.A., "Micromachined Transducers Sourcebook", *McGraw-Hill, Inc.*, New York, NY, 1998
5. Adams, A.C., "Dielectric and Polysilicon Film Deposition", *McGraw-Hill, Inc.*, New York, NY, 1983, pp. 93-129
6. Najafi, K., "Lecture Notes on Doping, Center for Wireless Integrated MicroSystems", *University of Michigan*
7. Williams, K.R., and Muller, R.S., "Etch Rates for Micromachining Processing", *Journal of Microelectromechanical System*, vol.5, no. 4, pp. 256-269, Dec. 1996
8. Bean, K. E., "Anisotropic Etching of Silicon", *IEEE Transactions on Electron Devices*, vol. ED-25, no. 10, pp. 1185-1193, Oct. 1978
9. Siedel, H., "The Mechanism of Anisotropic Silicon Etching and Its Relevance for Micromachining", *Proceedings of Transducers '87, Record of the 4<sup>th</sup> International Conference on Solid-State Sensors and Actuators*, Tokyo, Japan, pp. 120-125, June 2-5, 1987
10. Petersen, K.E., "Silicon as a Mechanical Material", *Proceedings of IEEE*, vol. 70, pp. 420-457, May 1982
11. Siedel, H., Csepregi, L., Heuberger, A., and Baumgärtel, H., "Anisotropic Etching of Crystalline Silicon in Alkaline Solutions II: Influence of Dopants", *Journal of the Electrochemical Society*, vol.137, no. 11, pp. 3626-3632, Nov. 1990
12. Tabata, O., "pH-Controlled TMAH Etchants for Silicon Micromachining", *Proceedings of Transducers '95/Eurosensors IX*, Stockholm, Sweden, 1995, vol. 1, pp. 83-86, June 25-29

13. Finne, R.N., and Klein, D.L., "A Water-Amine Complexing Agent System for Etching in Silicon", *Journal of Electrochemical Society*, vol. 114, no. 9, pp. 965-970, Sept. 1967
14. Jansen, H. de Boer, M., Legtenberg, R., and Elwenspoek, M., "The Black Silicon Method: A Universal Method for Determining the Parameter Setting of a Fluorine-Based Reactive Ion Etcher in Deep Silicon Trench Etching with Profile Control", *Journal of Micromechanics and Microengineering*, vol.5, no.2, pp. 115-120. , June 1995
15. Buser, R., A., and N. F. de Rooij, "ASEP: A CAD program for Si anisotropic etching", *Sensors Actuators*, vol. 28, 1991, pp. 71-78
16. Buser, R., A., and N. F. de Rooij, "Integration of the ASEP within the CAEMEMS CAD/CAE framework"
17. Li, G., Hubbard, T. and Antonsson, E. K., "SEGS: On-line WWW web etch simulator", *IEEE MSM'98*, Santa Clara, CA., pp. 420-457, May 1982
18. Buttgenbach, S. and Than, O., "SUZANA: A 3D CAD tool for anisotropically etched silicon microstructures", in Proc. European Design Test Conf., pp. 454-458, 1996
19. Marchetti, J., He, Y., Than, O., and Akkaraju S., "Efficient process development for bulk silicon etching using cellular automata simulation techniques," in SPIE Micromachined Devices Comp. IV Conf., Santa Clara, CA, pp. 287-295, Sept. 1998
20. IntelliSense Corporation, "AnisE User Manual", Introductory Tour, Version 2.5, February, 1998.
21. IntelliSense Corporation, "AnisE Internal Examples", 98AE13, 1998.
22. Zhu, Z. and Liu, C., "Anisotropic crystalline etching Simulation using a continuous cellular automata algorithm", *J. Computer Modeling Engineering Sciences*, vol. 1, no. 1, pp. 11-19, 2000
23. Zhu, Z. and Liu, C., "Micromachining Process Simulation Using a Continuous Cellular Automata Method", *Journal of Microelectromechanical Systems*, vol. 9, no. 2, June 2000
24. Gesellschaft für Mikroelektronikanwendung Chemnitz, "SIMODE Reference Manuel", Version 3.6, 2001
25. "<http://www.coventor.com/memulator/>", last viewed on Nov. 2005
26. Silvaco International, "Athena User Guide"

27. "<http://www.renard.org/alife>" last viewed on Nov. 2005
28. "<http://www.fourmilab.ch/cellab/>" last viewed on Nov. 2005
29. Von Neumann J. et Burks A. ed., "Theory of Self-Reproduction Automata", University of Illinois Press, 1966.
30. "OpenGL Programming Guide", Addison-Wesley Publishing Company, Chp. 11
31. "Report on KOH Process Module, Etch Characteristics and Design Guide", NCSU Nanofabrication Facility, January 25th, 2005
32. Nga P. Pham and Pasqualina M. Sarro, "High-aspect-ratio Bulk Micromachined Vias Contacts", ProcSAFE & Prorisc 2004, Veldhoven, pp. 742-746, Nov. 25-26, 2004
33. Sato, K. et al., "Characterization of orientation-dependent etching properties of single-crystal silicon: effects of KOH concentration", Sensors and Actuators A 64, pp. 87-93, 1988
34. Hull, R., "Properties of Crystalline Silicon", INSPEC, London, 1999
35. Shikida M., Sato K., Tokoro K., Uchikawa D., "Lecture Notes", Dept. of Micro Systems Engineering, Nagoya University, Japan