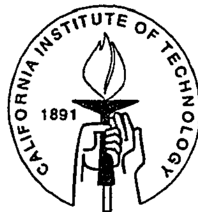
Capillary–Driven Reflow of Thin Cu Films with Submicron, High Aspect Ratio Features

Thesis by

Ruth A. Brain

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy



California Institute of Technology
Pasadena, California

1996
(Submitted July 7, 1995)

© 1996

Ruth A. Brain

All Rights Reserved

Acknowledgements

I have had the good fortune to work at JPL, Caltech, and Intel during my five years of graduate life. I would like to thank the many people with which I have had the pleasure to work.

At the Jet Propulsion Laboratory, I would like to thank my fellow graduate students, Ofer Iny and Colleen McDonough, for showing me the ropes. I would also like to thank Marty Barmatz for advising me, and John Gates and Dick Zanteson for expert engineering assistance.

I would like to thank many people in Components Research at Intel. Steve Sanz and Floyd Taylor always provided excellent technical assistance, with a smile. I would also like to thank my fellow co-op students at Intel, Ling Chen, Justin Gee, Marc Puich, and Hein Vu; I really enjoyed working with all of you. Ling, Justin, and Marc worked above and beyond the call of duty in providing SEM support at a moment's notice. I also received much help from the working group, which included Gang Bai, Neal Cox, Don Gardner, Anne Mack, Tom Marieb, and Xiao-Chun Mu, and had many fruitful discussions with Paul Flinn and Jick Yu. I would also like to thank Sally Lui and Sharon Kirk for their support in helping me coordinate my Caltech life and Intel life. I would especially like to thank Dave Fraser, he has been my second advisor during my many visits to Intel; I cannot thank him enough for his support of me and this work.

I have also had the pleasure of working with many people at Caltech. I have benefited greatly from the support of my group: Renato Camata, Heather Frase, Imran Hashim, Gang He, Hyun Sung Joo, Sue Melnik, Kyu Min, Kirill Shcheglov, Selmer Wong, and Jimmy Yang. I would especially like to thank Imran and Hyun for their help with the sputtering system at Caltech, and Selmer for her all around support and for reading rough drafts of my thesis. I would also like to thank Channing Ahn for his always useful and always amusing discussions, and his help with the sputtering

system. Carol Garland has also been a great help during all the TEM work; she taught me the wedge technique that is used for all the samples in this thesis, and she really made a difference. I also must thank Joe Fontana for engineering assistance. Finally, I certainly must thank my advisor, Harry Atwater. I have really enjoyed working with him–his enthusiasm for projects is infectious.

My family has played a big role in my education and in encouraging me to finish this thesis. They have always expressed interest in my work and have always expressed their conviction that I would succeed (Mark and Paul's encouragement often took the form of good-natured sarcasm and ridicule, of course). Thank you, Dad, Mother, Mark and Paul.

Last, but not least, I would like to thank my husband, Aaron. He was there during the good days, and the bad ones too, and he always had encouraging words to help me along. I seriously doubt that I would have finished graduate school without his love and support. I love you, Aaron.

I would also like to acknowledge financial support from Intel Corporation, Applied Materials, Jet Propulsion Laboratory, and the National Science Foundation.

Abstract

Conventional sputtering techniques are no longer sufficient for the fabrication of interconnects as trench widths enter the submicron regime and aspect ratios become greater than 1:1. The goal of this thesis is to investigate Cu as a potential interconnect metal for use in integrated circuit technology. Since sputtering is well established and widely used in the integrated circuit industry, we have used current sputtering technology as our deposition technique of choice. An alternative approach to modify the nonconformal deposition profiles obtained by sputtering is to reflow (planarize by capillary–driven surface diffusion) the metal film during a post-deposition anneal. In particular, reflow is performed for thin Cu films deposited on refractory metal barrier layers (Mo, Ta, and W) at temperatures $\leq 500°C$.

With a goal of developing and understanding a post-deposition reflow process for Cu, we have studied the following topics. Chapter 1 introduces relevant current concepts in ultra-large scale integration (ULSI) for interconnect technology to motivate the approaches described in this thesis. Chapter 2 investigates several techniques to improve the initial Cu coverage obtained from a magnetron sputtering source. This was found to be necessary since thin Cu films agglomerate on many underlayers, and this constrains the initial Cu thickness requirements for successful reflow. Chapter 3 describes an investigation of the atomic transport mechanism during reflow of Cu films, to understand the kinetics of reflow and measure the appropriate kinetic constants. Extensive transmission electron microscope (TEM) work was done to examine reflowed profiles and to relate the extent of reflow to the morphology, texture, grain size, and orientation of the Cu films. Hot-stage TEM experiments were performed to observe dynamically the reflow of a very low aspect ratio film. Chapter 4 develops a finite-element model to study surface diffusion mediated reflow in high aspect ratio trenches. We have considered (i) reflow of typical continuum, as-deposited profiles from a magnetron sputtering source, (ii) reflow of continuum profiles including an

anisotropic surface energy, and (iii) reflow with the inclusion of grain boundaries. We also discuss some limitations of a post-deposition reflow process, and we make recommendations to facilitate the ability to reflow Cu in high aspect ratio trenches. Chapter 5 examines a non-infrared annealing technique to reflow Cu films. In particular, we have examined the possibility of annealing Cu films in a single-mode microwave cavity, which can be advantageous because it is possible to thermally isolate the substrate. Chapter 6 provides a summary of the work in this thesis and suggests some possibilities for future work.

List of Publications

"On the Surface Diffusion–Mediated Planarization of Thin Films: Surface Energy Anisotropy and Grain Boundaries," R.A. Brain and H.A. Atwater, Submitted to Mat. Res. Soc. Symp. Proc., Fall 1995.

"Reflow of Polycrystalline Cu Films. I. Experiment," Ruth A. Brain, Donald S. Gardner, David B. Fraser, Harry A. Atwater, Submitted to Journal of Materials Research.

"Reflow of Polycrystalline Cu Films. II. Model," Ruth A. Brain and Harry A. Atwater, Submitted to Journal of Materials Research.

"Ballistic Deposition and Consolidation of Al Clusters into High Aspect Ratio Trenches," J. Reid, R. Brain, C. Ahn, Twelfth International VLSI Multilevel Interconnection Conference Proceedings, 545 (1995).

"The Effect of Grain Boundaries on Surface Diffusion–Mediated Planarization of Polycrystalline Cu Films," R.A. Brain, H.A. Atwater, D.S. Gardner, D.B. Fraser, Submitted to Mat. Res. Soc. Symp. Proc., Spring 1995.

"Rapid Selective Annealing of Cu Thin Films on Si Using Microwaves," R.A. Brain, H.A. Atwater, M. Barmatz, Mat. Res. Soc. Symp. Proc. **347**, 519(1994).

Contents

Acknowledgements iii

Abstract v

List of Publications vii

List of Figures xi

List of Tables xvii

1 Introduction 1

1.1 The Problem 1

1.2 Metalization Requirements 2

1.2.1 Scaling 2

1.2.2 Aluminum versus Other Metals 3

1.3 Overview of Sputtering Technology 6

1.4 Aluminum Reflow 8

1.5 Goals of this Thesis 9

2 Sputtering Techniques to Improve Initial Coverage and Filling 11

2.1 UHV Magnetron Sputtering System 11

2.2 Initial Attempts to Reflow Cu 13

2.3 Agglomeration of Thin Films 16

2.4 Improving Initial Trench Coverage 18

2.4.1 Target Geometry 20

2.4.2 Substrate Bias 20

2.4.3 Additional Sputtering Parameters and Techniques 22

2.5 Summary 27

3 Cu Reflow on Refractory Metal Barrier Layers 28

3.1 Planarization Mechanisms 28

3.2 Reflow of Polycrystalline Cu Films 30

3.2.1 Texture Scans 37

3.2.2 TEM Cross-sections of Reflowed Cu Films 44

3.3 Reflow of Low Aspect Ratio Polycrystalline Cu Films 56

3.4 Hot-stage TEM of Cu films 58

4 Modeling Cu Planarization 64

4.1 Modeling Surface Diffusion–Mediated Reflow 65

4.1.1 Some Considerations 66

4.1.2 Results 70

4.2 Modeling Reflow with Surface Energy Anisotropy 74

4.2.1 Choosing the Form of $\gamma_s(\theta)$ 76

4.2.2 Results 78

4.3 Modeling Reflow with Grain Boundary/Surface Interactions 84

4.3.1 Grain Boundary Intersecting an Otherwise Flat Surface 84

4.3.2 Grain Boundary Intersecting a General Surface 87

4.3.3 Some Considerations 88

4.3.4 Reflow Including a Single Grain Boundary 89

4.3.5 Reflow Including Multiple Grain Boundaries 90

4.4 Comparison with Experiment 94

4.5 Additional Considerations Not Modeled 98

5 Rapid Selective Annealing of Cu Thin Films on Si using Microwaves 102

5.1 Thin Film Microwave Annealing Experiments 102

5.2 Results and Discussion 103

5.2.1 Silicon Heating Curves 103

5.2.2 Cu Reflow 109

5.3 Summary 113

6 Summary and Future Work **114**

Bibliography **119**

A Algorithm for Surface Diffusion–Mediated Reflow **124**

A.1 Input Parameters 124

A.2 Subroutines 124

A.3 Code 125

A.3.1 reflow.h 125

A.3.2 reflow.cpp 126

B Algorithm for Reflow with an Anisotropic Surface Energy **138**

B.1 Subroutines 138

B.2 Code 138

B.2.1 reflowan.h 138

B.2.2 reflowan.cpp 139

C Algorithm for Reflow with Grain Boundary/Surface Interactions **152**

C.1 Subroutines 152

C.2 Code 153

C.2.1 reflowgb.h 153

C.2.2 reflowgb.cpp 153

List of Figures

1.1 Interconnects formed by a Damascene process. In step (1), SiO_2 followed by photoresist are deposited on the substrate, and the photoresist is patterned. In step (2), the dielectric is etched back to the Si to form trenches and then the resist is removed. In step (3), the metal is deposited on the dielectric. In step (4), the metal is polished back by CMP to the height of the underlying dielectric. 5

2.1 Wafer temperature calibration for J-type thermocouple bead and heater shield thermocouple with the wafer facing either the closed shutter of the conical target or the opposite window. 12

2.2 0.5 μm Cu/Mo on a 0.5 μm by 0.5 μm trench that has been reflowed for 30 minutes at (a) 500°C and (b) 300°C. 14

2.3 (a) As-deposited 0.5 μm Cu/Si_3N_4 on a 0.5 μm by 1.0 μm trench. (b) 0.5 μm Cu/Mo on a 0.5 μm by 1.0 μm trench, annealed for 5 minutes at 500°C. 15

2.4 Agglomeration of a uniform 2-dimensional thin film with initial thickness t and grain size D. 17

2.5 500 Å Cu deposited on an (a) *in situ* and (b) *ex situ* W barrier layer and annealed at 500°C for 30 minutes. 19

2.6 1.0 μm Ta film deposited from the planar target in (a) a 1.0 μm by 1.0 μm trench and (b) a 0.5 μm by 1.0 μm trench. 21

2.7 Cu deposited on SiO_2 with a -60 V substrate bias on a (a) 1.0 μm by 1.0 μm trench and (b) 0.5 μm by 1.0 μm trench. Result was not reproducible. 23

2.8 -60 V substrate biased Cu/Ta on trenches with widths of (a) 0.86 μm, (b) 0.59 μm, (c) 0.48 μm and (d) 0.38 μm that was reflowed for 60 minutes at 500°C. 25

3.1 Wafer 06 (Cu(25°C)/Ta, reflowed 2 hours at 500°C). Approximately 1% of the trenches are filled. 32

3.2 Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). Approximately 20% of the trenches are filled. 33

3.3 Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). Approximately 80% of the trenches are filled. 34

3.4 Wafer 14 (Cu(25°C)/W, reflowed at 500°C for 17 hours). There is some trench filling everywhere. 35

3.5 Wafer 16 (Cu(150°C)/W, reflowed at 500°C for 11 hours). Less than 1% of the trenches are filled. 36

3.6 Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak. 39

3.7 Wafer 08 (Cu(25°C)/Ta, as-deposited). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak. 40

3.8 Wafer 09 (Cu(150°C)/Ta, annealed at 250°C for 30 minutes). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak. 41

3.9 Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak. 42

3.10 Wafer 08 (Cu(25°C)/Ta, as-deposited). (a) Bright field (BF) image of an as-deposited 0.8 μm by 1.0 μm trench and (b) SAD pattern using a 0.5 μm aperture. Typical grain size is on the order of 500 - 1000 Å. 45

3.11 Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). (a) BF image of a filled 0.8 μm by 1.0 μm trench. (b) Dark field (DF) image of grain in flat portion between trenches. (c) BF image of an unfilled, agglomerated trench. (d) BF image of a groove – anti-groove pair. . . 46

3.12 Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). (a) BF image of a filled 1 μm by 1 μm trench. (b) DF image of (a). (c) SAD from single grain in trench. (d) SAD from single grain above trench shoulder to the left of trench. 47

3.13 Wafer 11. (a) BF image of a filled 1 μm by 1 μm trench that had grains stacked from the bottom of the trench upwards. (b), (c) and (d) BF images of unfilled trenches. Note the faceting within single grains, and that the grain boundaries are often perpendicular to the surface. . . . 48

3.14 Wafer 11. (a) BF image of the flat region between trenches. (b) BF image of an "avalanched" region near a filled trench. 49

3.15 Wafer 11. BF image of a faceted grain and the SAD image from the grain indicating the orientation of the surfaces. 50

3.16 Wafer 11. (a) BF image of a groove – anti-groove pair and micro-diffraction images from the grains on each side of the anti-groove. (b) and (c) Groove – anti-groove pairs along trench sidewalls. 51

3.17 Wafer 14 (Cu(25°C)/W, reflowed at 500°C for 17 hours). (a) BF image of a Cu_xSi_{1-x} grain on the surface of the Cu film. (b) EDX beam marks in the SiO_2 shows Cu had diffused into the SiO_2. W diffusion barrier was insufficient for this long anneal. Cu has diffused through the W and the Si_3N_4 into the SiO_2, and Si has diffused into the Cu. 52

3.18 Wafer 16 (Cu(150°C)/W, reflowed at 500°C for 11 hours). (a) BF image of a 1 μm by 1 μm trench where the Cu has agglomerated on the sidewall. (b) BF image of the flat region between trenches. Most grains are columnar and the film surface is relatively rough. (c) and (d) BF images of trenches that have not agglomerated. Typical trenches were approximately half full. 53

3.19 Ion-beam sputtering system. The glass tube to the right of the picture contains the sample during the optical diffraction experiments. 57

3.20 Diffraction results for a very low aspect ratio film annealed at 505°C. The normalized n = 1 intensity is increasing during the anneal, and the normalized n = 3 intensity changes slope during the anneal. . . . 58

3.21 Plan-view of the 1000 Å Cu/250 Å W sample before annealing. The trenches are 1 μm wide and are 6 μm apart. The black "islands" in the trenches are regions of Cu agglomeration due to sample heating during preparation. 60

3.22 The sample temperature has been ramped to 340°C in 15-20 minutes. The island in (a) is the lower right island in (b). The appearance of the islands in (b) took place within 5 seconds. 61

3.23 Sample after being ramped to 440°C over the course of 1 hour. 63

4.1 SEM image of an as-deposited Cu film from a conical magnetron sputtering system that was digitized and used as the initial profile in these simulations. 69

4.2 Simulation of Cu reflow at 800 K for (a) 1 μm by 1 μm trench with t_{fill} = 68 minutes; (b) 0.5 μm by 1 μm trench with t_{fill} = 17 minutes; (c) 0.33 μm by 1 μm trench with t_{fill} = 8 minutes; (d) three 0.5 μm by 1 μm trenches that are spaced by 0.5 μm apart with t_{fill} = 19 minutes. 72

4.3 (a) 0.2 μm thick conformal film on a 0.5 μm by 1.0 μm trench with t_{fill} = 3.5 minutes; (b) reflow of profile from a highly collimated target on a 1 μm by 1 μm trench with t_{fill} = 1.9 minutes. 73

4.4 A surface with an anisotropic surface energy will have a Wulff plot with $\gamma_1 < \gamma_2$ where γ_1 is the surface energy for a low index surface. In this case, the surface can lower its energy by faceting even though the surface area is increased. 76

4.5 (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta) + \frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ using $\gamma_s(111) = \gamma_s(100) = \gamma_s(110) = 1800$ ergs/cm^2, $\alpha = 0.012$, $\beta = 7$ and $G = 225$. (c) Reflow of a 0.5 μm by 1 μm trench, assuming an anisotropic surface energy as shown in (a) and (b). Note the formation of facets along the y-axis and at 54.7° from the y-axis, and the suppression of tails on the outside of the shoulders near the top of the trench. t_{fill} = 27 minutes. 79

4.6 (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta)+\frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ using $\gamma_s(111) = 1789.2$ ergs/cm^2, $\alpha(111) = 0.012$, $\beta(111) = 7$, $\gamma_s(100) = 1800$ ergs/cm^2, $\alpha(100) = 0.007$, $\beta(100) = 5$, and $G = 225$. (c) Reflow of a 0.5 μm by 1.0 μm trench assuming an anisotropic surface energy as shown in (a) and (b). t_{fill} = 24 minutes. 80

4.7 (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta)+\frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ using $\gamma_s(111) = 1789.2$ ergs/cm^2, $\alpha(111) = 0.012$, $\beta(111) = 7$, $\gamma_s(100) = 1800$ ergs/cm^2, $\alpha(100) = 0.007$, $\beta(100) = 5$, and $G = 144$. (c) Reflow of a 0.5 μm by 1.0 μm trench assuming an anisotropic surface energy as shown in (a) and (b). t_{fill} = 22 minutes. 82

4.8 Reflow of a 0.5 μm by 1.0 μm trench, assuming $\gamma_s = 1800$ ergs/cm^2, $\alpha = 0.024$, $\beta = 7$, and $G = 225$. 83

4.9 A grain boundary groove forming on a surface initially at y = 0. The grain boundary is along the negative y axis. 85

4.10 An idealized grain boundary groove. In case **a**, the grain boundary may escape the notch, while in case **b** the grain boundary anchored to the notch. The effectiveness of the notch as an anchor depends only upon θ_c, and not on the depth of the groove. 86

4.11 (a) Grain boundary position on the initial 1 μm by 1 μm profile and (b) the resulting, normalized reflow times for a grain boundary with $\gamma_{gb} = 300$, 600, or 900 ergs/cm^2. The normalized reflow time is the reflow time with inclusion of the grain boundary as indicated in (a) divided by the reflow time for the same profile without a grain boundary. 91

4.12 Grain boundary position as the profile evolves for a 1 μm by 1 μm trench with $\gamma_{gb} = 600$ ergs/cm^2. The grain boundary position is indicated by a o. 93

4.13 Reflowed profiles assuming $\gamma_{gb} = 600$ ergs/cm^2 for both grain boundaries. The symbols indicate the position of the grain boundary on the reflowing profile. 96

5.1 Cross-sectional view of the microwave cavity from top. The antenna is sealed with an o-ring and is adjustable to enable critical coupling to the cavity. Above and below the quartz sample support are sealed quartz windows to enable temperature measurements with an optical pyrometer. 104

5.2 (a) Q_{loaded} versus input power and (b) temperature versus input power for Si with resistivity $\rho = 10.8$ Ω·cm. These are values taken after the temperature of the Si had stabilized at the given power level. 106

5.3 (a) Q_{loaded} versus input power and (b) temperature versus input power for Si with resistivity $\rho = 1000$ Ω·cm. The arrows indicate whether the input power was being ramped up or down for a given branch of the heating curve. These are values taken after the temperature of the Si had stabilized at the given power level. 107

5.4 Cu films on patterned SiO_2 before microwave annealing. The trench depth is 0.8 μm and the trench widths range from 0.3 μm to 1.2 μm. Note that the trench filling decreases as the aspect ratio increases. . . 110

5.5 Cu films on patterned SiO_2 after microwave annealing. The trenches are now well filled. We believe that the surface cracks are due to the rapid cooling of the samples after annealing. 111

List of Tables

2.1 The results of annealing thin Cu films on *in situ* and *ex situ* refractory metal underlayers to test for agglomeration. 18

3.1 Reflow results for 0.75 μm Cu films deposited with the substrate at room temperature or 150°C, on Ta and W barrier layers. In this table, "20% filling" means that 20% of the trench lengths had reflowed completely or almost completely, and 80% had reflowed negligibly. The base pressure for the depositions on Ta was $< 5 \times 10^{-10}$ Torr, while the base pressure for depositions on W was $\leq 5 \times 10^{-9}$ Torr. 30

3.2 Summary of $\theta - 2\theta$ x-ray diffraction and rocking curve results for Cu films deposited on Ta. 43

Chapter 1 Introduction

1.1 The Problem

Advances in integrated circuit technology have driven a reduction in the minimum feature size to enable increases in the circuit speed and integration density. This trend has placed severe constraints on interconnect technology because the cross-sectional area of the interconnection decreases as the device dimension is reduced, while the length of the interconnection increases as the chip size grows. In order to minimize interconnect capacitance, interconnection levels are separated by dielectric layers of increasing thickness, which requires the formation of continuous metal lines over very high aspect ratio (= height:width) microstructures. Also, more stringent requirements are being placed on the interconnect materials themselves, both to lower the electrical resistance to increase speed, and to increase their electromigration resistance to improve reliability.

In recent years, interconnections have become increasingly important as they have become the limiting factor in performance, manufacturing yield, reliability, and scaling in integrated circuit technology. Aluminum or aluminum alloys have long been the material of choice for interconnects due to its low resistivity, compatibility with silicon processing, excellent adhesion to dielectrics, and self-limiting oxide. As line widths enter the submicron regime, however, Al-based metalization faces several critical problems related to performance and reliability, including (i) high line resistance, which limits speed; (ii) poor electromigration resistance, which limits the current carrying capability; and (iii) susceptibility to stress-induced voiding, which shortens lifetime.

1.2 Metalization Requirements

Ideally, metalization used in integrated circuits should have the following properties:

- low electrical resistivity
- high electromigration resistance
- high resistance to stress-induced voiding
- resistance to oxidation and corrosion
- good adhesion to barrier layers and/or dielectrics
- ease of patterning by plasma etching
- good thermal stability
- low cost

Unfortunately, we don't live in utopia, and no single material exhibits all of these properties. Certain trade-offs will have to be made and we will discuss these in the following sections.

1.2.1 Scaling

As integrated circuits are scaled to ever smaller design rules, the challenge for metalization becomes formidable. The signal propagation delay, τ, caused by interconnections is $\tau = RC$, where R is the interconnect resistance and C is the dielectric capacitance. The interconnect resistance is defined as

$$R = \rho \frac{L}{Wt}, \tag{1.1}$$

and the dielectric capacitance is defined as

$$C = LW \frac{\epsilon k_{ox}}{t_{ox}}, \tag{1.2}$$

where L is the interconnection length, W is the interconnection width, t is the interconnect thickness, k_{ox} is the dielectric constant, and t_{ox} is the dielectric thickness.

As new generations of integrated circuits are introduced, the device size on the chip is scaled by $1/S$, and simultaneously, the chip size is scaled by S_c, where S and S_c are greater than one. The global interconnection resistance, R_g, increases by the factor S^2S_c and the global capacitance, C_g, increases by the factor S_c; hence, the interconnection delay increases as $\tau_g \sim (SS_c)^2$. For the local interconnects, R_ℓ increases by a factor S, while C_ℓ decreases by a factor $1/S$, so that $\tau_\ell \sim 1$[1].

It is advantageous to use a different scaling law for the metalization than that used for the devices. In particular, if we scale all lateral dimensions by $1/S$ and maintain a constant interconnect thickness for the local interconnects, then $\tau_\ell \sim 1/S$[1]. Hence, we can decrease the propagation delay for local interconnects by going to higher aspect ratio lines. The difficulty then becomes either making high aspect ratio lines or filling high aspect ratio trenches, and this is the crux of the problem discussed in this thesis. For global interconnects, where the line density is not as important, it is clearly advantageous to use wider lines whenever this option is available. Thus, the problem of creating high aspect ratio lines is most critical for "metal 1" in multilevel interconnects. It is also clear from Equations 1.1 and 1.2 that simply lowering the resistivity of the interconnect metal and using dielectrics with lower dielectric constants results in an immediate increase in speed.

1.2.2 Aluminum versus Other Metals

Aluminum metalization is an old friend of the integrated circuit industry due to its relatively low electrical resistivity and compatibility with silicon processing. Al-based metalization has not changed much in the past few years except to add $\sim$ 1.0% Si to prevent spiking of aluminum into the silicon, and to add $\sim$ 0.5% Cu to improve interconnect reliability. The addition of silicon eliminates interdiffusion of silicon and aluminum during high temperature processing[2]. Copper is added to aluminum to improve reliability, since the Cu segregates to the grain boundaries and

lowers grain boundary diffusion, which can lead to void formation if not checked; the addition of copper also retards stress voiding[3]. The resistivity of bulk aluminum at room temperature is 2.7 $\mu\Omega$·cm; however, the aluminum alloys commonly used have resistivities from 3.0 $\mu\Omega$·cm to 3.5 $\mu\Omega$·cm.

Layered metalization composed of Al alloy lines and thermally stable refractory metal layer (W, in particular) are designed to prevent hillock formation[4] and various contact-related problems[5, 6]. Silicides and refractory metals have been introduced to supplement, not replace, Al metalization. W has become almost the universal choice as the via (vias are "plugs" that run between interconnection levels or between interconnection levels and device regions) filler material because of the relative ease with which selective chemical vapor deposition of tungsten is achieved.

The main requirement for a metal to replace aluminum is electrical resistivity. Ag (ρ = 1.63 $\mu\Omega$·cm), Cu (ρ = 1.69 $\mu\Omega$·cm), and Au (ρ = 2.24 $\mu\Omega$·cm) all have lower bulk resistivities than Al (ρ = 2.67 $\mu\Omega$·cm); however the difference between Au and Al is not large enough to justify the investment for a transition to a new metal, and Au is a major recombination–generation center in Si, forms a low temperature eutectic with Si and is very difficult to pattern by plasma etching[7]. Thermal stability and electromigration resistance are also primary considerations for a metal choice. Generally, metals with a high melting point are thermally stable and metals with large atomic weights are more resistant to electromigration. Ag is the best choice if considering resistivity alone; however, the agglomeration behavior of thin Ag films makes it unlikely to be compatible with Si processing[8]. Hence, Cu seems to be the best choice for future metalization schemes.

Copper metalization has many difficulties to overcome. It corrodes and oxidizes easily; it does not have a self-passivating oxide; it has a high diffusivity in Si and SiO_2, and it adheres poorly to SiO_2. It is also very difficult to dry etch, however, chemical-mechanical polishing (CMP) has been demonstrated to be a viable etchback technique for Cu[9, 10]. In this technique, a Cu film is deposited on pre-etched trenches in the dielectric, then the metal is polished back using a pad infused with a chemical slurry as shown in Figure 1.1. Cu also offers up to three orders of magnitude

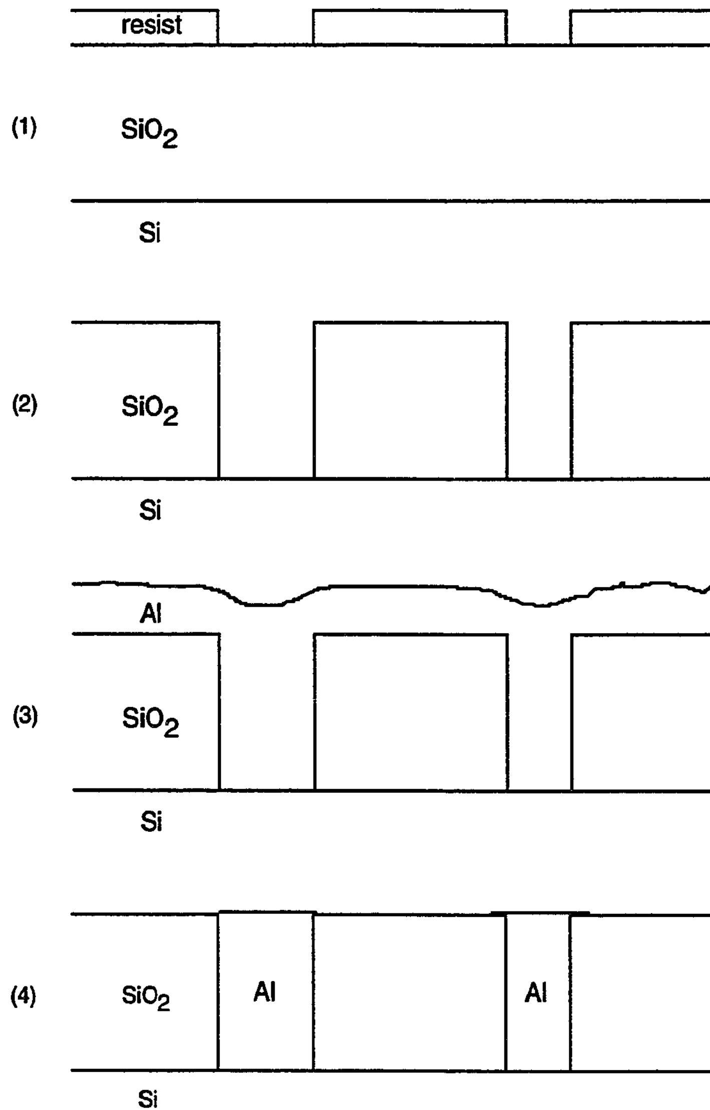


Figure 1.1: Interconnects formed by a Damascene process. In step (1), SiO_2 followed by photoresist are deposited on the substrate, and the photoresist is patterned. In step (2), the dielectric is etched back to the Si to form trenches and then the resist is removed. In step (3), the metal is deposited on the dielectric. In step (4), the metal is polished back by CMP to the height of the underlying dielectric.

better electromigration lifetime than aluminum alloys[11, 12].

1.3 Overview of Sputtering Technology

Thin film deposition occupies an important place among the various applications of ionized beams. Surfaces of solids erode under particle bombardment, a phenomenon first observed in gas discharges in the middle of the last century. Thin films of a given material can be grown by sputtering a target material onto a designated substrate. There are two basic types of sputtering—physical sputtering and chemical (or reactive) sputtering[13]. In physical sputtering, the incoming ions transfer kinetic energy to target atoms which are subsequently ejected if they have enough energy to overcome the binding forces exerted by the target. In chemical sputtering, a chemical reaction is induced by the impinging ions which produces an unstable compound at the target surface. The number of atoms ejected from the target material per incoming ion is called the sputtering yield. This thesis focuses on physical sputtering because we sputtered metals by inert gas (Ar^+) bombardment.

Many possible sputtering configurations are currently used. In this thesis, both magnetron and ion-beam sputtering were employed. In magnetron sputtering, an intense plasma (typically an Ar plasma) region is ignited near the target cathode by applying a high voltage to the cathode, and an $\mathbf{E} \times \mathbf{B}$ field configuration is used to enhance the plasma interaction with the target surface. In this configuration, the target is formed into a torus and acts as the cathode. An anode plate is placed in the center of the torus, which is grounded. To prevent the plasma from interacting with the sidewalls and anode, they must be well grounded so that there is no sputtering of sidewall or other material which would contaminate the film[14]. Another method of sputtering materials, which is not used extensively in the integrated circuits industry, is ion-beam sputtering. In this case, the ion gun consists of a discharge chamber filled with Ar gas and a W filament as a source of electrons which are accelerated to the anode. The presence of cylindrical magnets along the walls of the discharge chamber forces the electrons into helical orbits, increasing their ionization cross section and

creating an Ar plasma. A screened grid which has the same potential as the cathode allows some of the ions from the plasma to escape and these ions are then accelerated by a 1-2 kV potential. This ion beam can then be focused on a target that is some distance away.

Sputtering is a complicated process, and the sputtering yield depends on many variables, such as the incoming ion energy and the mass ratios of the target atoms to incoming ions. For an ion energy range from 0.1 to 20 keV, the ions generate a collision cascade near the target surface, possibly leading to the ejection of target atoms. As the ion energy is increased from zero, the sputtering yield increases (after overcoming the work function of the solid), but at energies higher than 20 keV, the ions penetrate deeper into the solid resulting in a decrease in sputtering yield, and increased ion implantation. The average energy of a sputtered atom ranges from 5 - 10 eV[15], while for atoms in the physical vapor deposition (PVD) process of evaporation have energies of only a few tenths of an eV[16].

It is also important to realize that the variables that affect the sputtering process also affect the deposited film morphology, crystal orientation, grain size, deposition profile, etc. For example, it is not currently possible to determine *a priori* whether a sputtered film deposited on a given underlayer will have tensile or compressive stresses[15]. The details of the film properties will be determined by such parameters as the energy of the sputtered particles hitting the film surface, the percent of impurities that are implanted in the film (for instance, the sputtering gas), and the interaction of the plasma with the substrate.

Conventional PVD techniques, such as sputtering and evaporation, have long been used in interconnect metalization. These techniques, however, do not provide a conformal deposition profile, which is essential to the multilevel metalization in high density integrated circuits as trench widths enter the submicron regime and aspect ratios become greater than 1:1. In this regime, conventional PVD techniques alone will no longer fill the via and trenches between interconnect levels and devices. Typical profiles of metal films deposited on submicron width trenches show incomplete filling of trenches due since the sputtered atoms do not follow a path which is normal to the

substrate. The lack of a conformal deposition profile motivated the research described in this thesis.

1.4 Aluminum Reflow

Aluminum is currently the metal of choice for the integrated circuits industry. To fill the high aspect ratio trenches that are needed in future integrated circuit designs, new sputtering and planarization techniques are needed. High temperature Al sputtering and reflow have been investigated as promising solutions for achieving planarization and complete trench and via filling. It is extremely desirable to continue to use a sputtering-based technology for metalization because it is a relatively simple process, it has low cost of ownership, the workforce has much experience, it lacks environmentally hostile precursors and by-products, and there has already been a large capital investment made by the industry.

There have been promising results for reflowed Al films at temperatures as low as 450°C[17, 18]. The key factors for reliable trench filling seem to be: low base pressure, adequate coverage of the wetting layer, and continuous coverage of the first Al layer in the contacts. It has been demonstrated that contact holes of 0.25 μm by 1.2 μm can be completely filled using a two step sputtering process where a thin Al layer is deposited near room temperature and then a second layer of Al is deposited at 400°C to 500°C. The Ti wetting layer used in this study seemed to be extremely important to successful reflow[17]. Another group[18] has found that Al(1.0%)Si(0.5%)Cu can be reflowed successfully on both Ti and $TiSi_{2.4}$, but that successful reflow could be achieved at a lower temperature (430°C) by using a $TiSi_{2.4}$ underlayer. These results are currently understood at only a phenomenological level, however Al reflow is being seriously considered as a production-worthy process by many groups.

1.5 Goals of this Thesis

Given that all computer users constantly desire increased computer speed and power for their money, integrated circuit technology clearly will be driven to smaller critical dimensions in order to put more devices on a piece of silicon at lower cost. This trend has placed severe constraints on interconnect technology, which has become the limiting factor in performance, manufacturing yield, reliability, and scaling.

We will investigate Cu as a potential metal for use in integrated circuit technology. Since sputtering is established in the industry, we will use current sputtering technology as our deposition technique of choice. A possible solution to the nonconformal deposition profiles obtained by sputtering is to reflow (planarize by capillary–driven surface diffusion) the metal film during a post-deposition anneal; the feasibility of this method has been demonstrated for Al films. In this thesis, we have examined the possibility of reflowing Cu at low temperatures as a possible solution to the nonconformal deposition profiles for use in advanced interconnects. In particular, we will reflow thin Cu films deposited on refractory metal barrier layers (Mo, Ta, and W) at temperatures $\leq 500°C$. There are certainly many other possible barrier layers for Cu being investigated[19]; however, to understand the reflow process it is useful to use refractory metal barrier layers since their solid solubility in Cu is $< 1\%$[20]—we wish to study the "pure" system. Cu/Ta/Si and Cu/W/Si multilayers have been shown to retain their multilayer structures after annealing at 600°C for 1 hour in H_2 without resistivity increases[21].

With a goal of developing an understand of a post-deposition reflow process for Cu, we have studied the following topics. In Chapter 2, we investigated several techniques to improve the initial Cu coverage obtained from a magnetron sputtering source. This study is necessary because thin Cu films agglomerate on many underlayers, leading to minimum initial Cu thickness requirements for successful reflow. In Chapter 3, we attempt to verify the transport mechanism for Cu films to understand the kinetics of reflow and gain the appropriate material constants. We also performed extensive transmission electron microscopy (TEM) work to examine the profiles that have been

reflowed and to understand the dependence of reflow on the morphology of the Cu films. Also, we completed hot-stage TEM experiments to investigate the reflow dynamics of a very low aspect ratio film. In Chapter 4, we develop a finite-element model of surface diffusion mediated reflow in high aspect ratio trenches. Here, we studied the effects of an isotropic versus an anisotropic surface energy, as well as the inclusion of grain boundaries that are typically seen in a sputtered film. We also discuss some of the limits of a post-deposition reflow process and make recommendations to improve the ability to reflow Cu in high aspect ratio trenches, however many of the results may be applicable to Al as well. In Chapter 5, we examine a non-infrared annealing technique to reflow Cu films. In particular, we examined the possibility of annealing Cu films in a single-mode microwave cavity. This technique can be advantageous since it becomes possible to selectively anneal the Cu film with relatively minimal heating to the substrate, thus reducing the taxation of the "thermal budget." Finally, Chapter 6 provides a summary of this thesis and suggests some possibilities for future work in this area.

Chapter 2 Sputtering Techniques to Improve Initial Coverage and Filling

2.1 UHV Magnetron Sputtering System

The films described here were grown in a custom-designed ultrahigh vacuum magnetron sputtering system[22]. The system includes a load-lock chamber with base pressures in the low 10^{-6} Torr range, a transfer chamber with base pressures in the low 10^{-9} Torr range, and the main deposition chamber with ultimate base pressures as low as 3×10^{-11} Torr. Unless otherwise noted, all films in this work were deposited at base pressures $< 5 \times 10^{-10}$ Torr. The main chamber is pumped with a helium cryopump, a titanium sublimation pump, and a liquid nitrogen cold stage; all seals in the main chamber are metal. The main chamber contains two sputtering targets, a planar refractory metal target (either Mo, Ta, or W) and a conical (S-gun) Cu target[2]. The main chamber is also equipped with a quadrupole mass spectrometer to analyze the residual gases, and the substrate can be heated *in situ* so that deposition and annealing of a barrier layer and the Cu film can be accomplished without a vacuum break.

Several parameters determine the purity of a sputtered film besides the base pressure, including Ar purity, target purity (both solid and residual gas impurities), deposition rate, and substrate outgassing. Ultra-pure Ar was further purified with a titanium getter before introduction into the sputtering chamber; the residual impurities in the Ar were < 50 ppb. The targets used in this system were carefully chosen from those commercially available to have extremely low solid impurities and residual gases, hence they were of 99.9999% purity. The quest for this level of purity typically requires the target to be cast in a high vacuum environment. Also, the magnetron sputtering system is capable of deposition rates greater than 1 μm/minute. During

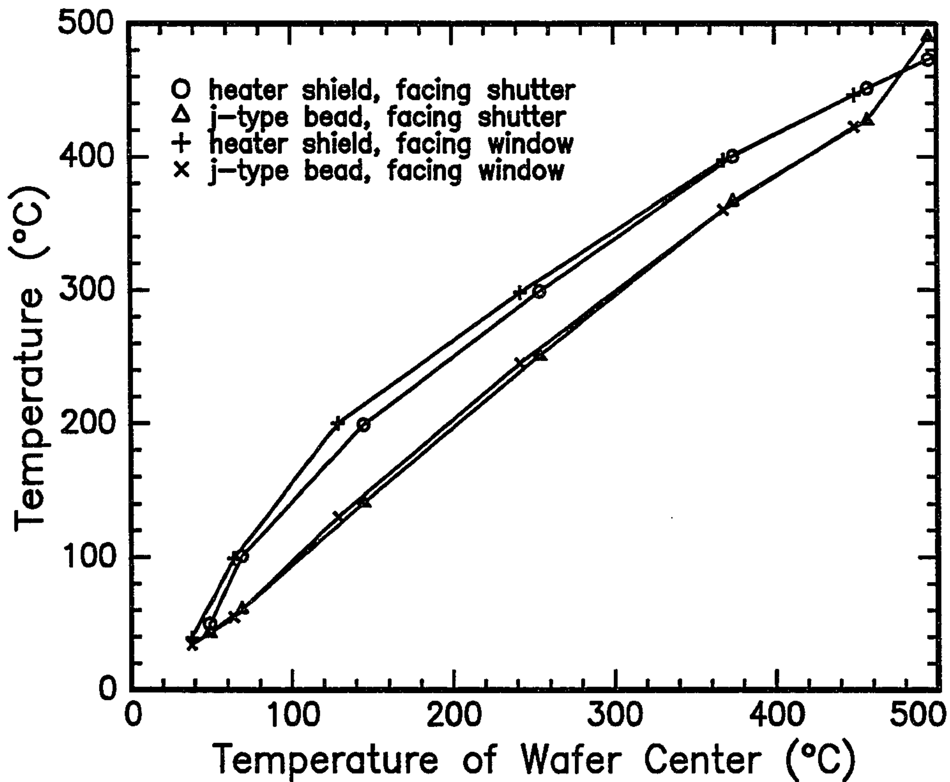


Figure 2.1: Wafer temperature calibration for J-type thermocouple bead and heater shield thermocouple with the wafer facing either the closed shutter of the conical target or the opposite window.

film deposition the cryopump is not throttled, so the system has a true base pressure of $< 5 \times 10^{-10}$ Torr. Before any depositions, the wafer was loaded into the main chamber and baked for 30 minutes at 500°C and then allowed to cool to avoid any outgassing during deposition[22].

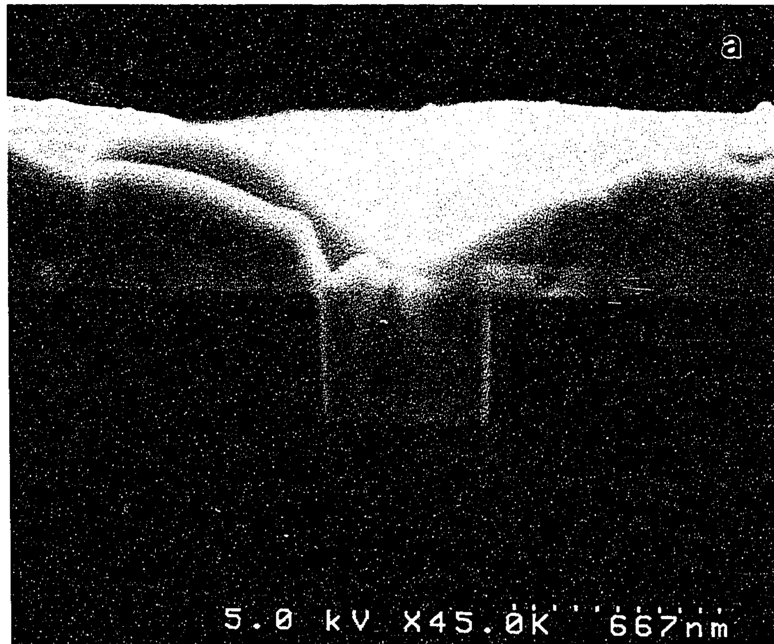
The wafer holder is equipped with a backside radiative heater to allow *in situ* annealing. The true wafer temperature must be carefully calibrated as it is in poor thermal contact with the substrate holder. The substrate holder is equipped with a J-type thermocouple bead that is located between the wafer position and the heater position, and a thermocouple attached to the heater shield. The heater can be controlled by a feedback loop from either of these thermocouples. The J-type thermocouple is the preferable controller since it has a relatively quick response time due

to its small size and, consequently, low thermal mass. Since it is not reasonable to assume that the temperature of the J-type thermocouple bead or the heater shield thermocouple is equivalent to the wafer temperature, these thermocouples were calibrated against a 6" Si wafer that had a thermocouple embedded in it[23]. The results of this temperature calibration are shown in Figure 2.1, which shows the temperature calibration for a wafer that has been coated with Cu and is in two specific geometries in the main chamber, namely, facing the closed shutter for the Cu target, and facing the window opposite the Cu target. The temperature listed in the text that follows indicate the true wafer temperature obtained from these calibrations.

2.2 Initial Attempts to Reflow Cu

Figure 2.2(a) and (b) show the results of annealing a 0.5 μm Cu/Mo film in 0.5 μm by 0.5 μm trenches for 30 minutes at 500°C and 300°C, respectively. The 300°C anneal results in incomplete trench filling, while the 500°C anneal results in complete trench filling, but not total planarization. Total planarization of the sample is preferable if it can be obtained without exceeding the limits of the thermal budget allotted to planarization of the metal. The lack of planarization can, however, cause complications during the chemical-mechanical polish (CMP) that is used in a Damascene technology[9]. However, the most difficult problem, that of filling the high aspect ratio trench, is solved at this point.

Figure 2.3 shows the as-deposited profile and the reflowed profile for a 0.5 μm Cu/Si_3N_4 in a 0.5 μm by 1.0 μm trench at 500°C for 5 minutes. This figure clearly demonstrates one of the most serious difficulties that a post-deposition reflow process must overcome, namely, thinning of the Cu film along the sidewall regions until the film agglomerates. Many other attempts were made to reflow thicker Cu films on Si_3N_4, as well as refractory metal barriers such as Mo, Ta and W. Because the results of these trials were similar, SEM micrographs have not been included; the Cu film on the sidewall regions thins until the film separates and reflow is no longer possible.



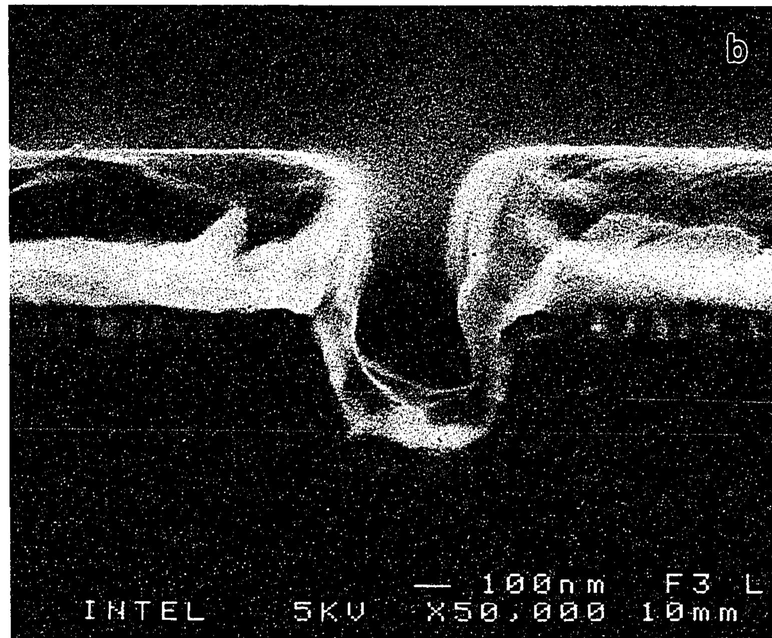
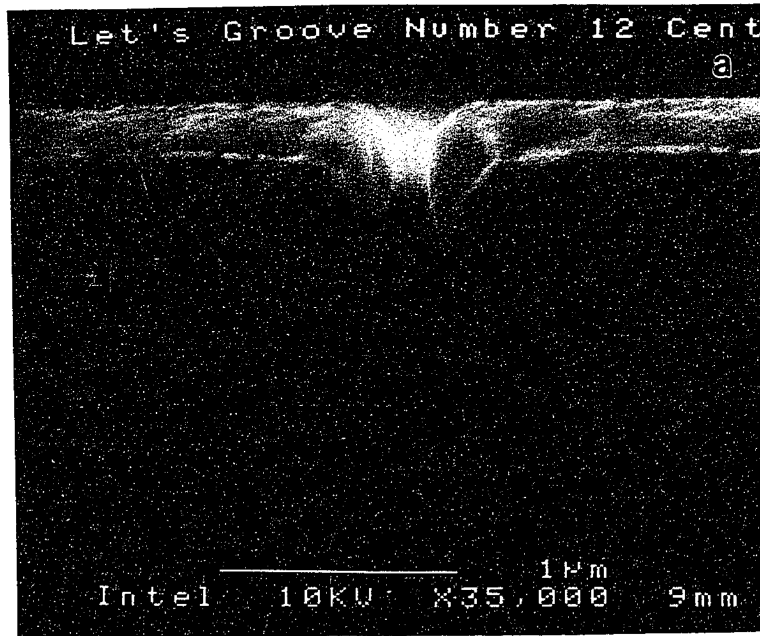


Figure 2.2: 0.5 μm Cu/Mo on a 0.5 μm by 0.5 μm trench that has been reflowed for 30 minutes at (a) 500°C and (b) 300°C.



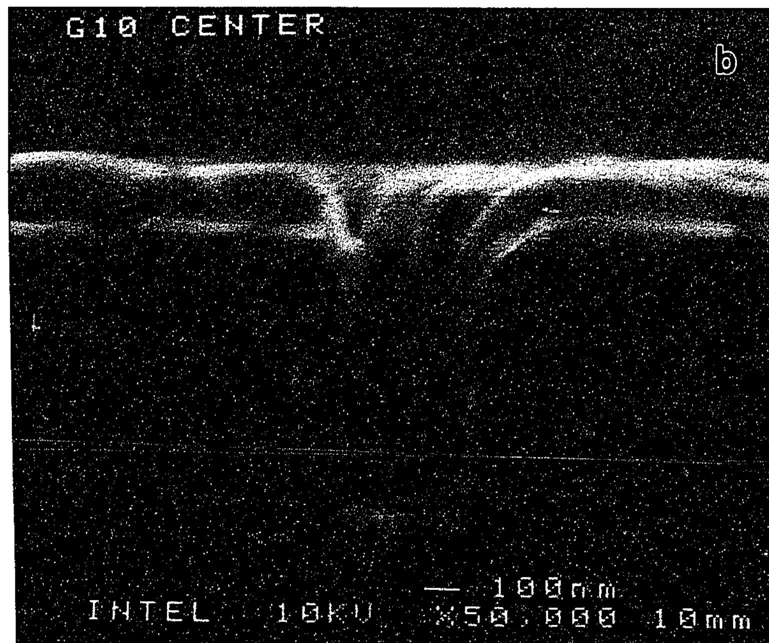


Figure 2.3: (a) As-deposited 0.5 μm Cu/Si_3N_4 on a 0.5 μm by 1.0 μm trench. (b) 0.5 μm Cu/Mo on a 0.5 μm by 1.0 μm trench, annealed for 5 minutes at 500°C.

2.3 Agglomeration of Thin Films

Agglomeration is the decomposition of a thin, continuous film into a collection of beads on a substrate at temperatures less than the melting temperature. It has been experimentally observed in a variety of thin metal films, including Au[24, 25], Ni[24], Sn[26], and Cu[27], and is caused by the development of grooves at grain boundaries/surface intersections due to surface diffusion. The agglomeration process has several stages: (i) a substrate intersecting perturbation must occur due to grain boundary grooving, dislocations, or other imperfections; (ii) once holes are formed in the film, the holes grow in radius by any one of a number of possible transport mechanisms (e.g. surface diffusion, volume diffusion, or evaporation-condensation); and (iii) the growing holes finally impinge upon one another, forming islands which continue to evolve toward their equilibrium state[28, 29, 30].

For an isolated grain boundary, heating the sample will cause the grain boundary groove to deepen as[31]

$$d \sim t^{\frac{1}{4}}, \tag{2.1}$$

where d is the groove depth and t is the time. In a polycrystalline sample, however, it is possible to maintain a finite groove depth for a range of film thicknesses and grain sizes. The driving force for agglomeration is the minimization of the free energy

$$E = A_s\gamma_s + A_{gb}\gamma_{gb} + A_i\gamma_i + A_{sub}\gamma_{sub}, \tag{2.2}$$

where A_s is the film-vapor surface area, A_{gb} is the grain boundary area, A_i is the film-substrate interfacial area, A_{sub} is the substrate-vapor area, and γ are the corresponding energies per unit area. For a continuous film, A_{sub} is zero and $A_i\gamma_i$ is constant, so the driving force for grooving is only due to the first two terms of Equation 2.2.

Assume that we have a model film composed of uniform grains of initial grain size D and thickness t. For a film that has not yet agglomerated, the configurational change as the film grooves can be described by the value of the groove angle, Ψ, as

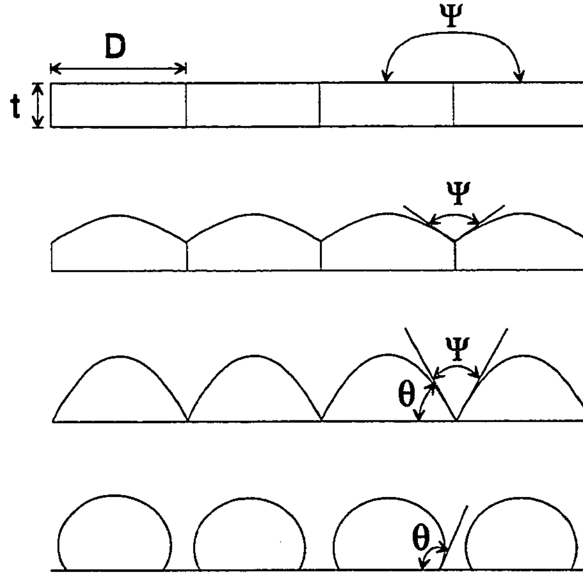


Figure 2.4: Agglomeration of a uniform 2-dimensional thin film with initial thickness t and grain size D.

shown in Figure 2.4. If we define Ψ_{crit} as the value of Ψ at which the grain boundary disappears, then the critical value of Ψ is related to D/t by the relationship[32]:

$$\frac{D}{t} = \frac{4(1 + \cos \Psi_{crit})}{\pi - \Psi_{crit} - \sin \Psi_{crit}}. \tag{2.3}$$

We also know that Ψ_e, the equilibrium value of Ψ, is related to the ratio of the grain boundary energy to surface energy by

$$\frac{\gamma_{gb}}{\gamma_s} = 2 \cos \frac{\Psi_e}{2} \tag{2.4}$$

so we must have $\Psi_e > \Psi_{crit}$ to avoid agglomeration. For Cu, a high angle grain boundary will typically have $\gamma_{gb}/\gamma_s = 0.32$[33], so that $D/t < 38$ to avoid agglomeration. For these ultra-pure Cu films, typical grain sizes are on the order of 1 μm, which requires a minimum film thickness of 300 Å over the entire microstructure to avoid agglomeration.

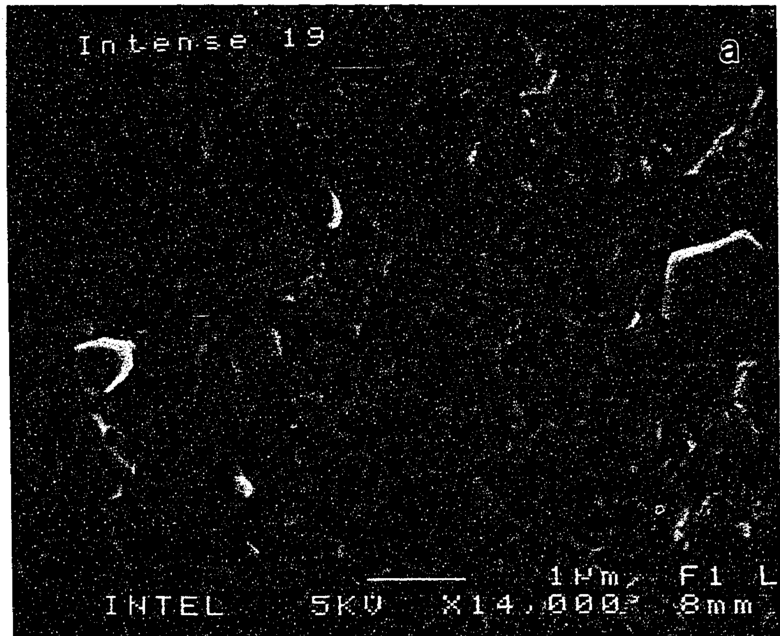
Cu thickness	underlayer	agglomerate?
500Å	*in situ* Mo	no
500Å	*in situ* Ta	no
500Å	*in situ* W	no
500Å	*ex situ* W	yes
1000Å	*ex situ* W	no
500Å	Si_3N_4	yes

Table 2.1: The results of annealing thin Cu films on *in situ* and *ex situ* refractory metal underlayers to test for agglomeration.

We experimentally examined the agglomeration of thin Cu films on refractory metal barrier layers. Unfortunately, sputtered films might not be initially continuous for very thin films, so the thinnest film that we deposited was 500 Å. The wafers were prebaked at 500°C for 30 minutes and allowed to cool before depositing a 750 Å refractory metal underlayer. Then for the *in situ* and *ex situ* case, a Cu film was deposited without or with a vacuum break, respectively, then annealed at 500°C for 30 minutes at base pressures $< 5 \times 10^{-10}$ Torr. The results, summarized in Table 2.1, indicate that we can avoid agglomeration of the Cu film for all *in situ* refractory metal underlayers if the minimum Cu film thickness is $\geq$ 500 Å. Figure 2.5 shows a 500 Å Cu film deposited on *in situ* and *ex situ* W. The as-deposited Cu film shown in Figure 2.3(a) in a 0.5 μm by 1.0 μm trench does not have the minimum Cu thickness to avoid agglomeration along the sidewalls, so we need to make improvements in the initial Cu coverage.

2.4 Improving Initial Trench Coverage

Reflowing Cu in high aspect ratio trenches, then, seems to require (i) an *in situ* deposited barrier layer and (ii) a minimum of 500 Å Cu coverage over the entire structure. Requirement (i) is easily satisfied by the UHV sputtering system described in Section 2.1, but requirement (ii) has not been satisfied for trenches with aspect ratios much greater than 1:1. Improved sputtering techniques are required to improve the initial coverage before successful reflow is possible; better initial coverage will also



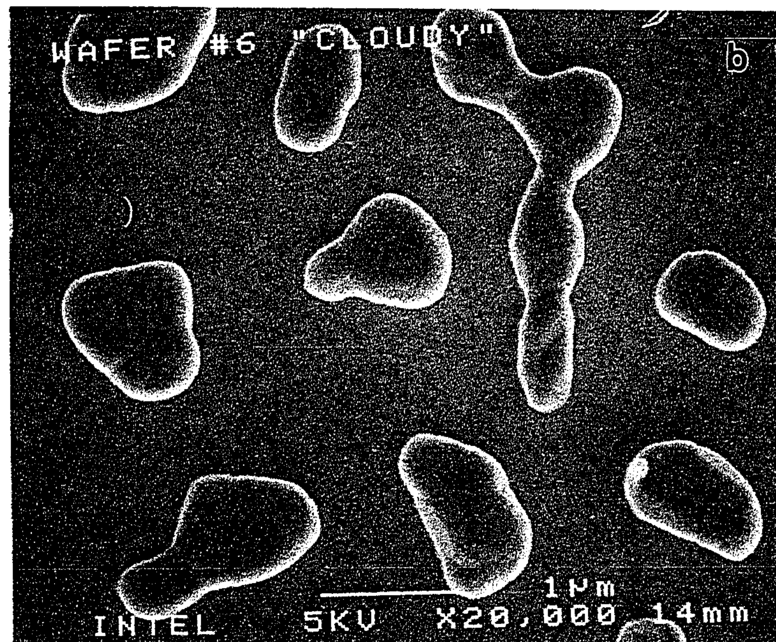


Figure 2.5: 500 Å Cu deposited on an (a) *in situ* and (b) *ex situ* W barrier layer and annealed at 500°C for 30 minutes.

decrease the amount of reflow that is necessary.

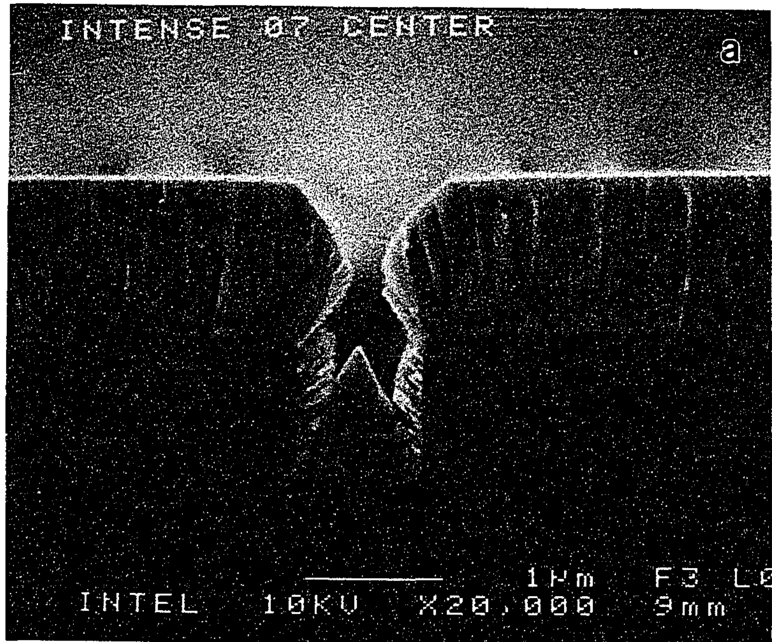
2.4.1 Target Geometry

The first obvious modification to make to improve the initial Cu coverage is to use a planar rather than a conical target. To first order, sputtering results in the emission of atoms from the near surface region of the target with an emission profile which is dependent on the cosine of the emission angle from the surface normal. Either target emits material along its normal, but the planar target's normal is parallel to the substrate normal, while the conical target's normal is tilted at approximately 45° with respect to the substrate normal. Figure 2.6 shows the results of sputtering a 1.0 μm thick Ta film from a planar target in trenches with aspect ratio 1:1 and 2:1, and clearly demonstrates that the sidewall coverage from the Ta planar target in a trench of aspect ratio 2:1 is much better than that for the Cu conical target shown in Figure 2.3(a). One would not expect the results from a planar Cu target to be identical to the result from the Ta target, since the Ta film shows clear columnar growth, indicating minimal diffusion during the deposition, which is not the case for a Cu film. It is reasonable to conclude, however, that the initial Cu coverage will be greatly improved by switching to a Cu planar target; there is clearly more than 500 Å Ta on the walls of the trench for aspect ratio as high as 2:1.

2.4.2 Substrate Bias

Another potential method to improve to the poor sidewall coverage in high aspect ratio trenches is to take advantage of the partial ionization of the sputtered Cu atoms. We attempted to improve the initial coverage by electrically isolating the substrate holder and applying a negative bias voltage to the substrate while depositing the Cu film. Biasing the substrate in this way will accelerate the positively charged Cu ions towards the substrate, and hopefully result in improved directionality of the deposited Cu.

Figure 2.7 shows one result of biasing the substrate to -60 V, which was the



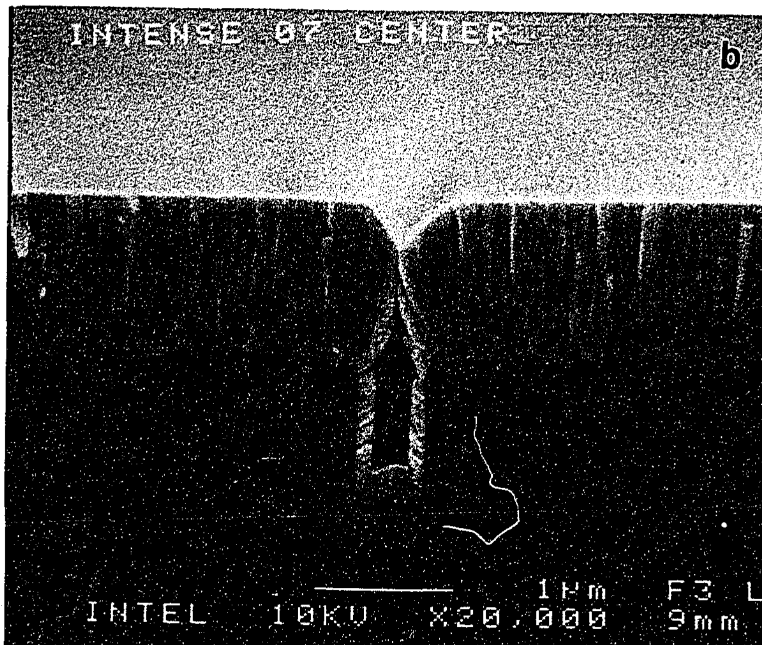


Figure 2.6: 1.0 μm Ta film deposited from the planar target in (a) a 1.0 μm by 1.0 μm trench and (b) a 0.5 μm by 1.0 μm trench.

maximum voltage our power supply would allow. To ensure that the substrate was in good electrical contact with the substrate holder, we coated the backside of the wafer with Cu to improve the electrical contact with the substrate holder. During the deposition, the measured substrate current was 149 mA. Assuming singly ionized Cu ions and using the experimentally determined deposition rate of 1.2 μm/minute, we calculated the ion to atom fraction to be only 0.03. It is surprising to see such a large improvement in the initial coverage considering this very low ion fraction; however, substrate bias may change the plasma physics considerably. Unfortunately, these results were not reproducible in subsequent attempts to bias the substrate, so we concluded that although there was good enhancement to some portion of the wafer, there was extremely poor uniformity across the 6" wafer.

Figure 2.8 shows the result of -60 V biased Cu/Ta for four different trench widths that were reflowed for 60 minutes at 500°C. In (a), the trench is 0.86 μm wide and the film on the sidewalls is continuous, in (b) it is 0.59 μm wide and the film has agglomerated along the sidewalls, in (c) it is 0.48 μm wide and the film has agglomerated along the sidewalls, and finally in (d) it is 0.38 μm wide and the film has bridged across the top. This figure illustrates an additional complication with reflowing metal in high aspect ratio trenches, namely, one must decide on the critical dimensions of interest (i.e. width) to create a successful reflow process because, for example, a reflow process that works for a 0.48 μm wide line will bridge for a 0.38 μm wide line as shown in Figure 2.8(c) and (d).

2.4.3 Additional Sputtering Parameters and Techniques

There are many additional sputtering parameters that can affect the initial Cu coverage, including Ar pressure during sputtering, target to substrate distance, and target geometry, and these factors are interdependent. The Ar pressure during sputtering can significantly affect the initial coverage. If the mean free path (mfp) of the ejected Cu atoms is less than the target to substrate distance, their trajectories will be randomized before reaching the substrate. Typical Ar pressures during sputtering from

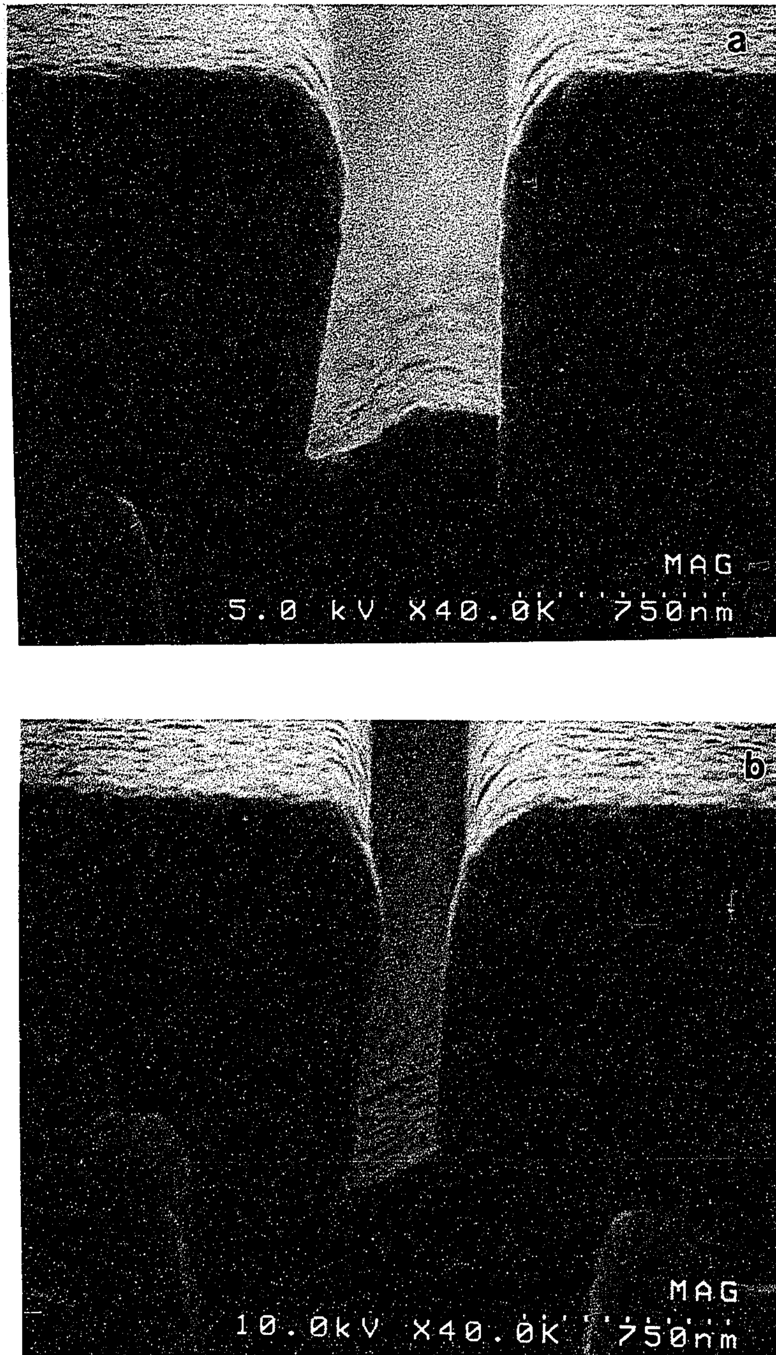
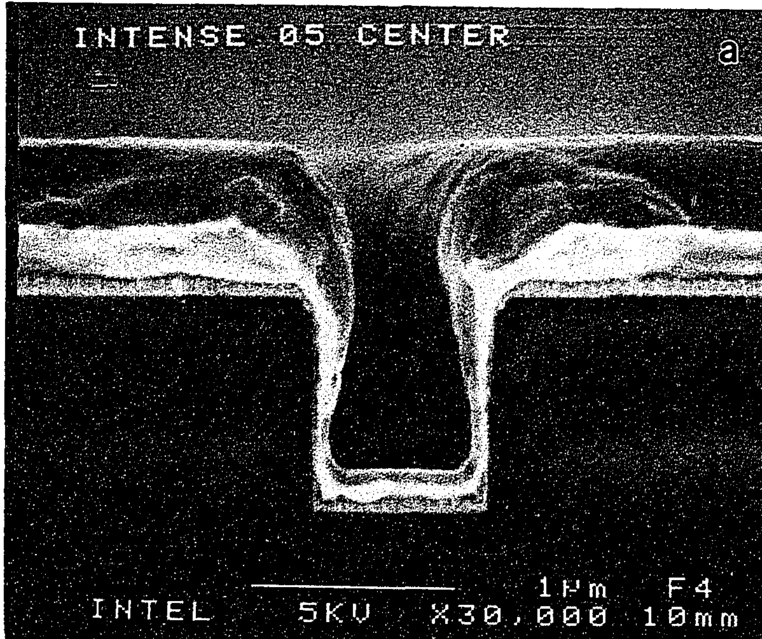
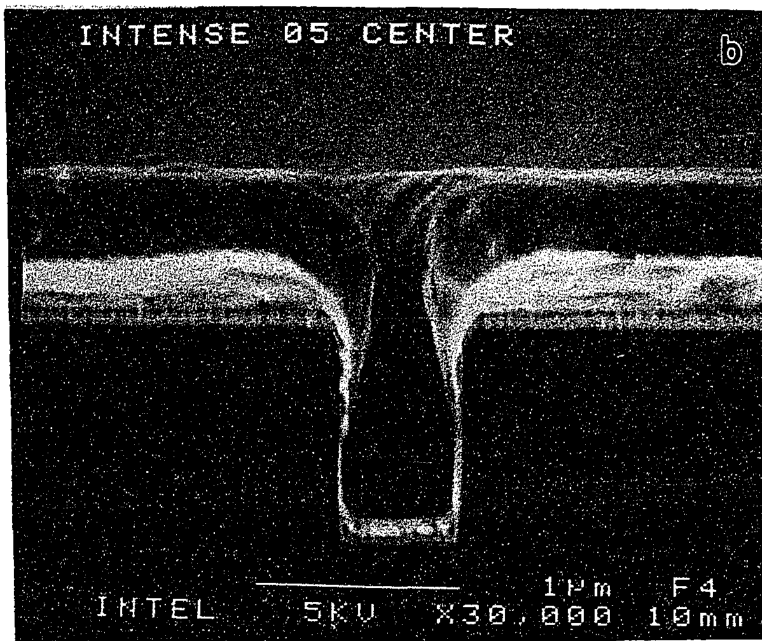
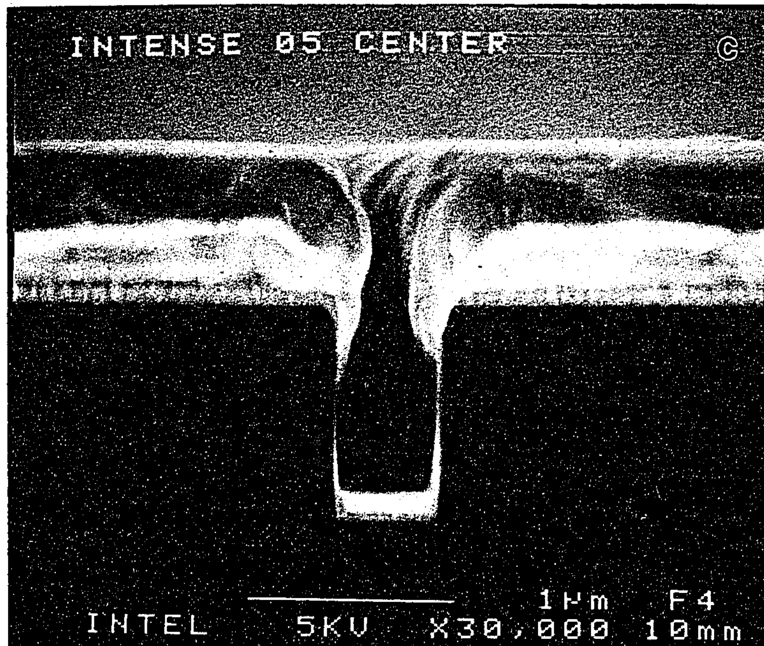


Figure 2.7: Cu deposited on SiO_2 with a -60 V substrate bias on a (a) 1.0 μm by 1.0 μm trench and (b) 0.5 μm by 1.0 μm trench. Result was not reproducible.


INTENSE 05 CENTER
a
1µm F4
INTEL 5KV X30,000 10mm


INTENSE 05 CENTER
b
1µm F4
INTEL 5KV X30,000 10mm



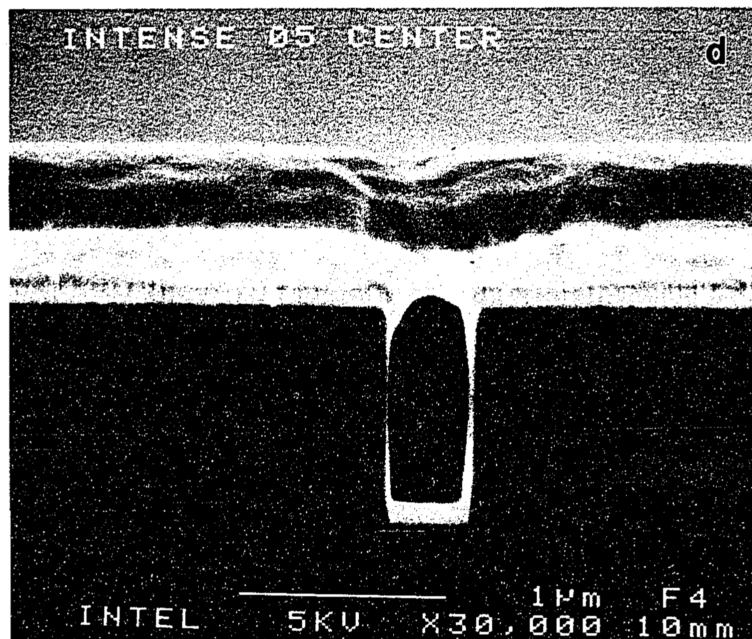


Figure 2.8: -60 V substrate biased Cu/Ta on trenches with widths of (a) 0.86 μm, (b) 0.59 μm, (c) 0.48 μm and (d) 0.38 μm that was reflowed for 60 minutes at 500°C.

the Cu conical target is 3 mTorr, and the mfp at this pressure is approximately 6 cm. However, the target to substrate distance for this target is approximately 10 cm. Unfortunately, it is impossible to maintain the plasma in front of the conical Cu target using lower Ar pressures. Alternatively, it could be advantageous to increase the target to substrate distance so that only target material that is emitted almost parallel to the substrate normal actually reaches the substrate, which would effectively collimate the emitted target material; however, this approach has several undesirable consequences, including lowering the deposition rate and increasing the size (and cost!) of the sputtering tool.

Many groups are working to develop novel sputtering techniques that improve filling capabilities in high aspect ratio features. These techniques include bias sputtering with high ionization fraction[34, 35], selective sputter/bias deposition[37, 38], energetic cluster impact[39], and collimation[36]. Several of these techniques yield good coverage, but not complete trench filling, and seem to be promising sputtering techniques to be used in combination with a post-deposition reflow process.

Rossnagel and Hopwood[34, 35] have enhanced the ionization fraction of the sputtered atoms by inductively coupling a radiofrequency (rf) plasma in the region between the sputtering cathode and the sample plane. Metal atoms sputtered from the cathode due to inert gas ion bombardment traverse the rf plasma and can be ionized. The metal ions can then be accelerated to the sample by means of a low voltage dc bias, such that the metal ions approach the substrate surface at normal incidence and at a specified energy. Rossnagel and Hopwood have been able to increase the ion fraction from an Al(0.5%)Cu target to around 0.85, by using a planar magnetron, rather than the conical magnetron used for our biasing experiments, and by having a much higher ionization fraction as well as a dc substrate bias from 0 to -300V.

Several studies also indicate that it is possible to selectively bias sputter deposit metals[37, 38]. The group in Reference [37] and [38] has experimentally demonstrated that (i) Al is selectively deposited on Si but not on W surfaces, and (ii) Ti is selectively deposited on Si but not on Pt surfaces during bias sputter deposition of Al and Ti with simultaneous Ar ion bombardment. In these cases, atomic deposition is

occurring simultaneously with ion bombardment, and there are clearly ballistic effects that depend upon the atomic weights of the deposited metal, the underlayer, and the sputtering gas. These experiments are currently understood only on a phenomenological level but selective bias sputtering might be a promising technique for improved trench filling.

There has also been work done by Haberland, *et al.*, to fill contact holes by energetic cluster impact[39]. In this case, the atoms are sputtered using a typical magnetron sputtering source, collected into clusters in a cooled aggregation tube, and then directed towards the substrate by biasing the substrate. Haberland, *et al.* have had some success in filling vias and trenches, and the results are extremely dependent upon the bias voltage and the substrate temperature.

2.5 Summary

A post-deposition reflow process clearly has several requirements to meet before success is possible. For a process that uses refractory metal barrier layers, the barrier and the Cu film must be deposited without a vacuum break, and the initial Cu film must be not only continuous, but at least 500 Å thick over the entire structure so that the film does not agglomerate. The film thickness is especially important along the sidewalls of high aspect ratio trenches. To achieve this minimum film thickness, the best technique that is readily available in the magnetron sputtering system is to change to a planar target. Ta films deposited on high aspect ratio trenches from a planar target show considerable improvement in initial coverage and conformality. Also, there are many groups working on novel and promising techniques to improve initial trench filling in high aspect ratio trenches; these include bias sputtering with high ionization fraction, selective sputter/bias deposition, and energetic cluster impact. Even if these techniques do not result in complete trench filling, perhaps they will provide better initial coverage for Cu reflow.

Chapter 3 Cu Reflow on Refractory Metal Barrier Layers

3.1 Planarization Mechanisms

An important part of developing a Cu reflow process is an understanding of the dominant transport mechanism for planarization. If the reflow occurs in a spatially homogeneous manner, the dominant transport mechanism and the appropriate constants can be determined experimentally using optical diffraction techniques. For a surface with a periodic structure, we can define a surface structure factor[40]:

$$S(\vec{k}) \equiv 4\pi^2 \Sigma [a_n(t)]^2 \delta(k_1 - 2\pi n/d)\delta(k_2) \tag{3.1}$$

where $\vec{k} = (k_1, k_2)$ is the component of the scattered wave vector parallel to the surface, $a_n(t)$ is the amplitude of the nth Fourier component of the surface, and d is the period of the grating. First-order scattering theory yields a scattering probability that is proportional to $S(\vec{k})$[40, 41], so normally incidence light will be scattered in the directions corresponding to $\sin\theta = \pm n\lambda/d$. The intensity of the nth peak, $I_n(t)$, is proportional to $[a_n(t)]^2$.

For a profile that has been written in terms of its Fourier series

$$y(x,t) = \sum_{n=0}^{\infty} [a_n(t)\cos\omega_n x + b_n(t)\sin\omega_n x] \tag{3.2}$$

with $\omega_n = 2\pi n/d = n\omega$ and $a_n(0)\omega_n \ll 1$, we know that[42]

$$a_n(t) = a_n(0)e^{-(F\omega_n + A\omega_n^2 + D\omega_n^3 + B\omega_n^4)t} \tag{3.3}$$

and similarly for $b_n(t)$, where F, A, D, and B are given by

$$\begin{aligned}
F &= \frac{\gamma_s}{2\mu} \text{ (viscous flow)} && (3.4)\\
A &= \frac{p_0\gamma_s\Omega^2}{(2\pi m)^{\frac{1}{2}}(kT)^{\frac{3}{2}}} \text{ (evaporation} - \text{condensation, mfp} \ll \frac{\pi}{\omega}) && (3.5)\\
D &= A' + C, && (3.6)\\
A' &= \frac{\rho_0 D_G\gamma_s\Omega^2}{kT} \text{ (evaporation} - \text{condensation, mfp} \gg \frac{\pi}{\omega}) && (3.7)\\
C &= \frac{D_v\gamma_s\Omega}{kT} \text{ (volume diffusion)} && (3.8)\\
B &= \frac{D_s\gamma_s\Omega^2\nu}{kT} \text{ (surface diffusion).} && (3.9)
\end{aligned}$$

Here, γ_s is the surface energy per unit area, μ is the coefficient of viscosity, p_0 is the equilibrium vapor pressure, Ω is the atomic volume, m is the mass of a molecule of vapor from the solid, k is Boltzmann's constant, T is the temperature of the film, ρ_0 is the equilibrium vapor density, D_G is the coefficient of diffusion of the vapor molecules in the inert atmosphere, D_v is the coefficient of volume diffusion, and D_s is the coefficient of surface diffusion. The first term in Equation 3.3 is due to viscous flow, the second term is due to evaporation-condensation in the limit in which the mean free path (mfp) of the evaporated atoms is long compared to the distance over which appreciable transport occurs (mfp $\ll \pi/\omega$), the third term is due to evaporation-condensation in the limit in which the mfp of the evaporated atoms is short compared to the distance over which appreciable transport occurs (mfp $\gg \pi/\omega$) and/or volume diffusion, and the last term is due to surface diffusion.

For a surface that is flattening, then, it is possible to determine the dominant transport mechanism by comparing the decaying intensities of different diffracted orders. The diffracted intensities decay as

$$\begin{aligned}
\ln[\frac{I_n(t)}{I_n(0)}] &= 2\ln[\frac{a_n(t)}{a_n(0)}] && (3.10)\\
&= -2(F\omega_n + A\omega_n^2 + D\omega_n^3 + B\omega_n^4)t && (3.11)\\
&= -2(Fn\omega + An^2\omega^2 + Dn^3\omega^3 + Bn^4\omega^4)t. && (3.12)
\end{aligned}$$

Wafer Number	Description	Time Reflowed at 500°C	Results
06	Cu(25°C)/Ta	2 hours	1% filled
07	Cu(25°C)/Ta	14 hours	20% filled
08	Cu(25°C)/Ta	-	-
09	Cu(150°C)/Ta	-	-
11	Cu(150°C)/Ta	12 hours	80% filled
13	Cu(25°C)/W	-	-
14	Cu(25°C)/W	17 hours	some everywhere
16	Cu(150°C)/W	11 hours	< 1% filled
17	Cu(150°C)/W	-	-

Table 3.1: Reflow results for 0.75 μm Cu films deposited with the substrate at room temperature or 150°C, on Ta and W barrier layers. In this table, "20% filling" means that 20% of the trench lengths had reflowed completely or almost completely, and 80% had reflowed negligibly. The base pressure for the depositions on Ta was $< 5 \times 10^{-10}$ Torr, while the base pressure for depositions on W was $\leq 5 \times 10^{-9}$ Torr.

If surface diffusion is the dominant planarization mechanism, then the slope of $\ln[\frac{I_n(t)}{I_n(0)}]$ versus t for the $n = 2$ peak will be 16 times the slope from the $n = 1$ peak, whereas if volume diffusion is the dominant planarization mechanism, the slope of the $n = 2$ peak will only be 8 times larger than the $n = 1$ peak. Once the dominant mechanism is confirmed, it is straightforward to calculate the appropriate material constant (B in the case of surface diffusion) using the same plot.

3.2 Reflow of Polycrystalline Cu Films

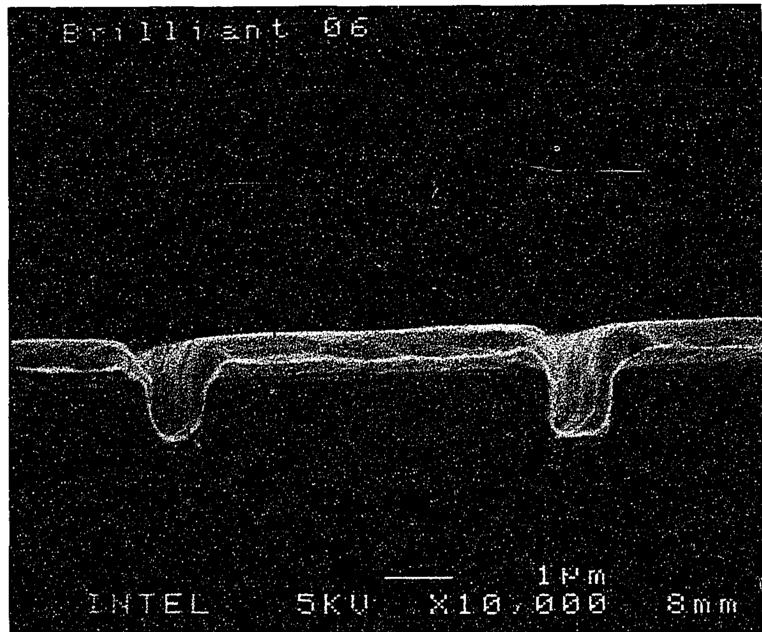
Cu films 0.75 μm thick were deposited on either 750 Å Ta or W barrier layers on a 1 μm wide by 1 μm deep array of SiO_2 trenches capped with Si_3N_4 spaced 6 μm apart. The Cu and Ta or W films were deposited *in situ* with the magnetron sputtering system described in Section 2.1, with the substrate temperature during the Cu deposition at room temperature or at 150°C. The samples were then annealed *in situ* at 500°C. The results are summarized in Table 3.1.

During the post-deposition anneal, the samples were monitored by the decay of the diffracted peaks intensities in optical diffraction from the periodic array of trenches, in an attempt to determine the planarization mechanism and the appropriate constants for Cu. The as-deposited profile in a 1 μm by 1 μm trench from our magnetron

sputtering system (see Figure 4.1) does not satisfy the small slope requirements where Equation 3.3 is valid, so for short reflow times, the intensity will not decay as shown in Equation 3.12; we chose to reflow Cu into deep trenches so that we would be able to gain information about the departure from the small slope regime and hopefully extract some information about reflow into higher aspect ratio trenches. Unfortunately, this created problems which will be demonstrated shortly, and made it impossible to verify the dominant mechanism or measure any material constants. The diffracted peak intensities decayed continuously during the anneal, indicating that the surface was planarizing during the entire annealing time for all the samples, but did not decay in a manner consistent with spatially uniform surface diffusion.

SEM micrographs taken after the samples were annealed are shown in Figures 3.1 through 3.5, and show that the surface reflowed in an extremely non-uniform manner. Approximately 80% of the trenches are filled on the sample that was deposited at 150°C on Ta, approximately 20% of the trenches are filled on the sample deposited at room temperature on Ta, $< 1\%$ of the trenches are filled on the sample deposited at 150°C on W, and there was some filling everywhere on the sample deposited at room temperature on W. The diffracted peaks intensities decayed during the entire annealing time, yet Figures 3.1 to 3.5 indicate that only localized sections of the films reflowed; from this, we conclude that local sections of each trench were filling on time scales much less than the total annealing time, which makes it impossible to use the technique developed in the previous section to determine the planarization mechanism.

The SEM micrographs reveal several interesting details. Wafer 06 was annealed for only 2 hours; the SEM micrographs indicated that approximately 1% of the trench lengths had reflowed and the filled sections are approximately 1 μm long. Also, there is some filling in all areas of wafer 06 as compared to the as-deposited profile. Additionally, the sample surface appears to be relatively smooth and the grain boundary grooves are apparent; the profiles show that material seems to be reflowing into the trenches in "drapes" (i.e. extremely local reflow of material that does not completely fill the trench) of material.



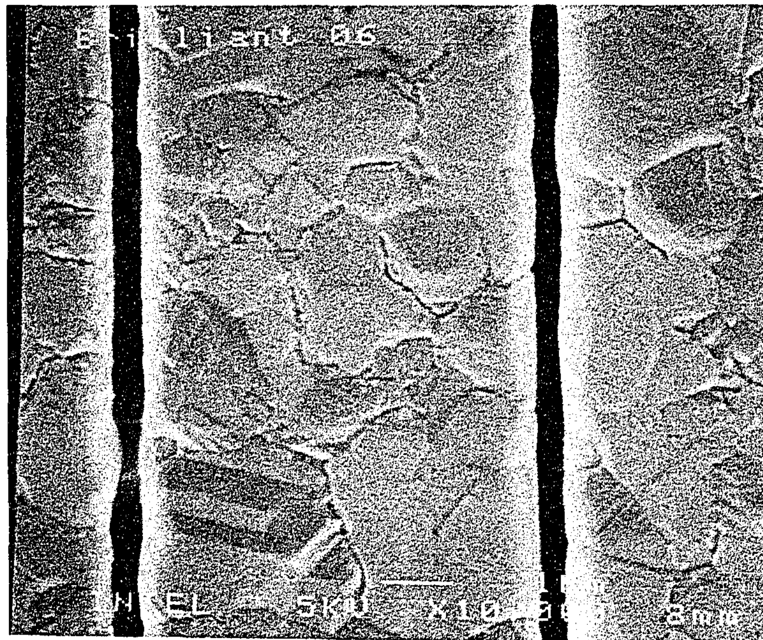
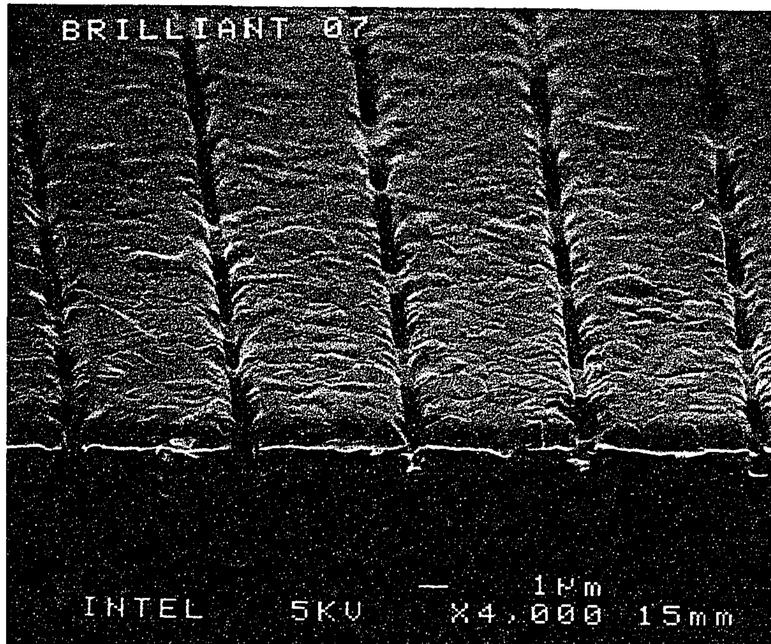


Figure 3.1: Wafer 06 (Cu(25°C)/Ta, reflowed 2 hours at 500°C). Approximately 1% of the trenches are filled.



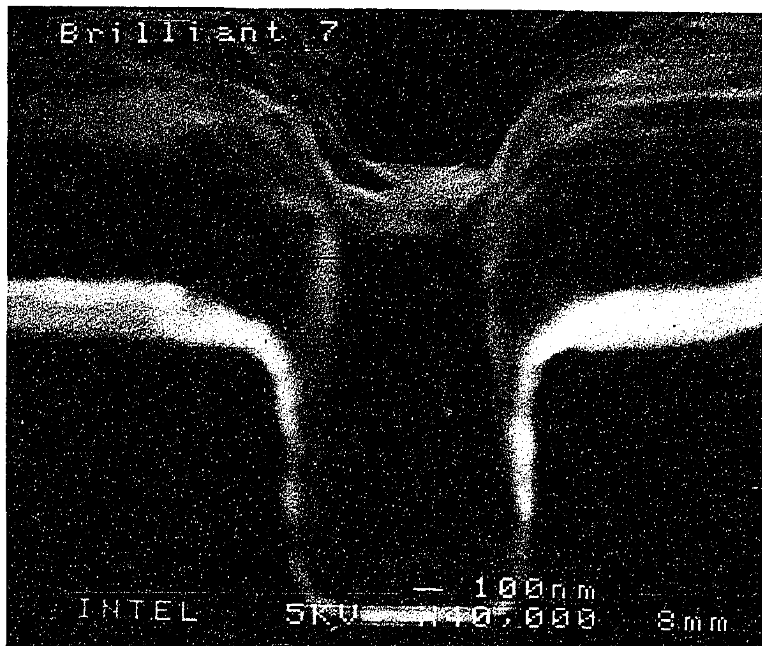
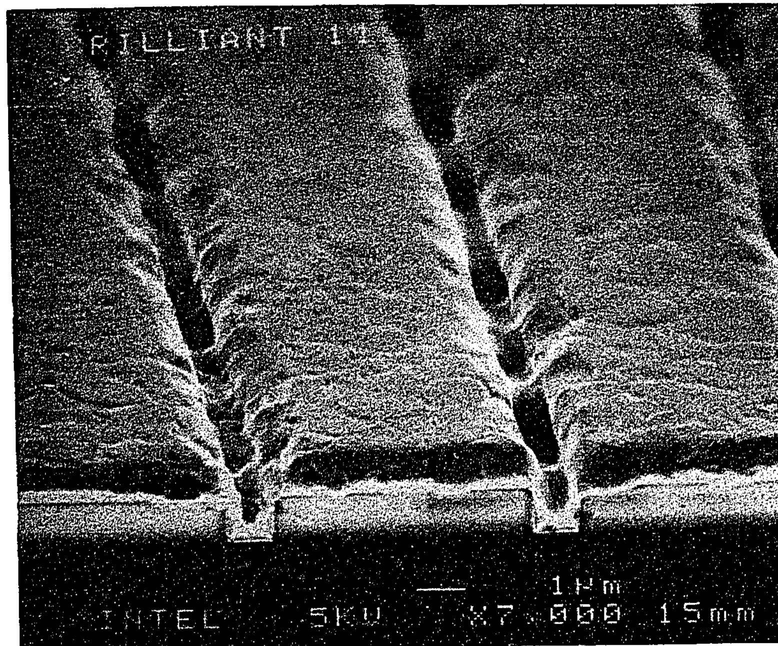


Figure 3.2: Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). Approximately 20% of the trenches are filled.



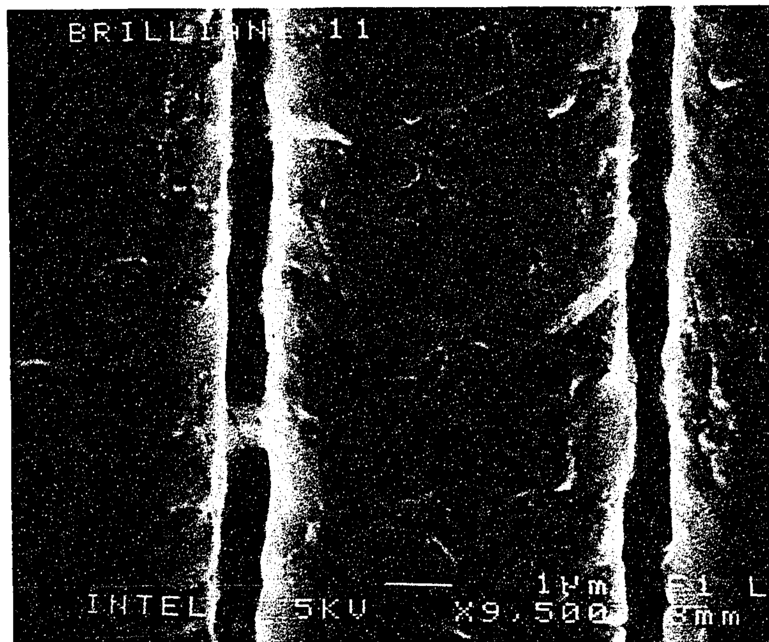
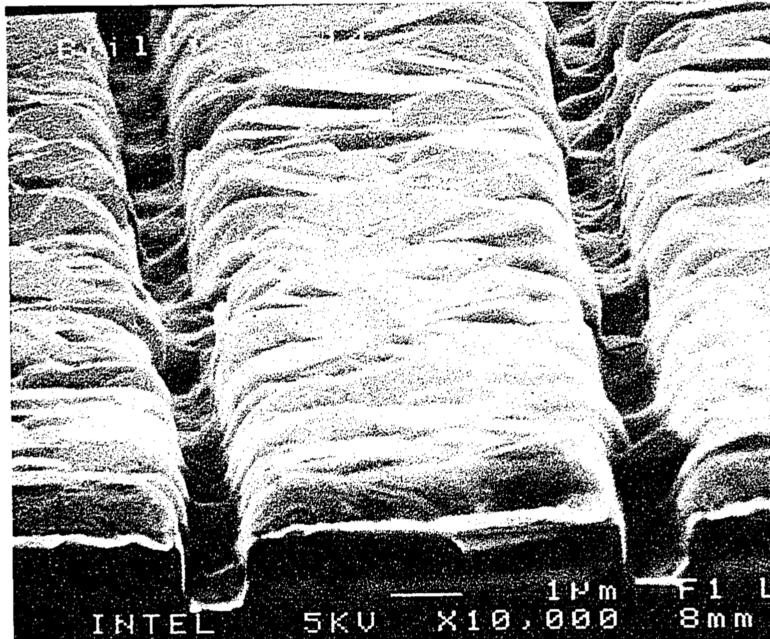


Figure 3.3: Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). Approximately 80% of the trenches are filled.



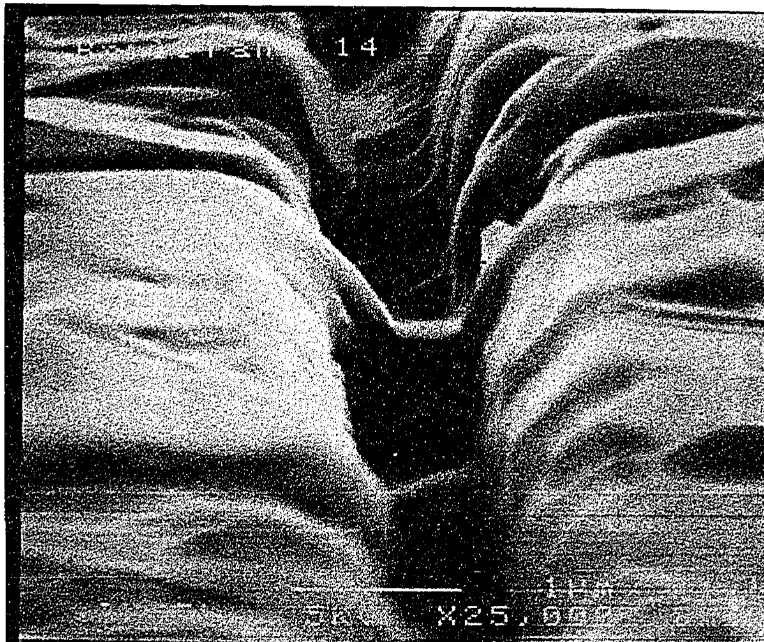
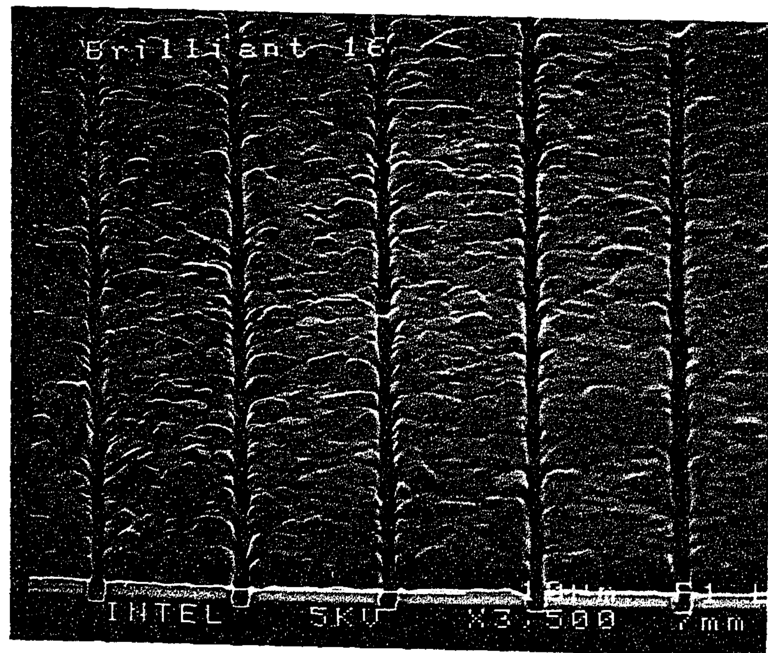


Figure 3.4: Wafer 14 (Cu(25°C)/W, reflowed at 500°C for 17 hours). There is some trench filling everywhere.



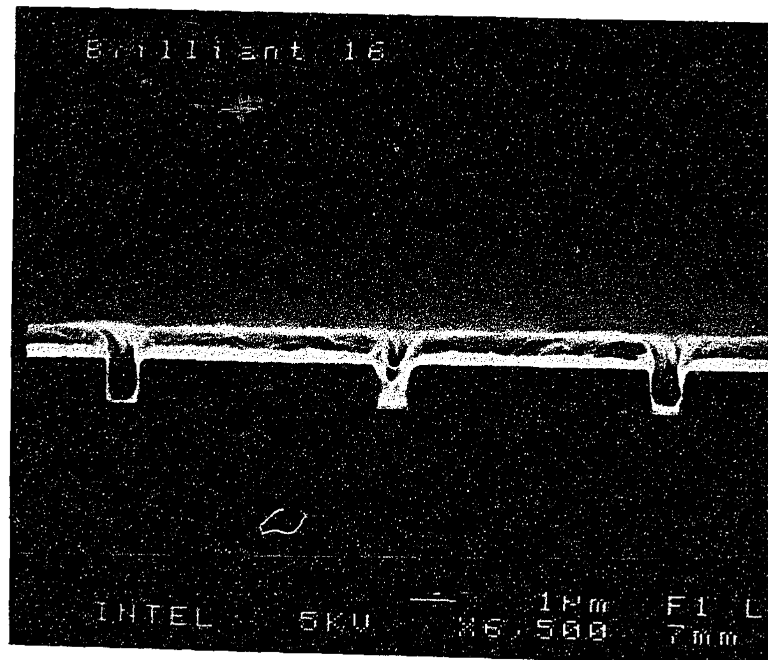


Figure 3.5: Wafer 16 (Cu(150°C)/W, reflowed at 500°C for 11 hours). Less than 1% of the trenches are filled.

Wafer 07 also has a relatively smooth surface with some degree of hillocking; grain boundary grooves are apparent on its surface as well. The film surface is approximately 20% filled and these filled sections are 1 - 3 μm long. The filled regions are well defined and drop to the nearby unfilled section within approximately 0.1 μm. The shoulders near the filled sections also appear to have "avalanched" (i.e. there is a very distinct region on the trench shoulder from which material has been removed to fill the trench) into the trench.

Wafer 11 has a smoother surface than wafer 07 and shows very little hillocking. Approximately 80% of the trenches are filled. Grain boundary grooves are apparent on this surface as well. The filled sections are approximately 3 - 6 μm long, but the filled regions are not as well defined as that for wafer 07. Also, the shoulders near the trench have "avalanched" into the filled region.

Wafer 14 has a much rougher surface than any of the previous films. The surface of each grain appears to be flat, but each grain's surface is tilted with respect to the sample normal. The trenches are filled in a much more uniform fashion and there is some filling everywhere in the trenches. The completely filled regions, however, are typically very short, on the order of 0.1 μm in length.

Wafer 16 has less than 1% of the trench lengths filled. The sample surface is smoother than the surface of wafer 14, but rougher than the other samples. The filled regions are very short–typically on the order of 0.1 μm.

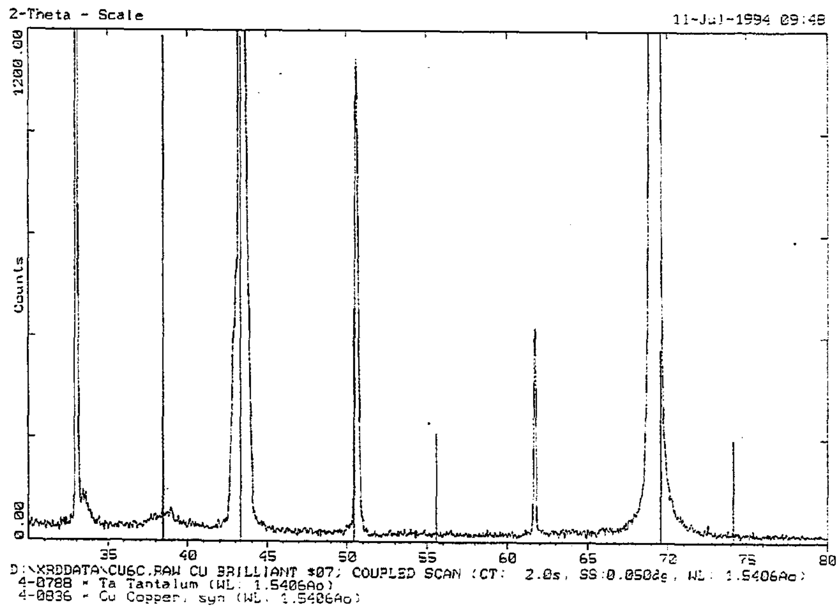
Unfortunately, SEM micrographs can give us only limited information concerning these perplexing variations in Cu reflow which are apparently due to the differences in underlayers and deposition conditions. Accordingly, we performed x-ray diffraction and transmission electron microscope (TEM) analysis to better understand the film morphology and texture, and how this affects reflow.

3.2.1 Texture Scans

The Cu films deposited on Ta listed in Table 3.1 were examined using a x-ray diffractometer (XRD) in a $\theta - 2\theta$ geometry and in a rocking curve geometry, and the results

are shown in Figure 3.6 - 3.9. The samples deposited on Ta at 25°C exhibited a strong (111) texture, and also a much weaker (200) texture. The ratio of intensities for the (200) to the (111) peak for the as-deposited sample (Wafer 08) was I_{200}/I_{111} = 0.006, and the ratio for the annealed sample (Wafer 07) was $I_{200}/I_{111} = 0.04$. For a randomly oriented powder sample $I_{200}/I_{111} = 0.46$. The samples deposited on Ta at 150°C exhibited a strong (111) texture and had no (200) texture present. These results are summarized in Table 3.2. The (111) peaks of each of these films was examined in a rocking curve geometry to measure the directionality of the (111) normal with respect to the substrate normal. The Cu films deposited at 25°C have a full-width at half-maximum (FWHM) of 6.4° and 6.0° for wafer 07 and wafer 08, respectively, while the Cu films deposited at 150°C have FWHM of 2.8° for both wafer 09 and wafer 11; hence the films deposited at 150°C are more sharply (111) textured. Comparison between the (111) FWHM and the SEM photos in Figures 3.2 and 3.3 suggests that the FWHM is correlated with the surface roughness, indicating that the flat surfaces within grains are largely (111) textured, but that these grains are tilted with respect to the substrate normal. Note that wafer 09 was annealed at 250°C for 30 minutes following the deposition so that it could be compared with the results from References [43] - [45], which will be discussed below. An anneal at 250°C for only 30 minutes will cause negligible reflow of material. The drawback of XRD is that it gives us information that is averaged over the beam area, whereas reflow differences are happening on lengthscales on the order of 1 μm, and it is extremely difficult to differentiate between the flat portions of the film and the film along the sidewalls of the trenches. We will argue in Chapter 4 that the reflow mechanism is driven by surface diffusion, so that we are interested in the texture of the surfaces, which is not always parallel to the substrate.

Cu films grown on Ta or W barrier layers at room temperature and at 150°C have been characterized in terms of grain size, stress and texture in References [43] - [45]. In these references, Cu on both barrier layers exhibited a strong (111) as-deposited texture. Cu deposited at room temperature exhibited bimodal grain growth, with abnormal growth of the (100) orientation, for annealing temperatures $\geq$ 150°C. How-



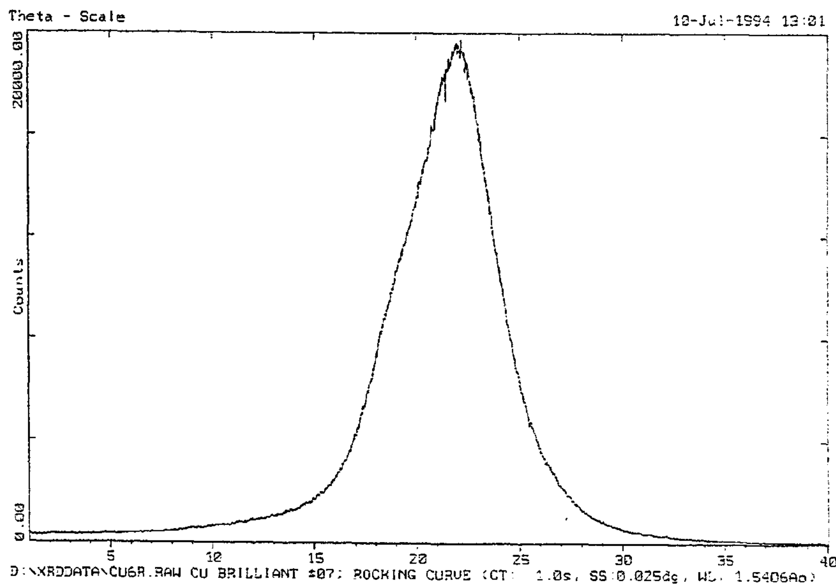
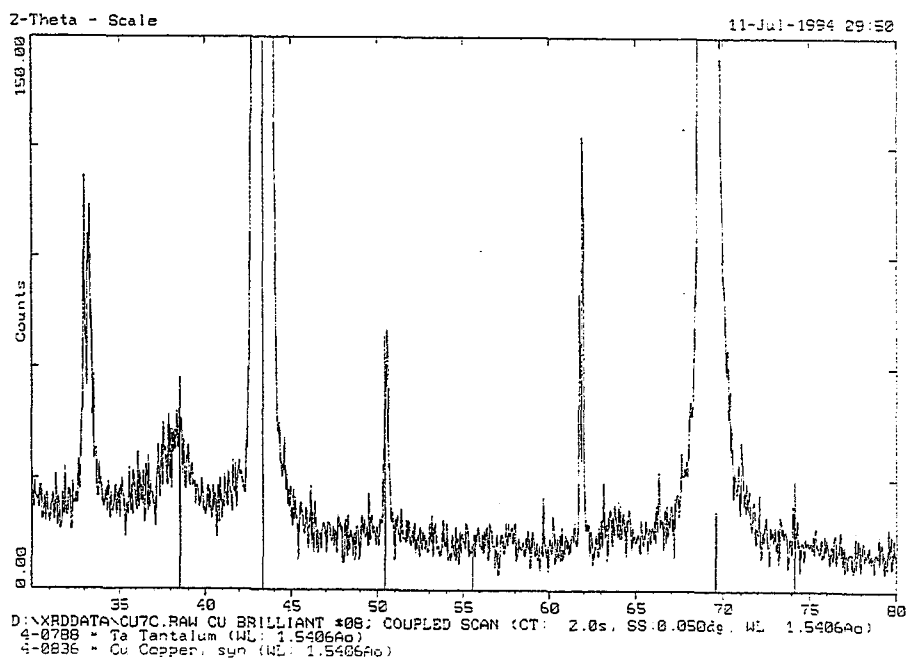


Figure 3.6: Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak.



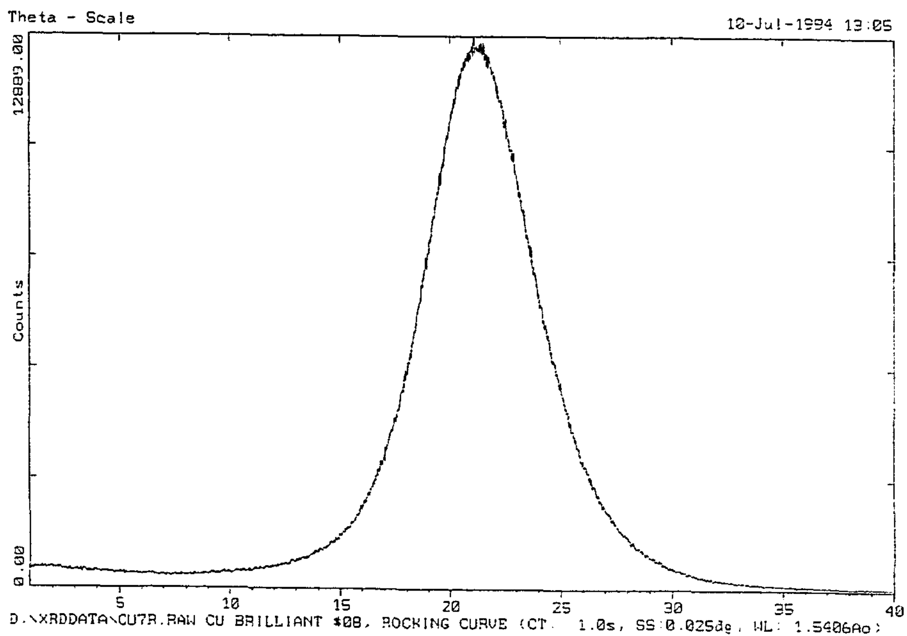
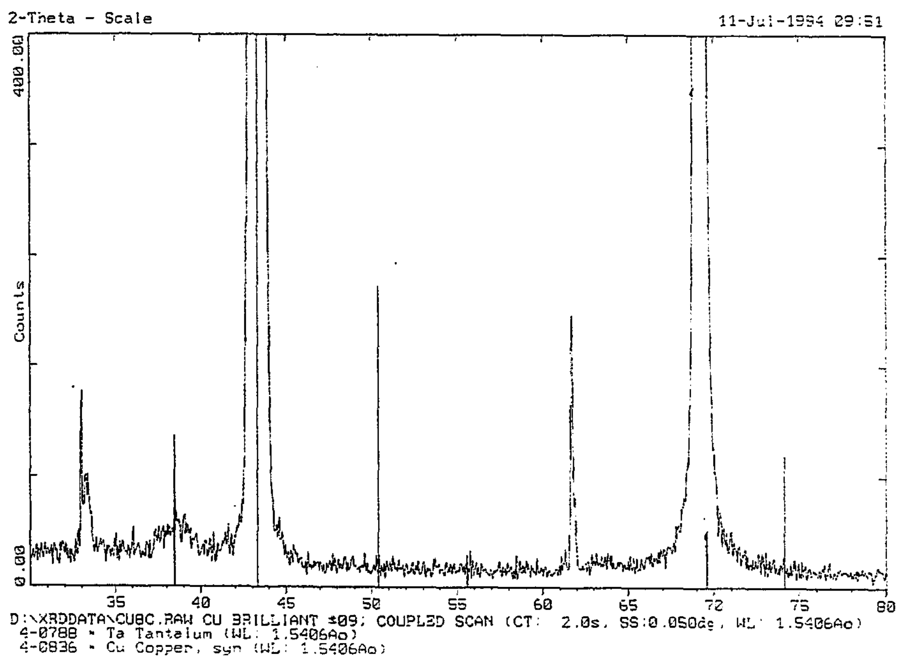


Figure 3.7: Wafer 08 (Cu(25°C)/Ta, as-deposited). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak.



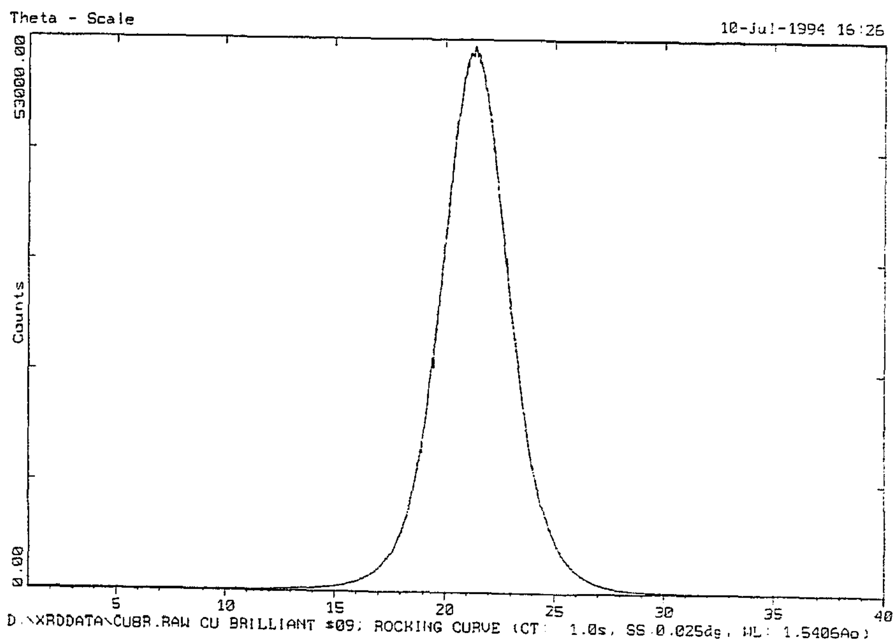
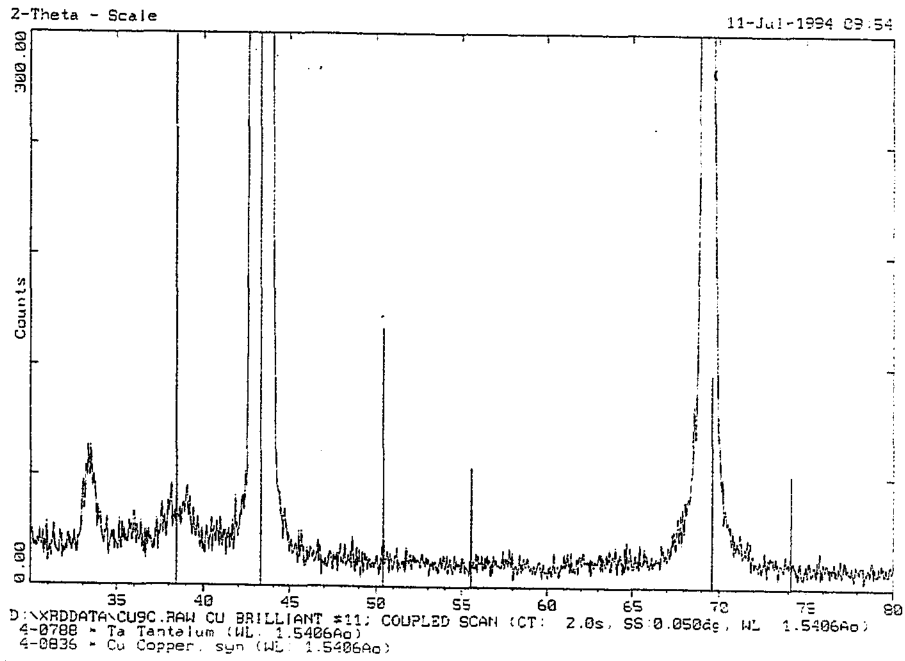


Figure 3.8: Wafer 09 (Cu(150°C)/Ta, annealed at 250°C for 30 minutes). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak.



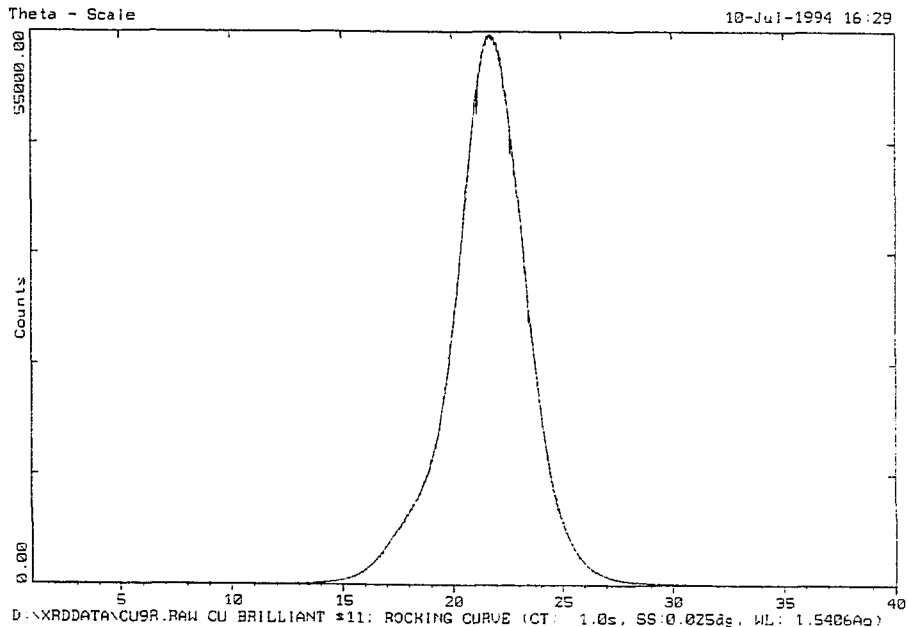


Figure 3.9: Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). (a) $\theta - 2\theta$ scan and (b) rocking curve for (111) peak.

Wafer Number	(111)?	(200)?	$\frac{I_{200}}{I_{111}}$	(111) FWHM
07	yes	yes	0.04	6.0°
08	yes	yes	0.006	6.4°
09	yes	no	–	2.8°
11	yes	no	–	2.8°

Table 3.2: Summary of $\theta - 2\theta$ x-ray diffraction and rocking curve results for Cu films deposited on Ta.

ever, abnormal grain growth could be suppressed by depositing Cu on Ta at 150°C and annealing at 250°C. Depositing Cu on W at 150°C resulted in the presence of both a (111) and (200) orientation for both the as-deposited sample and a sample annealed at 250°C. The grain size distribution of these films also varied, but in all cases, Cu films deposited on W exhibited more extensive grain growth. The authors of References [43] - [45] attributed this difference to the differences in texture of the various films. In particular, the (111) integrated peak intensity for Cu on W to that of Cu on Ta is 0.12, demonstrating that Cu deposited on W is more weakly textured in the as-deposited state. Because one driving force for grain growth increases sharply with decreasing average surface energy[46], and because the surface energy of Cu decreases sharply around the (111) orientation[47], the weaker the (111) texture, the larger the driving force for grain growth. Modeling the driving forces for grain growth is discussed in more detail in Section 4.5.

The limited $\theta - 2\theta$ x-ray diffraction scans performed on our reflowed and as-deposited Cu films on Ta are consistent with these results. It seems reasonable to assume, then, that while we might have slightly different grain size distribution than the results in References [43] - [45] due to differences in film purity, the results of their work is applicable to our films. It also seems clear that the differences in film texture, grain growth mechanisms, or stress is greatly affecting the reflow results on a given underlayer, although it is certainly not clear which of these is causing the large differences between the reflow results on these films.

3.2.2 TEM Cross-sections of Reflowed Cu Films

X-ray techniques are of limited usefulness because variations in the reflow occurred on lengthscales on the order of 1 μm, and it is difficult to determine the orientations on the trench sidewalls as opposed to the flat surfaces. Cross-sectional transmission electron microscope (TEM) samples were made to examine the microstructure in both the filled and non-filled regions. These samples were prepared using a wedge technique complemented by minimal ion milling; this technique is described in Reference [48]. The wedge technique is necessary for samples of this type since we wished to compare large numbers of trenches; this technique allowed us to image trenches over a region of $\approx$ 100 μm while standard dimpling techniques can only create a thin region on the order of 5 μm. TEM was performed on a Phillips EM430 electron microscope. The point resolution of the microscope is 2.3Å and it is equipped with an energy dispersive x-ray (EDX) detector. A double-tilt goniometer was used to align the substrate along the desired orientation.

Wafers 07, 08, 09, 14, and 16 were prepared in cross-section for TEM analysis as described above. TEM cross-sectional micrographs are shown in Figures 3.10 - 3.18. Figure 3.10 shows the as-deposited Cu(25°C)/Ta film (wafer 08) and a selected area diffraction (SAD) image taken with a 0.5 μm diameter aperture (the effective diameter of the sample region that is imaged is listed as the "aperture diameter" here). This image clearly shows all aspects of the trenches. 0.8 μm wide by 1.0 μm deep trenches are etched into the SiO_2 down to the Si and then capped with 350 Å Si_3N_4; the 750 Å Ta barrier layer is clearly visible above the Si_3N_4 cap and is too dense to image, and the 0.75 μm thick polycrystalline Cu film is on top. The crack present in (a) at the top edge of the trench is due to sample preparation; these cracks can be avoided after many long, character-building attempts are made at sample preparation. The grain size for this as-deposited Cu film is on the order of 500 - 1000 Å.

Figure 3.11 shows the Cu(25°C)/Ta film after it was annealed at 500°C for 14 hours (wafer 07). The filled sections of this sample, as shown in (a), were typically single grains or had a single grain boundary that extended from the bottom of the

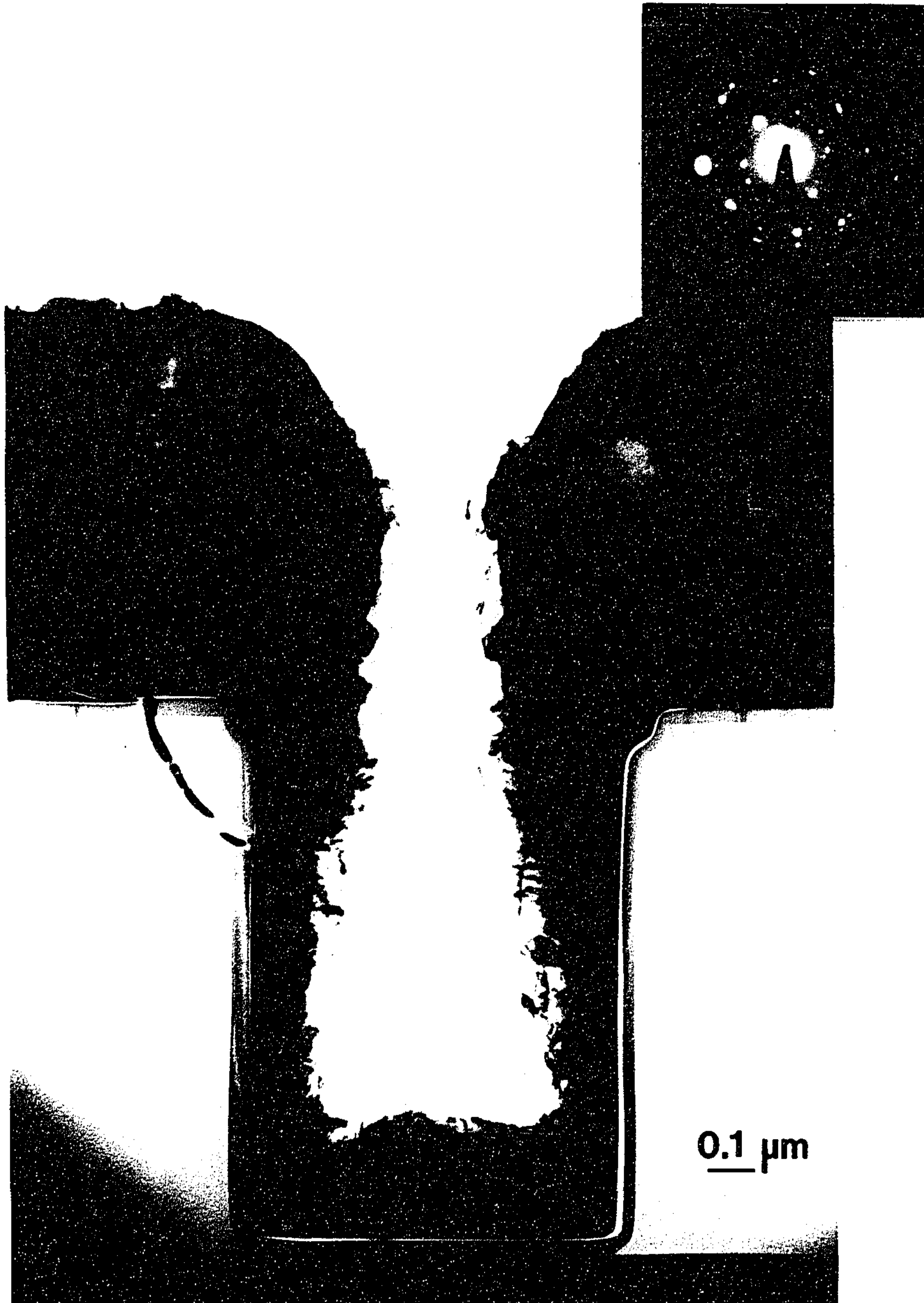


Figure 3.10: Wafer 08 (Cu(25°C)/Ta, as-deposited). (a) Bright field (BF) image of an as-deposited 0.8 μm by 1.0 μm trench and (b) SAD pattern using a 0.5 μm aperture. Typical grain size is on the order of 500 - 1000 Å.

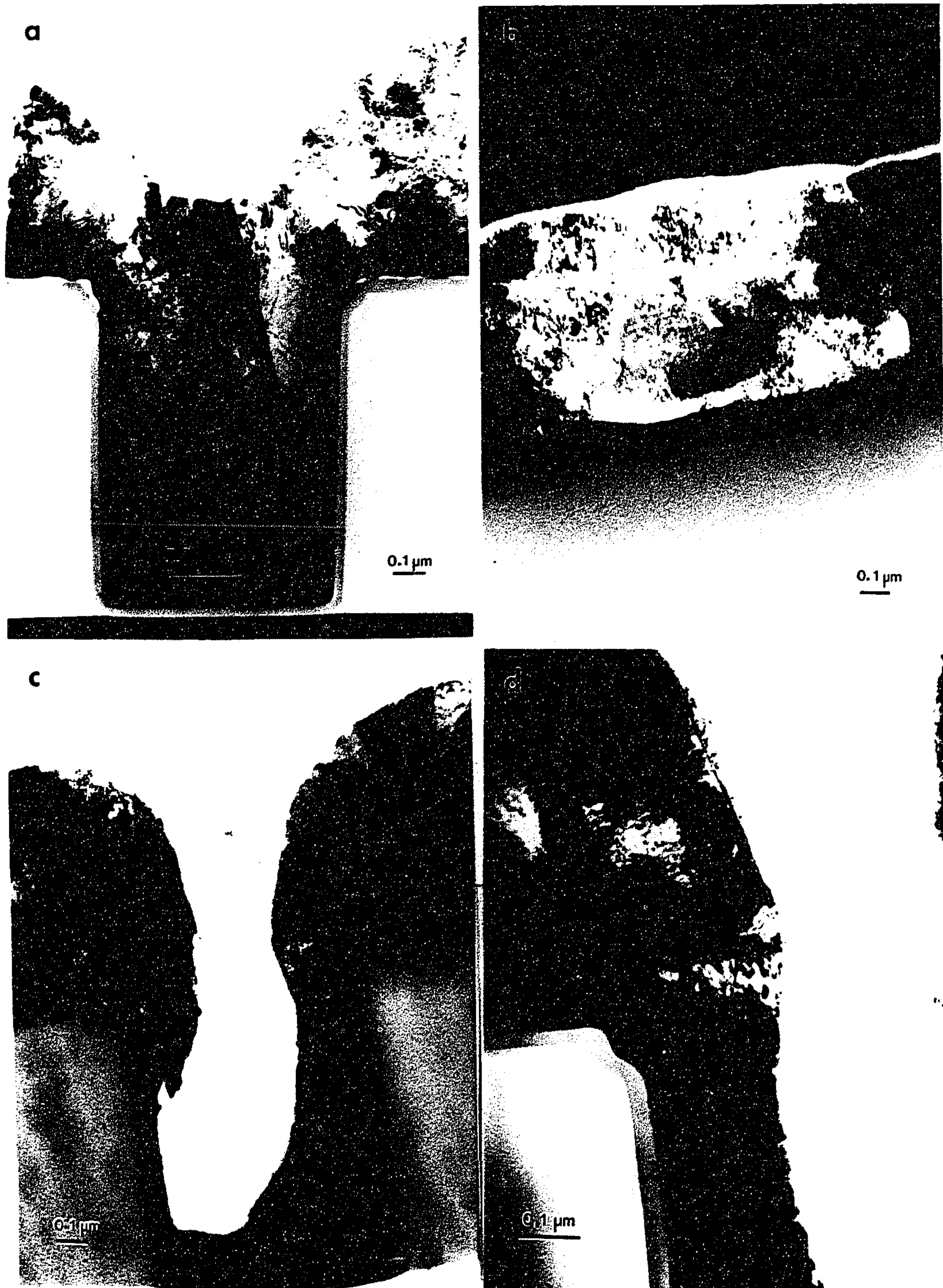


Figure 3.11: Wafer 07 (Cu(25°C)/Ta, reflowed at 500°C for 14 hours). (a) BF image of a filled 0.8 μm by 1.0 μm trench. (b) Dark field (DF) image of grain in flat portion between trenches. (c) BF image of an unfilled, agglomerated trench. (d) BF image of a groove – anti-groove pair.

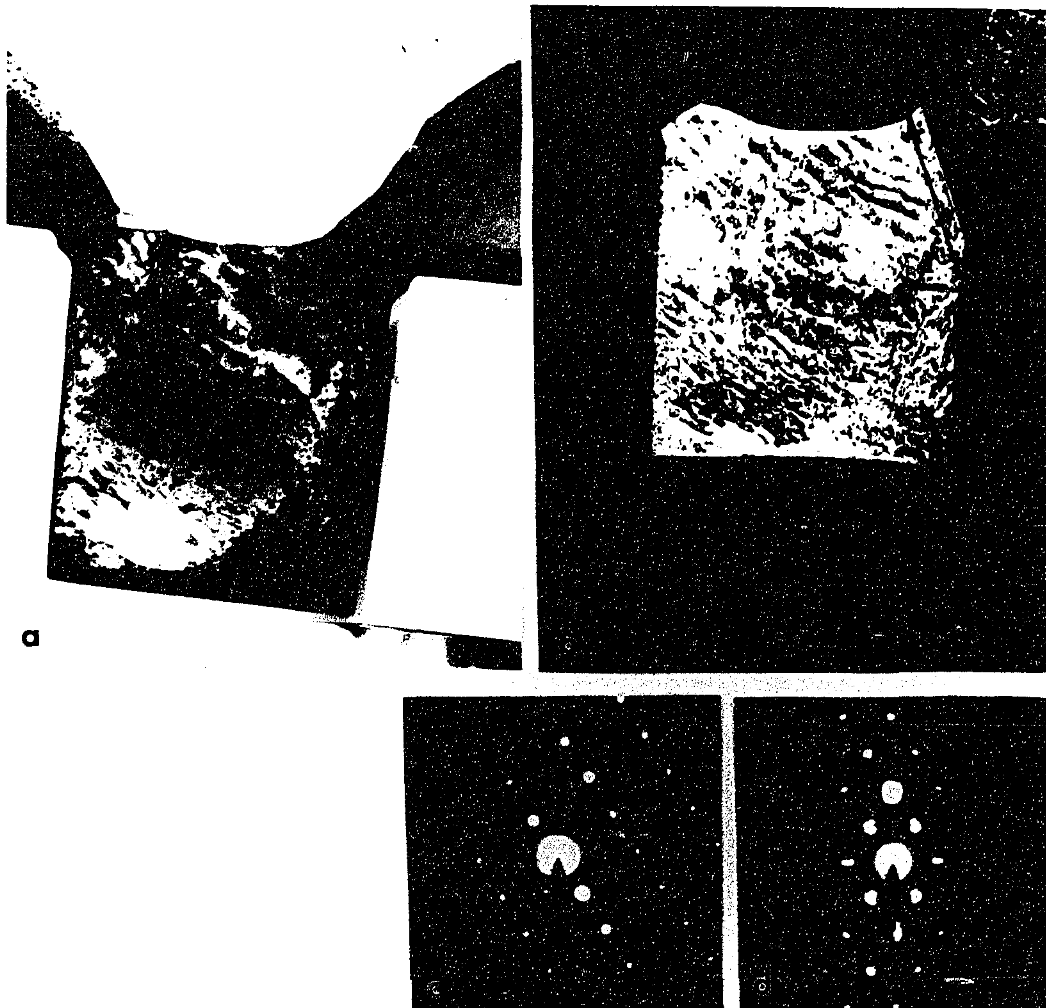


Figure 3.12: Wafer 11 (Cu(150°C)/Ta, reflowed at 500°C for 12 hours). (a) BF image of a filled 1 μm by 1 μm trench. (b) DF image of (a). (c) SAD from single grain in trench. (d) SAD from single grain above trench shoulder to the left of trench.

Figure 3.13: Wafer 11. (a) BF image of a filled 1 μm by 1 μm trench that had grains stacked from the bottom of the trench upwards. (b), (c) and (d) BF images of unfilled trenches. Note the faceting within single grains, and that the grain boundaries are often perpendicular to the surface.

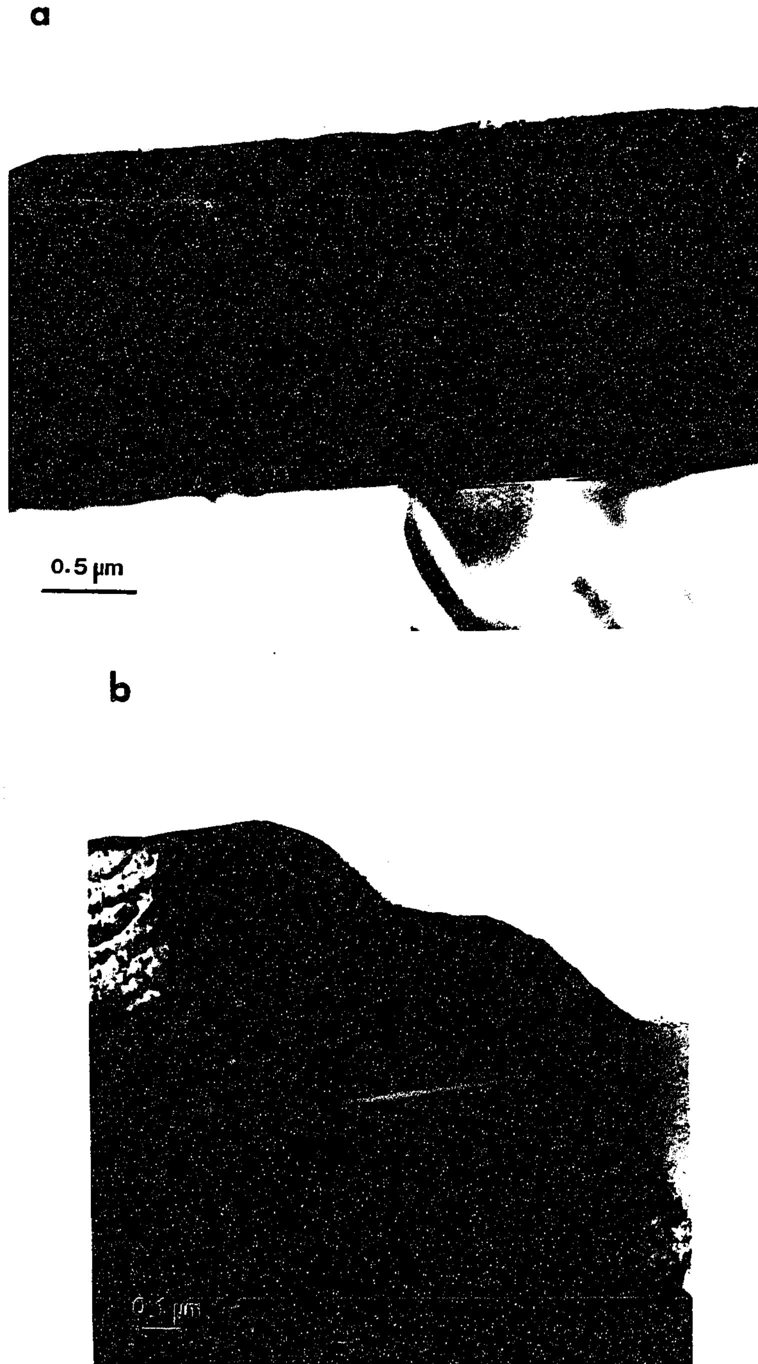


Figure 3.14: Wafer 11. (a) BF image of the flat region between trenches. (b) BF image of an "avalanched" region near a filled trench.

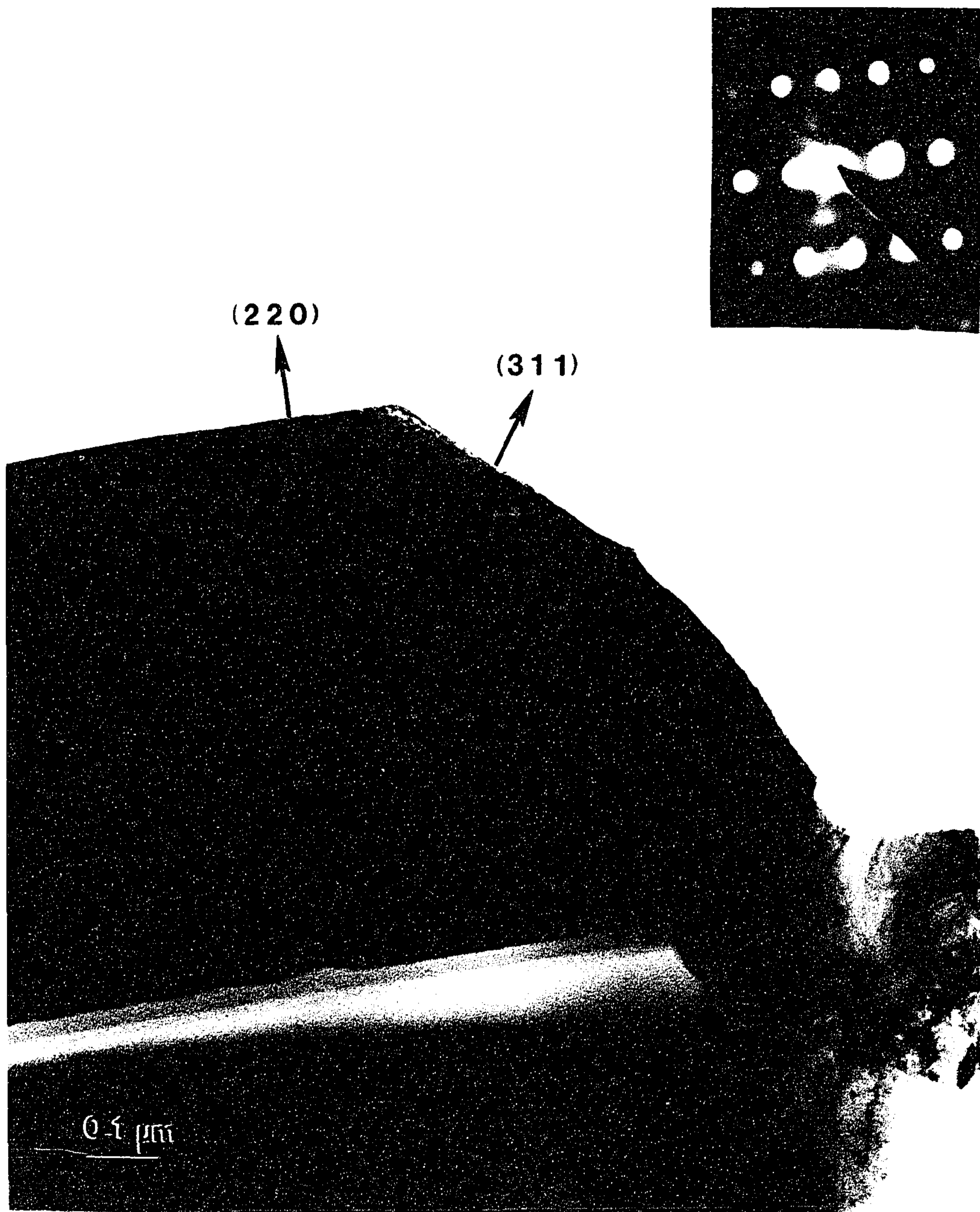


Figure 3.15: Wafer 11. BF image of a faceted grain and the SAD image from the grain indicating the orientation of the surfaces.

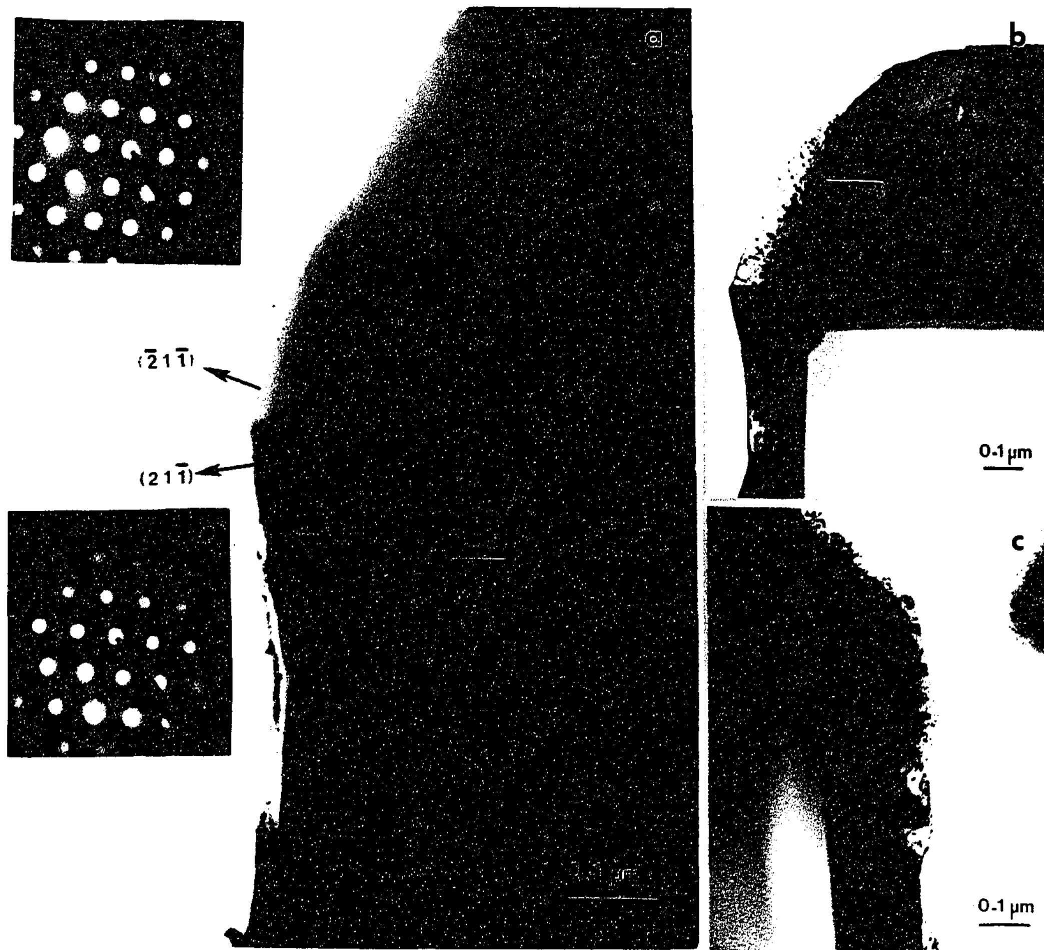


Figure 3.16: Wafer 11. (a) BF image of a groove – anti-groove pair and micro-diffraction images from the grains on each side of the anti-groove. (b) and (c) Groove – anti-groove pairs along trench sidewalls.

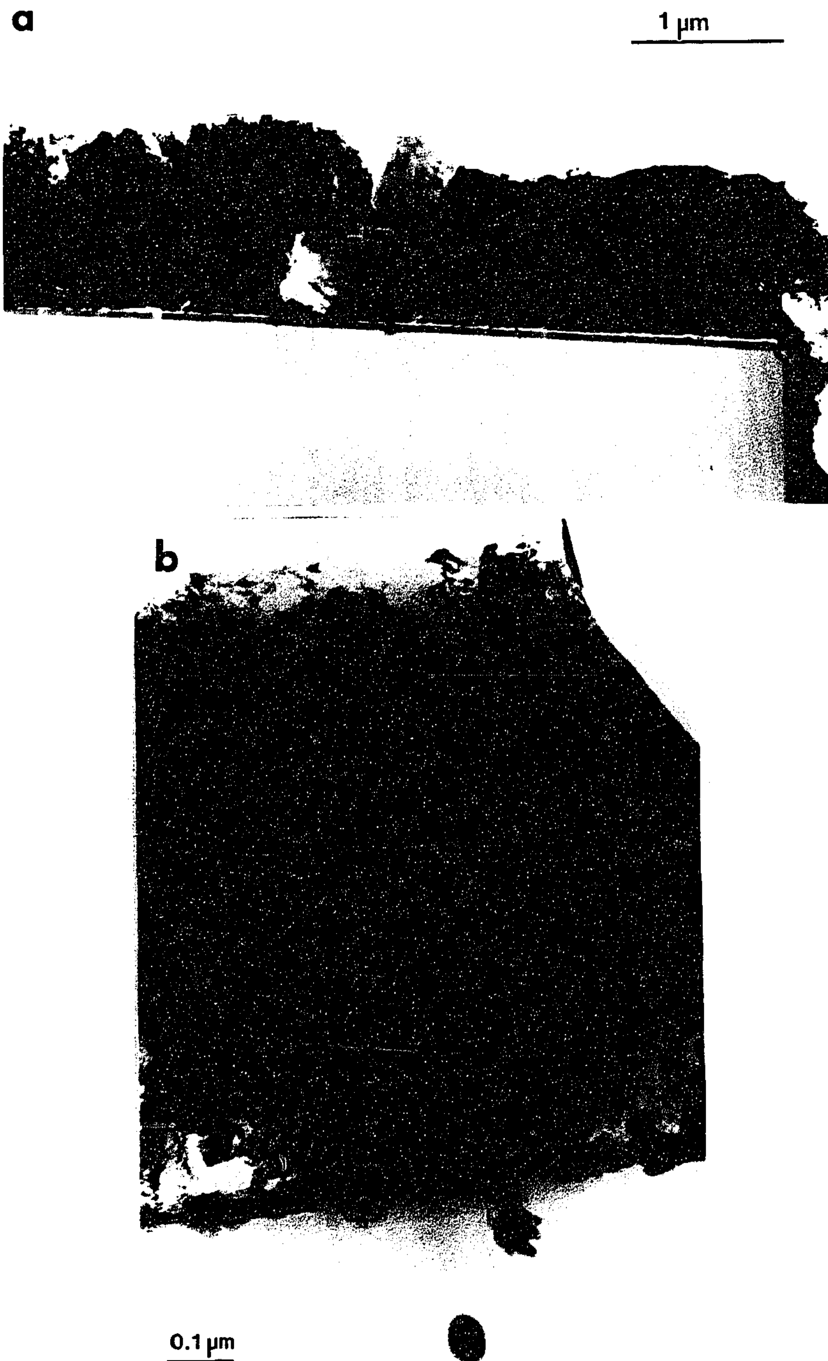


Figure 3.17: Wafer 14 (Cu(25°C)/W, reflowed at 500°C for 17 hours). (a) BF image of a Cu_xSi_{1-x} grain on the surface of the Cu film. (b) EDX beam marks in the SiO_2 shows Cu had diffused into the SiO_2. W diffusion barrier was insufficient for this long anneal. Cu has diffused through the W and the Si_3N_4 into the SiO_2, and Si has diffused into the Cu.

Figure 3.18: Wafer 16 (Cu(150°C)/W, reflowed at 500°C for 11 hours). (a) BF image of a 1 μm by 1 μm trench where the Cu has agglomerated on the sidewall. (b) BF image of the flat region between trenches. Most grains are columnar and the film surface is relatively rough. (c) and (d) BF images of trenches that have not agglomerated. Typical trenches were approximately half full.

trench to near the top center; the grain boundary was often a twin boundary. Typical grain sizes in this film were on the order of 0.75 - 1.25 μm on the flat regions of the sample between grain boundaries as shown in (b). The Cu film had agglomerated along one sidewall for a large number of the trenches as shown in (c); in the region that could be imaged, the Cu film had agglomerated on one sidewall in 16 out of 40 trenches, however, none agglomerated on both sidewalls simultaneously. Also, there were several groove – anti-groove pairs as shown in (c) and (d).

Figures 3.12 - 3.16 shows TEM images of the Cu(150°C)/Ta film that was reflowed at 500°C for 12 hours (wafer 11). Typical filled and unfilled section are shown in Figure 3.12 and 3.13. The filled sections typically had either (i) a grain boundary that extended from the bottom of the trench to the top center, (ii) a single grain within the trench, or (iii) several grains separated by grain boundaries parallel to the bottom of the trench. The unfilled sections show some filling, relative to the as-deposited profile, and typically have grain boundaries perpendicular to the local film surface along the sidewalls. The grains along the sidewalls in the unfilled regions are columnar, but much smaller in size that those on the flat surfaces between trenches. Both filled and unfilled trenches show the Cu surface is faceting, and there are often faceted surfaces within single grains and sharp discontinuities in the slope at the grain boundaries. Figure 3.14 shows the flat surfaces between grains. The films are columnar and the grain size is approximately 0.5 - 2 μm. Near the edges of filled trenches, the film has "avalanched" into the trenches as illustrated in Figure 3.14(b), and these avalanched surfaces appear to be faceting as well. The avalanched regions are the most exaggerated in sections of the trench that have reflowed. These filled regions typically form saddle-like structures that would minimize the surface area if the film is constrained to reflow only within a localized section. The Cu films are continuous for both filled and unfilled profiles and show no sign of agglomeration on any of the sidewall regions. Also, EDX analysis reveals no sign of Ta migration into the Cu, or Cu through the Ta and Si_3N_4 into the surrounding SiO_2, and the solid solubility of Ta in Cu is much less than 1%[20]. Figure 3.15 shows images of a facet within a single grain and Figure 3.16 shows images of groove – anti-groove pairs with

the appropriate surface normals labeled in the image.

Figure 3.17 shows TEM images of the Cu(25°C)/W film that was reflowed at 500°C for 17 hours (wafer 14). This film reflowed in a considerably different fashion than those discussed previously in that there was some filling everywhere rather than merely filling in sections. In this film, the W barrier layer was insufficient for the anneal. The large, hexagonal shaped grain on the surface of this film is Cu_xSi_{1-x}. EDX imaging reveals that Cu had diffused through both the W and the Si_3N_4 and into the SiO_2, as clearly illustrated in Figure 3.17(b); the two dark spots in the SiO_2 are EDX beam damage marks. The beam damage mark closest to the Si_3N_4 contained Cu (which is also diffracting); the beam mark below it did not contain Cu. It is also interesting to note that the W barrier layer is clearly not continuous in (a) and appears to have broken into disconnected pieces. EDX imaging did not detect any W in the bulk of the Cu film, but only along the lower edge. This W barrier layer was unable to withstand a 500°C anneal for 17 hours.

Finally, Figure 3.18 shows TEM images of the Cu(150°C)/W that was reflowed at 500°C for 11 hours (wafer 16). The film surfaces show a considerable degree of faceting, as shown in (a) and (c). Also, 17 of 25 trenches imaged had agglomerated along one sidewall; a typical example is shown in (a). The trenches that still have a continuous film show a considerable degree of filling, as shown in (c) and (d)–typically the trenches were approximately half full. The surface of the grains also appeared to be much rougher than those for Wafer 11.

The mechanism(s) that are limiting Cu reflow on Ta or W underlayers is still unclear; however, several important features of these films have been illustrated. The surface energy anisotropy is clearly high for Cu at 500°C, because the desire to facet seems to be driving many of the morphologies that are seen in these films. Groove – anti-groove pairs could be extremely stable structures because twin boundaries that are formed between the grains forming the pairs have relatively low energy, and it seems that these pairs can greatly stabilize the unfilled trench structures. The grains are largely columnar (except where Si had diffused into the Cu in wafer 16), but the grain size distribution is much different along the sidewalls of the trenches than

along the flat regions between trenches, indicating some interaction between reflow and grain growth. Filled sections of the trenches typically show either a single grain or grains that are stacked from the bottom of the trench. Some of the filled regions also show grain boundaries (most often twin boundaries) that run at approximately 45° from near the trench bottom to near the top center of the trench. Unfilled sections typically have grain boundaries that are perpendicular to the surface along the sidewalls.

3.3 Reflow of Low Aspect Ratio Polycrystalline Cu Films

In a second attempt to determine the planarization mechanisms and appropriate constants, 1200 Å Cu films were deposited on 1000 Å deep trenches patterned in SiO_2, capped with Si_3N_4 and spaced 6 μm apart. These films were deposited in an ion-beam sputtering system with base pressure $< 5 \times 10^{-9}$ Torr as shown in Figure 3.19. The Cu deposition rate was 20 Å/min as estimated from an oscillating-crystal thickness monitor.

After deposition, the sample was annealed at 505°C for 8 hours as measured by an optical pyrometer using an emissivity of 0.13 for Cu. The base pressure of the system was $\leq 5\times10^{-8}$ Torr during the entire post-deposition anneal. Also, the sample was monitored by the decay of the diffracted peaks intensities in optical diffraction from the array of trenches using a diode laser operating at 670 nm with a spot diameter of approximately 1.5 mm. The reflected peak and the first three diffracted peaks intensities were monitored with photodiodes during the entire anneal, and these data are shown in Figure 3.20. This array of trenches should satisfy the small slope requirements leading to Equation 3.3, so the intensity should decay as shown in Equation 3.12 during the entire anneal. A cursory glance at Figure 3.20 reveals that it does not decay in this manner, however. The sample appeared a shiny Cu color after the anneal. Additionally, we examined the surface of the sample with an Auger

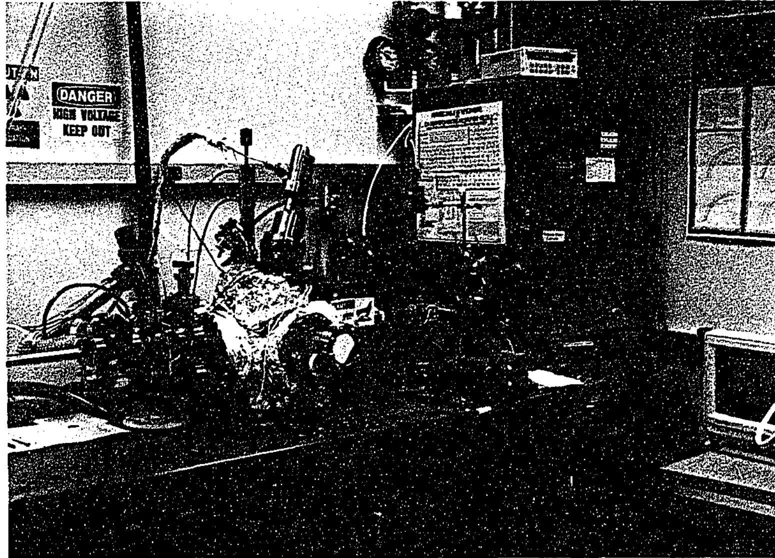


Figure 3.19: Ion-beam sputtering system. The glass tube to the right of the picture contains the sample during the optical diffraction experiments.

spectrometer to look for contamination of the surface. Oxygen and carbon were the only observable peaks besides Cu, however this is not surprising since there was a vacuum break prior to insertion into the Auger instrumentation; we did not detect a silicon peak, indicating that the Si_3N_4 barrier was still an effective barrier to Cu-Si reactions even after this long anneal.

Figure 3.20 has several interesting features: the normalized n=1 diffracted peak intensity is actually increasing during the anneal, the n=3 peak changes slope during the anneal, and after the n=3 peak has changed slopes, the n=2 and n=3 peaks have the same slope. This data does not obviously correspond to any of the decay mechanisms listed in Section 3.1. The surface diffusion constants have been measured at temperatures as low as 780 K [50] on single crystal Cu. The primary difference between their measurements and ours seems to be that their measurements were done on single crystal Cu and ours were done on a polycrystalline Cu film. Equation 3.12 assumes transport over a continuum surface, however, our films also contain grain boundaries, which will groove during the anneal. Also, we might expect significant interaction between the thin Cu film and the underlayer, both due to stresses in the

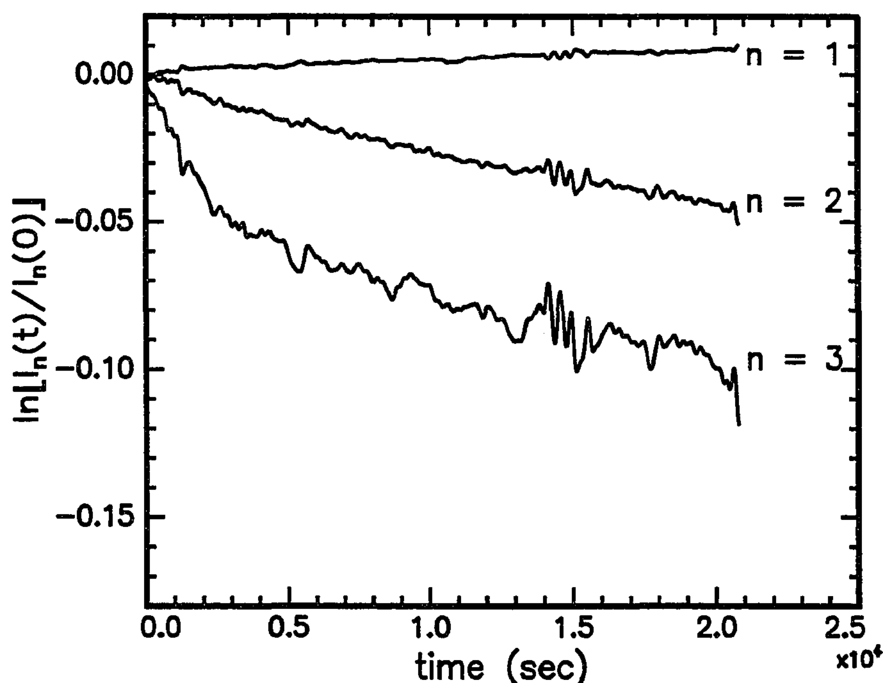


Figure 3.20: Diffraction results for a very low aspect ratio film annealed at 505°C. The normalized n = 1 intensity is increasing during the anneal, and the normalized n = 3 intensity changes slope during the anneal.

film and grain growth. It seems that even in the small slope case of reflowed Cu, there are significant interactions between surface diffusion and the specific morphology of the film. As in the case for the higher aspect ratio films, the diffracted peaks from these low aspect ratio films include information from areas that have reflowed completely and areas that have reflowed very little because the beam size is much larger than the local areas that have reflowed; the simple, continuum theory in Section 3.1 does not account for any spatial inhomogenaity of the reflow process.

3.4 Hot-stage TEM of Cu films

Hot-stage TEM experiments were performed to gain insight into the kinetics of reflow. Our sample consisted of 1000 Å Cu/250 Å W deposited (base pressure = 1.6×10^{-9} Torr, deposited in the magnetron sputtering system) on 1200 Å patterned SiO_2/Si.

Plan-view samples were prepared by dimpling, followed by backside ion milling. The prepared sample was then loaded into a hot-stage sample holder. To insure good thermal contact with the support ring, containing the heater, and thermocouple, a Pt ring was loaded followed by the Cu film, and clamped tightly in place with a c-clip. The Cu film surface was shiny upon insertion into the microscope. The base pressure of the microscope column was 2×10^{-7} Torr during the anneal.

A plan-view of the sample, before annealing, is shown in Figure 3.21 as imaged in a Philips 430 TEM with 300 keV electrons. In this figure, the trenches are 1 μm wide and the trench-to-trench spacing is 6 μm. The black "islands" in the trenches are on the order of 1 μm long and are regions where the Cu film has agglomerated and is too thick to image. The white "pools" around the islands are regions where there is no longer a Cu film. During ion milling, the sample was not cooled with liquid nitrogen, and heating caused by the milling resulted in the agglomeration of these regions. This conclusion is supported by the fact that the thicker regions of the sample had a much lower density of islands which correlates with a higher local thermal mass, and hence, less heating.

The sample was heated from room temperature to 320°C in 15 - 20 minutes. During this time there was no large scale movement of material. There was, however, some movement of grain boundaries, but it was very difficult to characterize these features due to the thickness of the sample; some of the fine lines in the surrounding material seemed to be moving by approximately 1000 Å on time scales on the order of 30-60 seconds. The sample temperature was ramped from 320°C to 340°C in approximately 5 minutes. At this point, there was rapid, large scale movement of material as shown in Figure 3.22. The island in Figure 3.22(a) is the same island as the lower right island in (b). Surprisingly, the appearance and thickening of the three additional islands shown in (b) took place within 5 seconds! The magnification of the microscope was 42500 during this time. Appearances of similar islands took place rapidly and seemed to be correlated, at this magnification, with the beam–as soon as the beam was moved to a new area, islands would appear within 10 seconds. However, it was not possible to induce island formation in the flat regions between

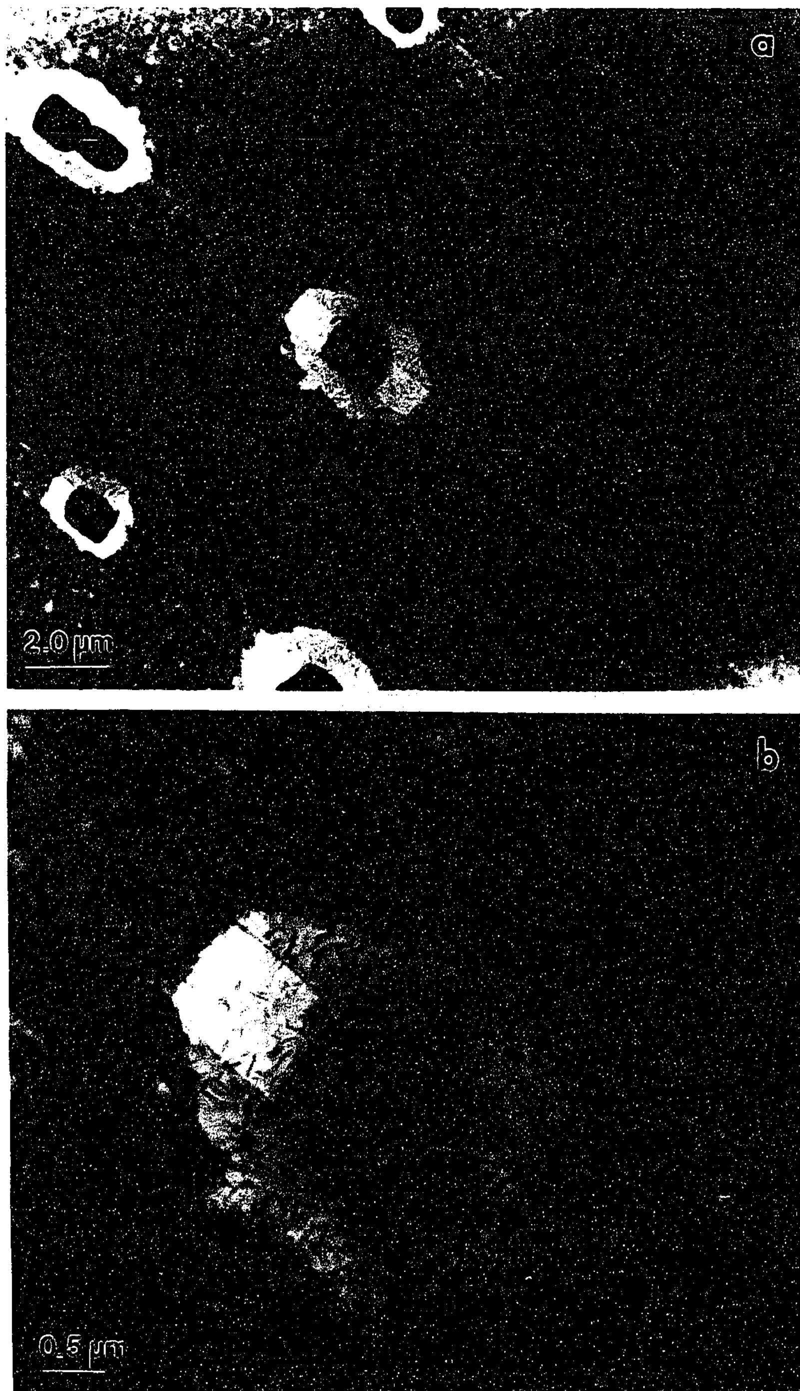


Figure 3.21: Plan-view of the 1000 Å Cu/250 Å W sample before annealing. The trenches are 1 μm wide and are 6 μm apart. The black "islands" in the trenches are regions of Cu agglomeration due to sample heating during preparation.

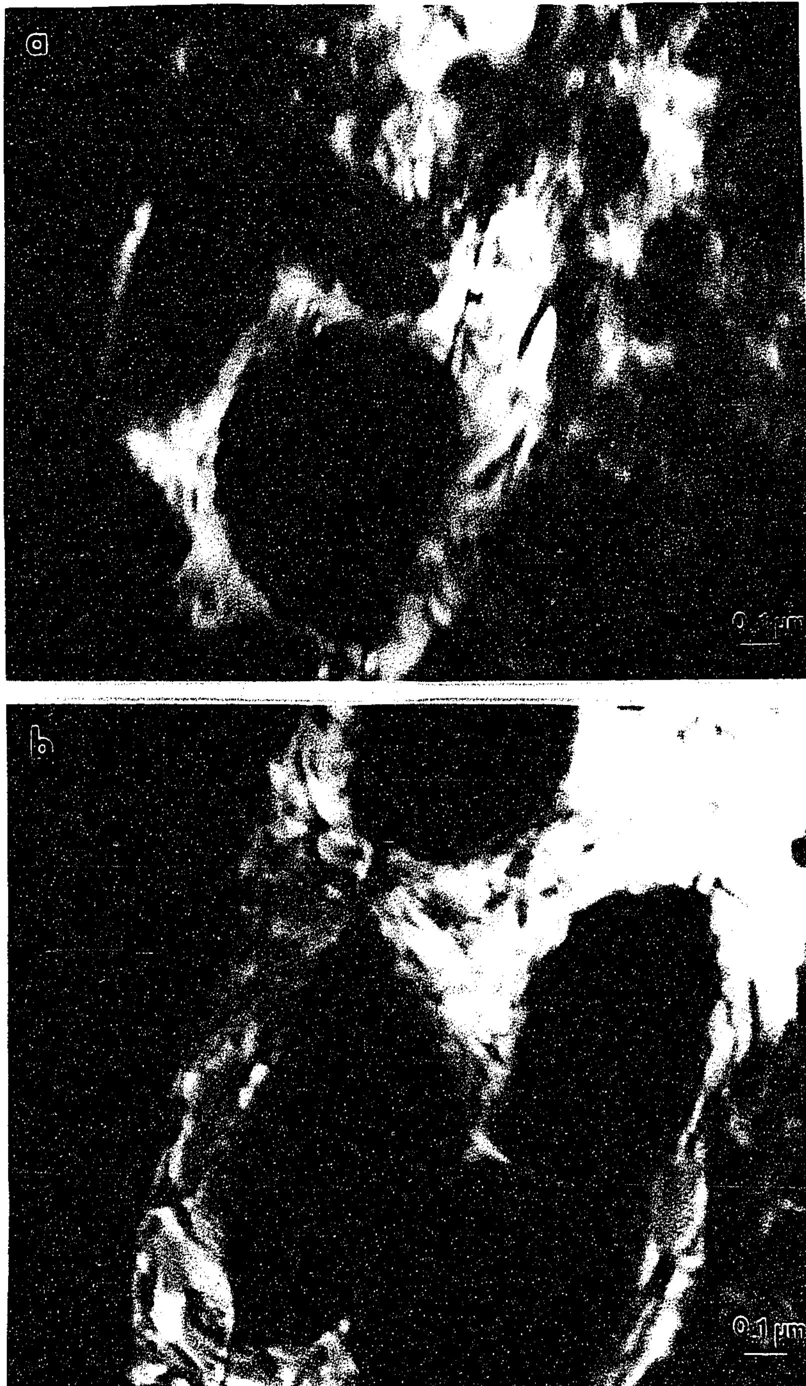


Figure 3.22: The sample temperature has been ramped to 340°C in 15-20 minutes. The island in (a) is the lower right island in (b). The appearance of the islands in (b) took place within 5 seconds.

the trenches with the beam. Moreover, at 13600 magnification, this effect seemed to be minimized, and the film agglomeration appeared to be relatively uniform over the area of sight; areas not previously scanned indicated that the agglomeration appeared to be happening over the entire sample.

The sample was heated from 320°C to 440°C in approximately 10 minutes. Figure 3.23(a) shows the sample after it was annealed for a total of 1 hour. The agglomeration continued to pull material from the flat portions of the sample until the entire flat region was devoid of Cu. It is interesting and perhaps relevant to note that the Cu grains are rectangular in shape, possibly indicating an orientation correlated with the substrate. Unfortunately, the island regions were too thick to image even with the aperture of the microscope open as far as possible, condenser 1 at 1, and the bias turned up as far as reason would allow. Upon removal from the microscope, the sample was no longer a pristine Cu color, but rather a dull W gray due to the agglomeration of the entire sample.

This experiment gives several important clues into Cu reflow. First, reflow can occur on extremely short timescales and can be extremely sporadic spatially. Even though this sample agglomerated, the result is consistent with the results of reflowing higher aspect ratio Cu films. Both the reflow process and the agglomeration process are driven by surface diffusion, however the agglomeration process is more complicated because the surface energy of the underlayer must be considered. Second, the reflow of this air-exposed Cu film at base pressures in the 10^{-7} Torr regime can happen at temperatures as low as 340°C. Exposing the Cu surface to air and annealing at higher base pressures could significantly affect the properties of the Cu surface. Finally, there is a possible orientational relationship between the Cu film and the W underlayer, which would make the choice of a barrier layer crucial for a Cu reflow process.

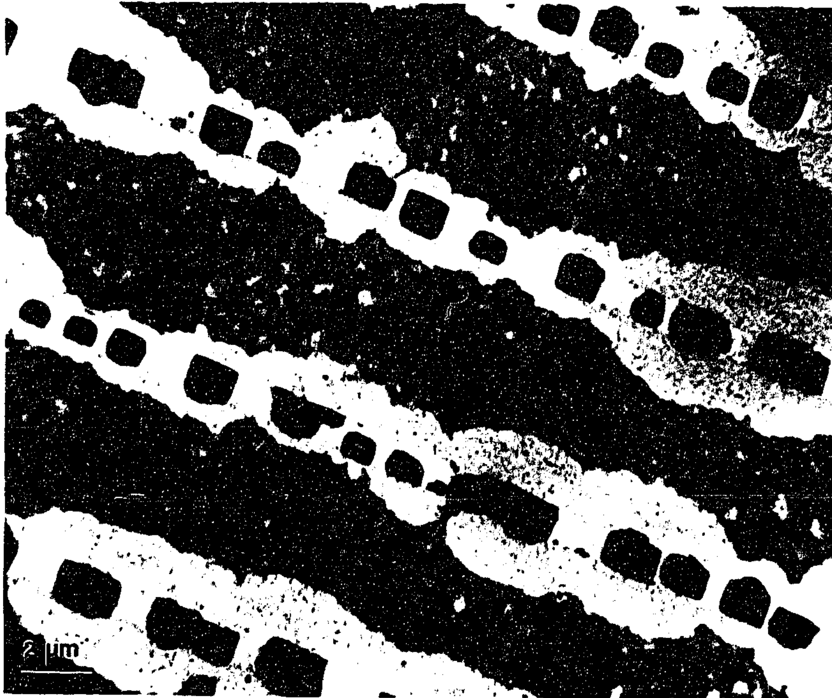


Figure 3.23: Sample after being ramped to 440°C over the course of 1 hour.

Chapter 4 Modeling Cu Planarization

There are several possible mechanisms by which a heated surface can flatten, including viscous flow, evaporation-condensation, volume diffusion, and surface diffusion[42]. For a polycrystalline Cu film annealed in ultrahigh vacuum, one can neglect viscous flow and evaporation-condensation as possible transport mechanisms. Typical literature values for the surface diffusion pre-exponential and activation energy are 0.07 cm^2/sec and 0.82 eV/atom[49], respectively, while typical values for the volume diffusion constants are 0.69 cm^2/sec and 2.18 eV/atom for Cu[50]. The 1/e decay time for a sinusoidal profile of frequency ω for surface and volume diffusion are

$$\tau_s = \frac{1}{B\omega^4} \quad \text{and} \quad \tau_v = \frac{1}{C\omega^3}, \tag{4.1}$$

respectively, where

$$B = \frac{D_s \gamma_s \Omega^2 \nu}{kT} \quad \text{and} \quad C = \frac{D_v \gamma_s \Omega}{kT}. \tag{4.2}$$

Here, D_s is the surface diffusivity, D_v is the volume diffusivity, γ_s is the surface energy per unit area, Ω is the atomic volume, ν is the number of atoms per unit surface area, k is Boltzmann's constant, and T is the temperature of the film. Using the values γ_s = 1800 ergs/cm^2[33], $\Omega = 1.2 \times 10^{-23}$ cm^3, and $\nu = 1.43 \times 10^{15}$ /cm^2 for Cu, and assuming a wavelength of 1 μm and T = 800 K, one finds that τ_s = 41 sec and τ_v = 1.6 x 10^6 sec. The surface and volume diffusion timescales will become comparable when

$$B\omega = C \;\Rightarrow\; \frac{D_s}{D_v} = \frac{1}{\Omega\nu\omega} \tag{4.3}$$

so for lengthscales $\leq 10\mu$m, surface diffusion dominates for $T \leq 1700$ K, which is greater than the melting point of Cu. These timescales are dependent upon the lengthscale of interest, but surface diffusion is clearly the dominant planarization mechanism for features less than 10 μm in size for Cu at temperatures less than 800

K.

4.1 Modeling Surface Diffusion–Mediated Reflow

Modeling the reflow of a Cu film into trenches and vias that have lengthscales on the order of 1 μm, then, requires modeling the capillary–driven surface diffusion of the film. We have developed a finite-element simulation based upon Mullins' surface diffusion equation for a general profile[31]. He assumed that the increase in chemical potential per atom that is transferred from a point of zero curvature to a point of curvature K on the surface is given by[51]

$$\mu(K) = K\gamma_s\Omega, \tag{4.4}$$

where Ω is the atomic volume, and the surface energy per unit area, γ_s, is assumed constant. Using the Nernst-Einstein relation, we find the average velocity, V, of the drifting surface atoms is

$$V = -\frac{D_s}{kT}\frac{\partial \mu}{\partial s}; \tag{4.5}$$

from this, we find the surface current, J, is

$$J = \nu V. \tag{4.6}$$

Now, we can calculate the speed of movement of a surface element along its normal

$$\begin{aligned} r_n &= -\Omega\frac{\partial J}{\partial s} & (4.7) \\ &= B\frac{\partial^2 K}{\partial s^2} & (4.8) \end{aligned}$$

where K(s) is the curvature, s is the arclength, and B is given in Equation 4.2. The constants used to obtain the results given here are γ_s = 1800 ergs/cm^2, D_{s0} = 0.07 cm^2/sec, Q_s = 0.82 eV/atom, $\Omega = 1.2 \times 10^{-23}$ cm^3 and $\nu = 1.43 \times 10^{15}$/cm^2.

4.1.1 Some Considerations

Using different constants will change the reflow time, but will not alter the topology of the evolving profile, as can be easily seen by considering Equation 4.8. If a profile of curvature K is reflowed using different constants, B and B', then

$$r_n = B\frac{\partial^2 K}{\partial s^2} \text{ and} \tag{4.9}$$

$$r'_n = B'\frac{\partial^2 K}{\partial s^2}, \tag{4.10}$$

and clearly the rates and timescales will scale as

$$\frac{r_n}{r'_n} = \frac{B}{B'} \quad \Rightarrow \quad \frac{t}{t'} = \frac{B'}{B} \tag{4.11}$$

respectively.

It is also revealing to consider the small slope approximation of Equation 4.8. Using

$$r_n = (1+y')^{-\frac{1}{2}}\frac{\partial y}{\partial t}, \tag{4.12}$$

$$K = -y''(1+y')^{-\frac{3}{2}}, \text{ and} \tag{4.13}$$

$$\partial s^2 = \partial x^2 + \partial y^2 \tag{4.14}$$

where $y' = \frac{\partial y}{\partial x}$, Equation 4.8 becomes

$$\frac{\partial y}{\partial t} = -B\frac{\partial}{\partial x}[(1+y'^2)^{-\frac{1}{2}}\frac{\partial}{\partial x}(\frac{y''}{(1+y')^{\frac{3}{2}}})] \tag{4.15}$$

where we have not made any assumptions except that the function $y(x,t)$ is one-to-one, so that y and its derivatives are unique. In the small slope limit ($y' \to 0$), this reduces to

$$\frac{\partial y}{\partial t} = -B\frac{\partial^4 y}{\partial x^4}. \tag{4.16}$$

If we have a sinusoidal profile

$$y(x,t) = a(t)\sin\omega x \tag{4.17}$$

with $a(0)\omega \ll 1$ (i.e. small slope), then

$$a(t) = a(0)e^{-B\omega^4 t} \tag{4.18}$$

so

$$y(x,t) = y(x,0)e^{-B\omega^4 t}. \tag{4.19}$$

Similarly, if we construct a profile using a Fourier series

$$y(x,t) = \sum_{n=0}^{\infty}[a_n(t)\cos\omega_n x + b_n(t)\sin\omega_n x] \tag{4.20}$$

with $\omega_n = 2\pi n/\lambda \equiv n\omega$, $a_n(0)\omega_n \ll 1$, and $b_n(0)\omega_n \ll 1$, we find that

$$a_n(t) = a_n(0)e^{-B\omega_n^4 t} \tag{4.21}$$

$$b_n(t) = b_n(0)e^{-B\omega_n^4 t}. \tag{4.22}$$

Then the profile evolves as

$$y(x,t) = \sum_{n=0}^{\infty}[a_n(0)e^{-B\omega_n^4 t}\cos\omega_n x + b_n(0)e^{-B\omega_n^4 t}\sin\omega_n x] \tag{4.23}$$

$$= \sum_{n=0}^{\infty}[a_n(0)e^{-B\omega^4 n^4 t}\cos n\omega x + b_n(0)e^{-B\omega^4 n^4 t}\sin n\omega x], \tag{4.24}$$

which will rapidly evolve to

$$y(x,t) = a_0(0) + a_1(0)e^{-B\omega^4 t}\cos\omega x + b_1(0)e^{-B\omega^4 t}\sin\omega x \tag{4.25}$$

since $a_n(t) \sim e^{-B\omega^4 n^4 t}$. High aspect ratio trenches cannot be modeled in this manner, however, since this solution restricts each component of the Fourier series to have a small slope.

It is also relevant to note that if we scale a profile by $1/m$, the topology of the reflow will be self-similar to the unscaled profile. If we assume that

$$(x', y') = \frac{1}{m}(x, y), \tag{4.26}$$

then the arclength, s, and the curvature, K, will scale as

$$s' = \frac{1}{m}s \tag{4.27}$$

$$r' = \frac{1}{m}r \quad \Rightarrow \quad K'(s') = mK(s) \tag{4.28}$$

so that

$$r'_n = \sqrt{(\frac{\partial x'}{\partial t})^2 + (\frac{\partial y'}{\partial t})^2} \tag{4.29}$$

$$= \frac{1}{m}\sqrt{(\frac{\partial x}{\partial t})^2 + (\frac{\partial y}{\partial t})^2} \tag{4.30}$$

$$= \frac{1}{m}r_n \text{ and} \tag{4.31}$$

$$B\frac{\partial^2 K'(s')}{\partial s'^2} = B\frac{\partial}{\partial s}[\frac{\partial (mK(s))}{\partial s}\frac{\partial s}{\partial s'}]\frac{\partial s}{\partial s'} \tag{4.32}$$

$$= B\frac{\partial}{\partial s}[\frac{\partial (mK(s))}{\partial s}m]m \tag{4.33}$$

$$= m^3 B\frac{\partial^2 K(s)}{\partial s^2} \tag{4.34}$$

Hence the reflow time will scale as $1/m^4$ for a profile that has been scaled by $1/m$.

The surface diffusivity is, in principle, orientation dependent, but the constants used here are typical for the (111), (100), and <110> direction in the (110) plane[50]. Cu has an anisotropic diffusivity in the (110) plane, and the <001> direction in this plane has a much lower diffusivity. This lower diffusivity can only limit the kinetics if the <001> direction of the (110) plane is along the direction of atomic transport. A typical lengthscale for the <110>(110) direction is ~ 5 μm, whereas a typical diffusion lengthscale for the <001>(110) is ~ 20 Å for one second at 800 K. For diffusion in the (110) plane, the diffusion coefficient in the direction Φ, where Φ is the angle relative

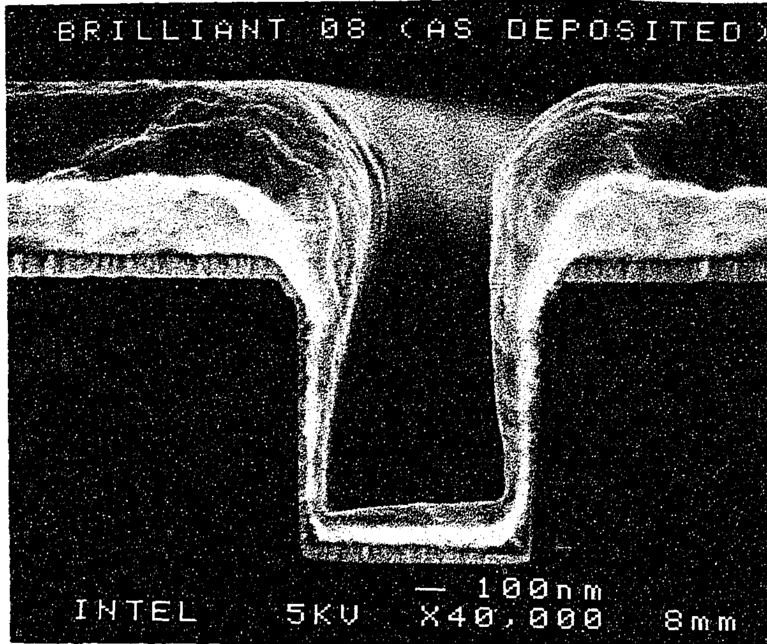


Figure 4.1: SEM image of an as-deposited Cu film from a conical magnetron sputtering system that was digitized and used as the initial profile in these simulations.

to <110> (the direction with the maximum diffusivity), is given by[50]

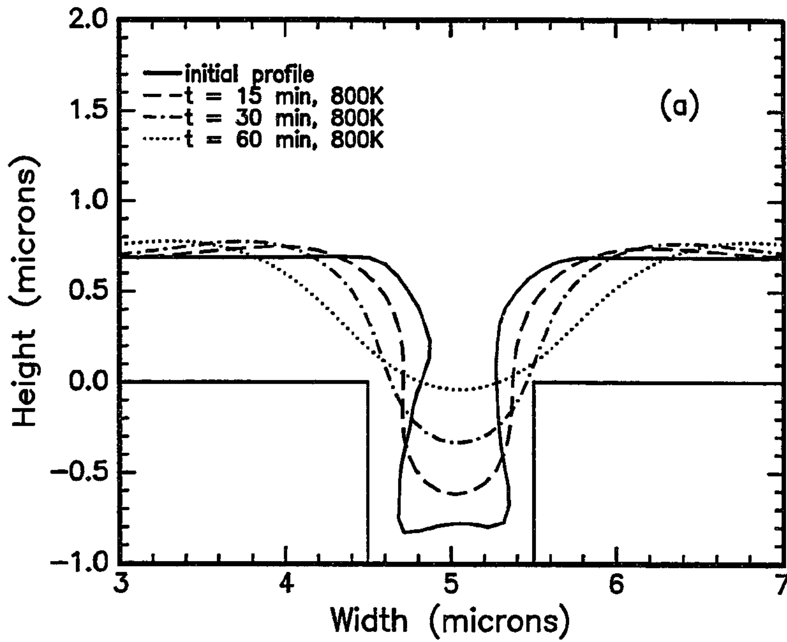
$$D_{(110)}(\Phi) = D_{<110>} \cos^2 \Phi + D_{<001>} \sin^2 \Phi. \tag{4.35}$$

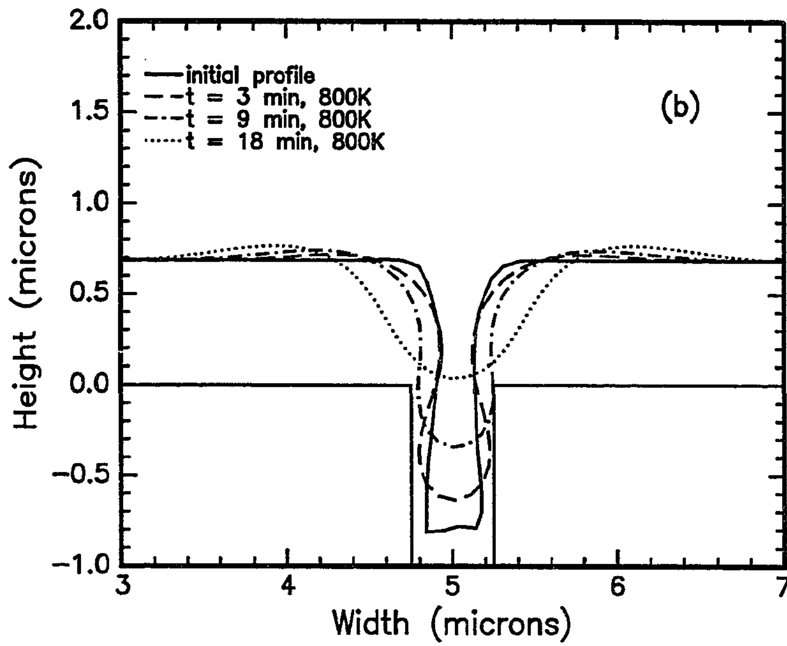
The initial profile for simulations was obtained by digitizing an experimental as-deposited profile for a 1 μm by 1 μm trench from the magnetron sputtering source described previously. An SEM image of the initial profile is shown in Figure 4.1. The initial profile for a 0.5 μm by 1 μm trench was obtained by scaling the 1 μm by 1 μm profile by one-half in the x-direction. This approach simplifies comparison of trenches of varying aspect ratios by avoiding any effects of the initial profile's topology; however, it also puts an excessive amount of material on the sidewalls as compared with experiment for aspect ratios greater than 1:1. For an isolated trench, the boundary conditions used in the simulation are that the surface is flat far from the non-planar portion of the film; that is, $\mathrm{K}(s \to \pm\infty) = 0$.

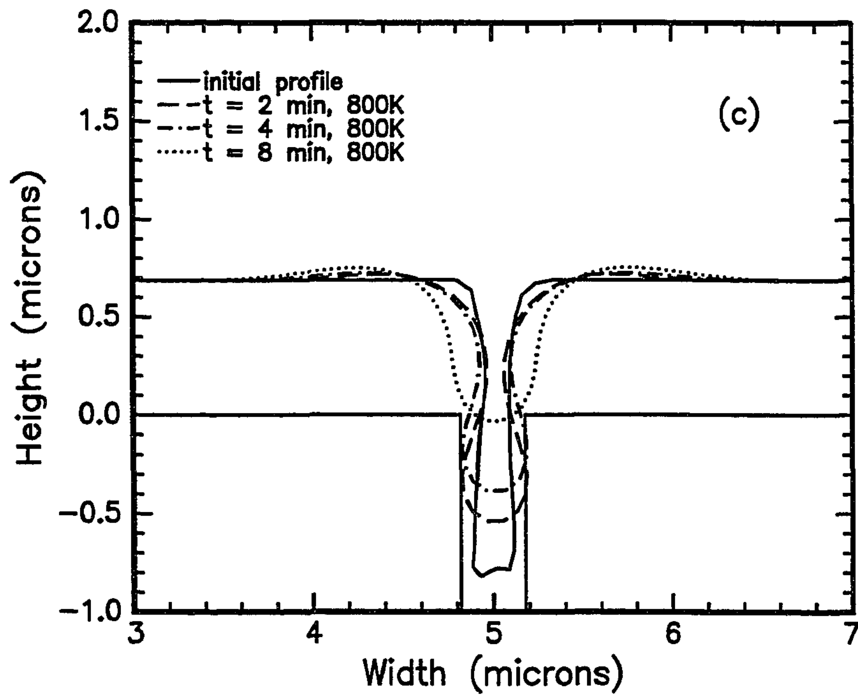
4.1.2 Results

Figure 4.2(a) shows time slices from reflow of a 0.75 μm thick Cu film at 800 K into a 1 μm by 1 μm trench, Figure 4.2(b) depicts reflow into a 0.5 μm by 1 μm, and Figure 4.2(c) depicts reflow into a 0.33 μm by 1 μm trench. As aspect ratios become greater than 1:1, the profile evolution is far different from that expected in the small slope limit. In particular, the profile initially evolves into an "hourglass" shape by creating regions of constant curvature at both top shoulders and at the trench bottom. This hourglass profile is metastable, as a circle is a stable solution to the surface diffusion equation for a closed surface. The only unstable regions are at the inflection points near the center of the trench sidewalls and on the outside of both trench shoulders, and there is little transport of material between the top shoulders and the bottom region of the trench. The instabilities at the inflection points slowly force the trench to fill in order for the surface to flatten. As the surface continues to reflow, it must eventually abandon this hourglass profile due to the instabilities at the inflection points, which force the shoulders outwards at long times, so the profile evolution eventually approaches that of the sinusoidal profile.

We have now revealed several problems for the experimental success of reflow in high aspect ratio trenches. The first problem is that of agglomeration of the Cu film. At the beginning of the anneal, the Cu along the sidewalls is thinned and this material is deposited at the bottom of the trench, which can cause the agglomeration of the Cu film along the sidewalls. For the initial profile used here, the Cu film will agglomerate at an aspect ratio of approximately 3:1. The second problem is that the shoulders at the top of the trench are forced into circular regions which can create additional overhang near the top of the trench, possibly causing the Cu to bridge across the trench. This imposes a fundamental limit on a post-deposition reflow process in very high aspect ratios. Our simulations suggest that for the assumed initial profile, the process becomes unfeasible due to these overhanging shoulders when the width has been scaled to approximately 0.2 μm. We have calculated the filling time versus the trench width and find that $t \sim w^{1.6}$ for aspect ratios greater than 1:1. This is a very


initial profile
t = 15 min, 800K
t = 30 min, 800K
t = 60 min, 800K
(a)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7


initial profile
t = 3 min, 800K
t = 9 min, 800K
t = 18 min, 800K
(b)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7



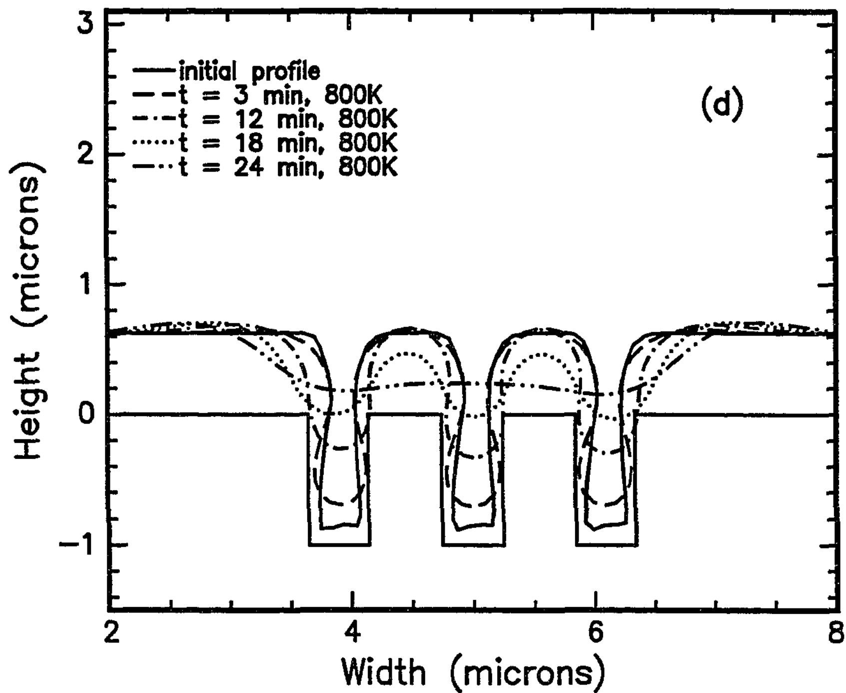


Figure 4.2: Simulation of Cu reflow at 800 K for (a) 1 μm by 1 μm trench with t_{fill} = 68 minutes; (b) 0.5 μm by 1 μm trench with t_{fill} = 17 minutes; (c) 0.33 μm by 1 μm trench with t_{fill} = 8 minutes; (d) three 0.5 μm by 1 μm trenches that are spaced by 0.5 μm apart with t_{fill} = 19 minutes.

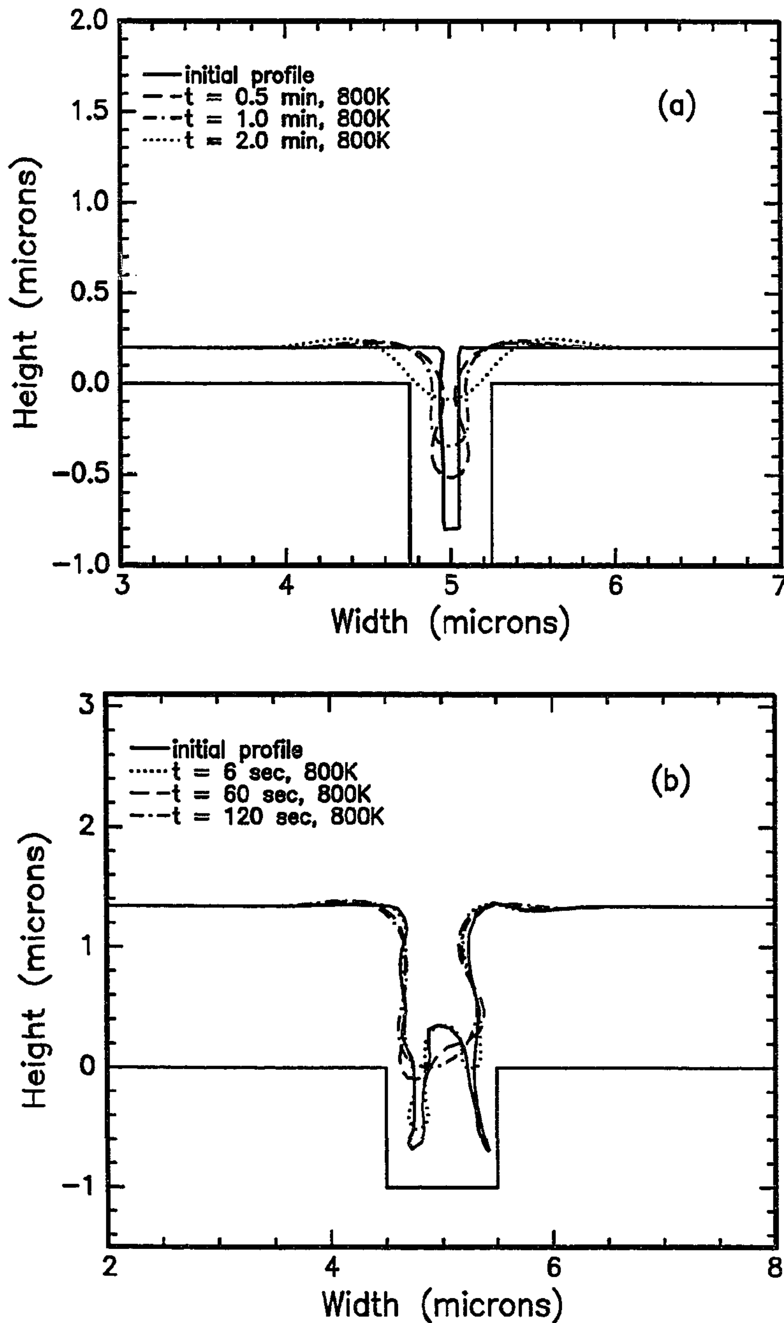


Figure 4.3: (a) 0.2 μm thick conformal film on a 0.5 μm by 1.0 μm trench with t_{fill} = 3.5 minutes; (b) reflow of profile from a highly collimated target on a 1 μm by 1 μm trench with t_{fill} = 1.9 minutes.

different scaling law than that expected from a small slope approximation for periodic structures, which gives t $\sim w^4$.

We have also considered the effect of nested trenches, and the results of three 1 μm by 0.5 μm are shown in Figure 4.2(d). Figure 4.2(b) shows that a trench can be treated as isolated if it is at least 2 μm from its nearest neighbor, while Figure 4.2(d) indicates that even nested trenches can be treated as isolated during the initial stages of reflow. During most of the reflow process the adjacent trenches are essentially non-interacting except that they form stable, circular regions between the top of the trenches. It is not until the trenches are almost filled that there is any loss of material from these very stable shoulder regions.

The evolution of an hourglass profile is not limited to the initial profile used in Figure 4.2. Figure 4.3(a) shows the evolution of a 0.2 μm thick conformal profile in a 0.5 μm by 1.0 μm trench. There is clearly sidewall thinning and shoulder formation on this initially conformal profile. A typical profile obtained from a highly collimated sputtering source is shown in Figure 4.3(b). Note here that the two "fingers" evolve separately, as would be expected from consideration nested, high aspect ratio trenches as shown in Figure 4.2(d). In an experiment, it is also possible that the very long, thin fingers in this collimated profile will bridge near the top before the finger has completely filled, thus forming a small void in the trench. Nonetheless, the evolution of this profile illustrates the extent of interaction between adjacent, high curvature features.

4.2 Modeling Reflow with Surface Energy Anisotropy

The surface profile evolution described above does not include any of the complications and intricacies of the Cu surface. The surface energy plot of Cu is known to be anisotropic with $\gamma_{111}/\gamma_{100} = 0.994$, $\gamma_{110}/\gamma_{100} = 1.011$, and $\gamma_{max}/\gamma_{100} = 1.015$ at 1030°C for a clean Cu surface annealed in dry hydrogen, obtained by measurements on thermally equilibrated grain boundary grooves in wires exhibiting a bamboo structure[47]. If the surface energy is anisotropic, the surface can lower its overall energy by faceting

as shown in Figure 4.4. Also, impurities (intentional or not) can enhance or reduce the anisotropy of the surface energy. There should be cusps in the surface energy for T = 0 K at every rational Miller index, but the depth of the cusp should be a rapidly decreasing function of index. For a 2-dimensional vicinal surface in the [1n] direction, we find[52]:

$$\Delta(\frac{d\gamma_s}{d\theta}) \sim \frac{1}{n^4}. \tag{4.36}$$

Also, the Cu (110) surface undergoes a roughening transition at around 1000 K[53], the Cu (113) surface has a roughening transition at 620 K, and higher order surfaces roughen at even lower temperatures[54]. However, we do not expect the close-packed (111) and (100) surfaces to undergo a roughening transition at temperatures less than the melting point of Cu[55]. Contrary to the near-noble and noble metals (such as Ir, Pt, and Au) which have a (2×1) missing-row reconstruction, where every second $[1\bar{1}0]$ row of atoms is missing, the Cu (110) surface has an unreconstructed, bulk-terminated (1×1) structure[56]. The tendency to reconstruct is intimately related to the relative strength of the first and second nearest-neighbor interactions parallel and perpendicular to the $[1\bar{1}0]$ rows, respectively, and is related to the surface energy anisotropy[57].

Hence, to model Cu reflow experiments at 500°C, the surface energy anisotropy should be included near the (111), (100) and (110) surfaces, at least. Thus we have added the effect of a surface energy anisotropy to the surface diffusion–mediated model for planarization of high aspect ratio trenches. If we no longer assume that the surface energy is isotropic, the chemical potential for a curved surface is greater than that for the flat surface by[58]:

$$\mu = K(s)[\gamma_s(\theta) + \frac{\partial^2 \gamma_s(\theta)}{\partial \theta^2}]\Omega. \tag{4.37}$$

Following the derivation in Section 4.1, we find that the rate of motion of a surface

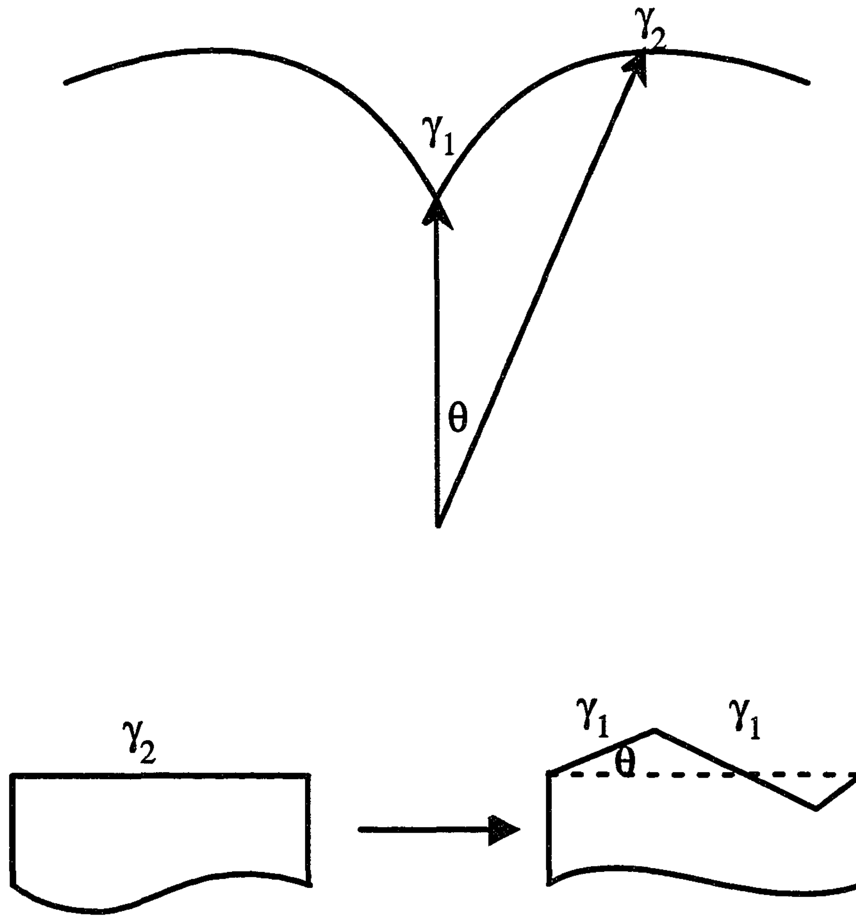


Figure 4.4: A surface with an anisotropic surface energy will have a Wulff plot with $\gamma_1 < \gamma_2$ where γ_1 is the surface energy for a low index surface. In this case, the surface can lower its energy by faceting even though the surface area is increased.

element along its normal is

$$r_n = B\frac{\partial^2}{\partial s^2}(\frac{1}{\gamma_s})[\gamma_s(\theta) + \frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]K(s) \tag{4.38}$$

for a surface with an anisotropic surface energy.

4.2.1 Choosing the Form of $\gamma_s(\theta)$

We might expect that the extent of faceting during Cu reflow should depend mainly upon the torque term, $\frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}$, because the maximum anisotropy of $\gamma_s(\theta)$ is only about

2% for Cu at 1030°C, even though we expect the anisotropy to be much greater at temperatures $\leq$ 800°C. Unfortunately, it is difficult to measure $\gamma_s(\theta)$ experimentally, especially in the detail needed near the low index planes where the torque term will vary the most. Data have been reported for Cu only for a single temperature[47]. With this limited information, we have chosen to follow the assumptions made in Reference [59] to fit $\gamma_s(\theta)$, viz.

$$\gamma_s(\theta^*) = \gamma_s(1 - \alpha g(\theta^*)) \tag{4.39}$$

$$g(\theta^*) = 2e^{-\beta(1-|cos\theta^* - sin|\theta^*||)} - 1 \tag{4.40}$$

$$\theta^* = \sqrt{\theta^2 + \frac{1}{G^2}} - \frac{1}{G} \tag{4.41}$$

where 2α is approximately the maximum surface energy anisotropy, β is the curvature of $\gamma_s(\theta)$, and G is a constant that determines the local curvature of $\gamma_s(\theta)$ in the vicinity of a cusp in the surface energy. To fit near the (111) plane of Cu at 1030°C, the best values for α and β are $\alpha = 0.012$ and $\beta = 7$, respectively. The most important and difficult part of fitting $\gamma_s(\theta)$ is near $\theta = 0°$, where at T = 0 K an infinitely large curvature exists. Depending on the choice of G, the local curvature of $\gamma_s(\theta)$ in the vicinity of $\theta = 0°$ can be made large (large G) or small (small G). Consequently, the function $\frac{1}{\gamma_s}[\gamma_s(\theta) + \frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ can also show great variation with G. We might expect the driving force for faceting to greatly increase as G is increased or as the anisotropy is increased, and we explore these dependencies below for several cases because accurate data for Cu at 800 K have not been reported.

For simulations of high aspect ratio trenches, additional assumptions are required. First, we assume that the surface of the film far from the trench is (111) in texture, which corresponds to the preferred texture of our sputtered Cu films. Also, we assume that we are tilting around the $(01\bar{1})$ axis and we use Equations 4.39, 4.40, and 4.41 to model the surface energy cusps in the regions near each of the (111), (110) and (100) planes individually.

4.2.2 Results

As a first, and simplest, attempt to model reflow with an anisotropic surface energy, we fit the surface energy around the (111), (100) and (110) planes with the same values of α, β, and G, as shown in Figure 4.5(a) and (b). The results are shown in Figure 4.5 for a 0.5 μm by 1 μm trench. This G corresponds to $\frac{1}{\gamma_s}[\gamma_s(\theta) + \frac{\partial^2 \gamma_s(\theta)}{\partial \theta^2}] =$ 38.8 for $\theta = 0°$[59]. There are several differences between these profiles for a 0.5 μm by 1.0 μm trench and those for the isotropic surface energy case shown in Figure 4.2(b). Including an anisotropic surface energy increases the reflow time from 17 minutes to 27 minutes. The surface energy anisotropy also decreases the width of the trench shoulders by forcing the material that was close to the (111) plane to adopt the (111) facet. Facets appear along the shoulders at 54.7°, which correspond to faceting along the (100) direction. The combination of the faceting along the (111) direction and the (100) direction tends to increase the height of the shoulders on each side of the trench. Also, the surface energy anisotropy can create facets at the bottom of the trench at 54.7° and -54.7°, thereby forcing the removal of material from the bottom of the trench. During the profile evolution, there are also many short-lived facets that do not appear in the timeslices shown in Figure 4.5. The creation of facets seems to form regions of great stability, which tend to lengthen the reflow time. Also, the trench bottom is no longer monotonically increasing in height. At times the bottom of the trench will facet to the (111) direction and at times it will facet so that it has a sharp point at the bottom center of the trench and has (100) and ($\bar{1}$00) facets to each side. Figure 4.5(a) and (b) only include $0° < \theta < 126°$ because the film normal during the evolution never extended into the region $126° < \theta < 180°$.

Now, we will assume a more realistic form for $\gamma_s(\theta)$ and fit the Cu surface energy plot at 1030°C from Reference [47]. A reasonable fit can be found by using $\gamma_s(111)$ = 1789.2 ergs/cm^2, $\alpha(111) = 0.012$, $\beta(111) = 7$, $\gamma_s(100) = 1800$ ergs/cm^2, $\alpha(100) =$ 0.007, $\beta(100) = 5$, and $G = 225$ for all orientations. Figure 4.6(a) and (b) show a plot of the surface energy and the torque term, respectively, and Figure 4.6(c) shows the results of this model. Figure 4.5(c) and Figure 4.6(c) are similar, except that

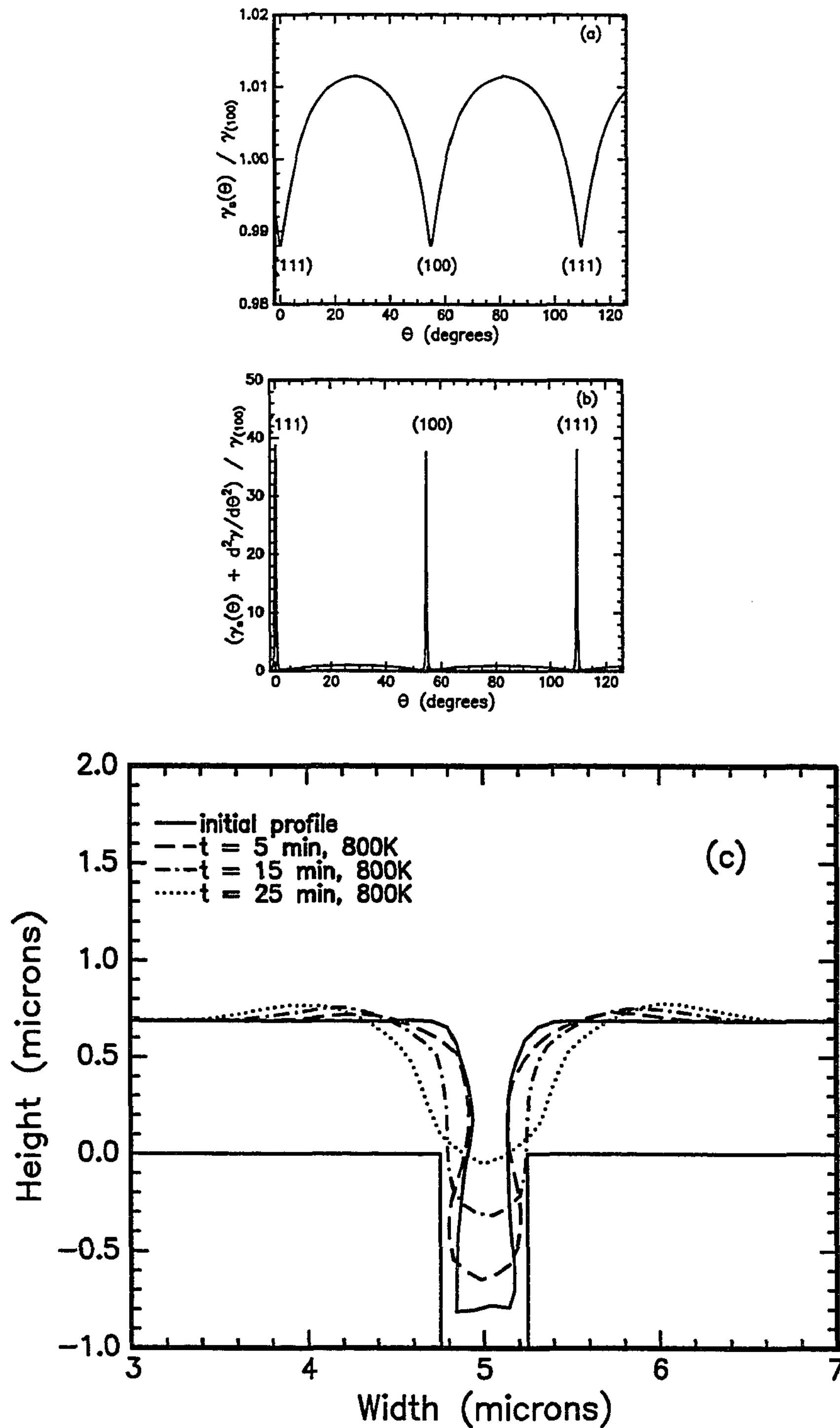


Figure 4.5: (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta)+\frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ using $\gamma_s(111) = \gamma_s(100) = \gamma_s(110)$ $= 1800$ ergs/cm^2, $\alpha = 0.012$, $\beta = 7$ and $G = 225$. (c) Reflow of a 0.5 μm by 1 μm trench, assuming an anisotropic surface energy as shown in (a) and (b). Note the formation of facets along the y-axis and at 54.7° from the y-axis, and the suppression of tails on the outside of the shoulders near the top of the trench. $t_{fill} = 27$ minutes.

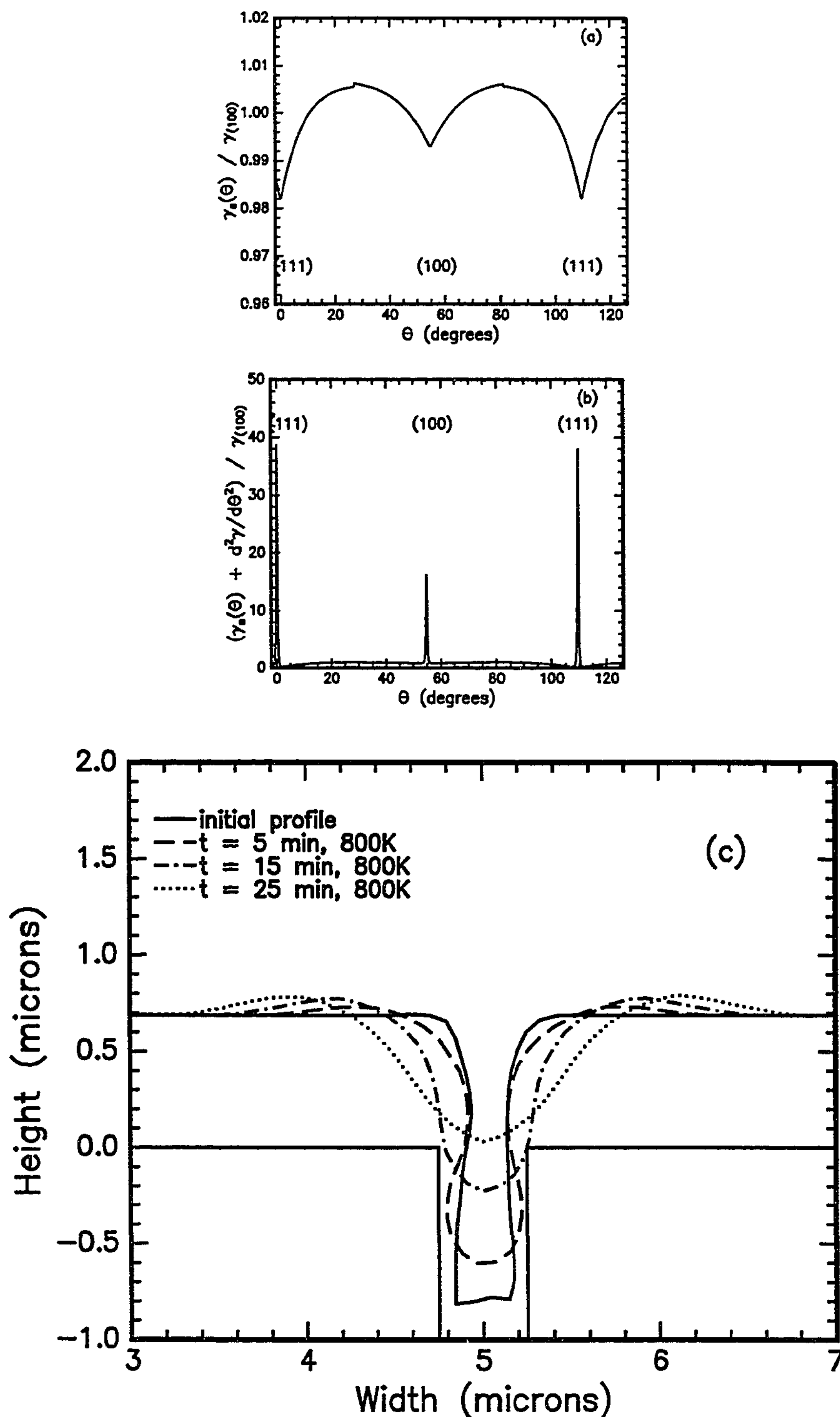


Figure 4.6: (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta) + \frac{\partial^2 \gamma_s(\theta)}{\partial \theta^2}]$ using $\gamma_s(111) = 1789.2$ ergs/cm^2, $\alpha(111) = 0.012$, $\beta(111) = 7$, $\gamma_s(100) = 1800$ ergs/cm^2, $\alpha(100) = 0.007$, $\beta(100) = 5$, and $G = 225$. (c) Reflow of a 0.5 μm by 1.0 μm trench assuming an anisotropic surface energy as shown in (a) and (b). $t_{fill} = 24$ minutes.

the (100) facets are not as pronounced and the reflow time is slightly shorter in the latter, which can be attributed to the fact that the surface energy cusp for the (100) orientation is not as deep.

We should also consider the effect of G on the evolving profile because it determines how sharp the cusp is near the low index planes. For the plot shown in Figure 4.7, we only decreased G with respect to Figure 4.6 and we found that the evolving profile is still faceting, but with less severity, and that t_{fill} was lowered to 22 minutes for a 0.5 μm by 1.0 μm trench.

Finally, we increased the anisotropy of $\gamma_s(\theta)$ because we would expect a much greater anisotropy at lower temperatures. Figure 4.8 shows results using $\gamma_s(111) = \gamma_s(100) = \gamma_s(110) = 1800$ ergs/cm^2, $\alpha = 0.024$, $\beta = 7$, and $G = 225$. We have increased the anisotropy to almost 5% and we can see that the profile evolves very differently. In particular, we see the formation of multiple facets along the initially flat surface and the agglomeration of the film along the sidewall. The development of facets along the initially flat top surface is particularly intriguing. During reflow with an isotropic surface energy, shoulders develop on each side of the trench top. When the anisotropic surface energy is added to the equations, these mounds still form, but then are pulled into the (100) facet on the left of the trench. This facet develops further and forms a dent into the initially flat surface, and this dent is pulled into the ($\bar{1}00$) facet as the evolution continues. These sorts of facets have been seen in the TEM pictures shown in Figure 3.16, and the experimental explanation of this phenomena was first attributed to an anisotropic surface energy and the inclusion of twin boundaries by Mykura [60, 61]. The kinetics are clearly dominated by the desire to facet in order to minimize the surface energy in the directions of low index planes and *not* by the minimization of the curvature. It is extremely unfortunate that better data has not been reported for Cu at lower temperatures so that the competition between these driving forces could be accurately modeled.

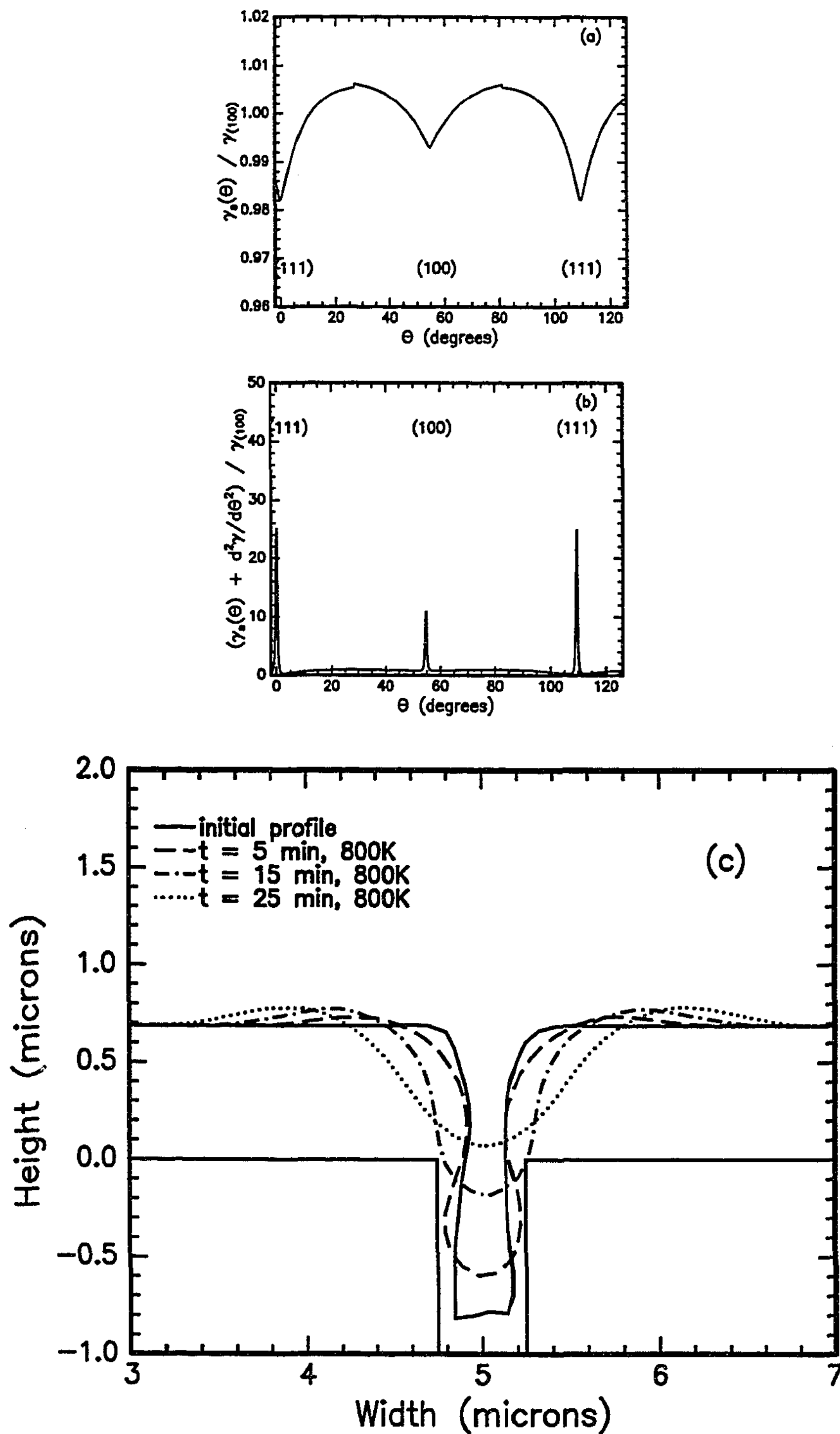


Figure 4.7: (a) $\frac{\gamma_s(\theta)}{\gamma_s(100)}$ and (b) $\frac{1}{\gamma_s(100)}[\gamma_s(\theta)+\frac{\partial^2\gamma_s(\theta)}{\partial\theta^2}]$ using $\gamma_s(111) = 1789.2$ ergs/cm^2, $\alpha(111) = 0.012$, $\beta(111) = 7$, $\gamma_s(100) = 1800$ ergs/cm^2, $\alpha(100) = 0.007$, $\beta(100) = 5$, and $G = 144$. (c) Reflow of a 0.5 μm by 1.0 μm trench assuming an anisotropic surface energy as shown in (a) and (b). $t_{fill} = 22$ minutes.

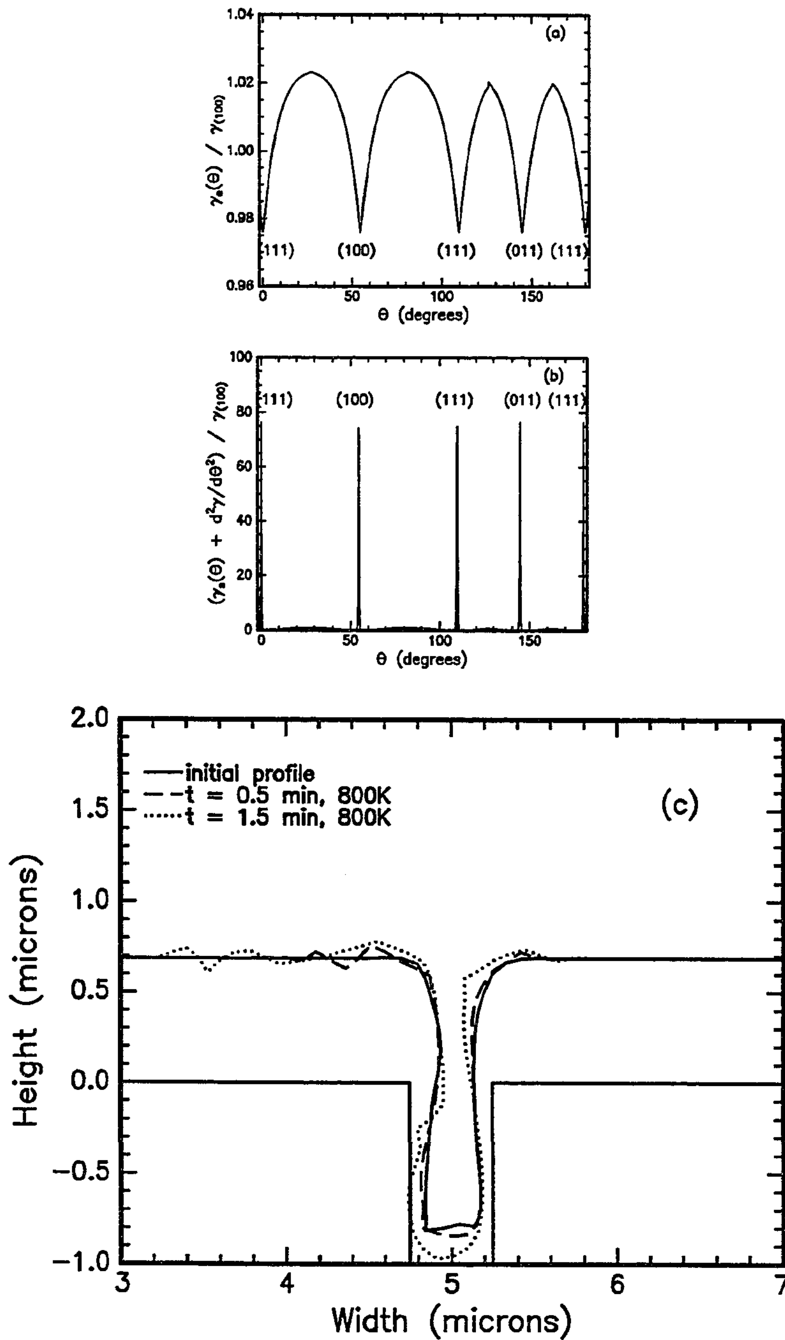


Figure 4.8: Reflow of a 0.5 μm by 1.0 μm trench, assuming $\gamma_s = 1800$ ergs/cm^2, $\alpha = 0.024$, $\beta = 7$, and $G = 225$.

4.3 Modeling Reflow with Grain Boundary/Surface Interactions

4.3.1 Grain Boundary Intersecting an Otherwise Flat Surface

Mullins has modeled the thermal grooving for an isolated grain boundary that is intersecting an initially flat surface[31]. In this case, it is reasonable to use the small slope approximation of the surface diffusion equation given in Equation 4.16. We can assume a symmetric solution across the grain boundary and require that Equation 4.16 be solved for a function y(x,t) subject to the following boundary conditions: the surface is initially flat, there is a fixed grain boundary groove angle, the current of atoms out of the grain boundary vanishes, the surface remains flat far from the groove, and the slope of the surface is zero far from the groove, i.e.

$$y(x,0) = 0 \tag{4.42}$$

$$\frac{\partial y(0,t)}{\partial x} = tan\beta \tag{4.43}$$

$$\frac{\partial^3 y(0,t)}{\partial x^3} = 0 \tag{4.44}$$

$$y(\infty,t) = 0 \tag{4.45}$$

$$\frac{\partial y(\infty,t)}{\partial x} = 0 \tag{4.46}$$

where $\sin\beta = \frac{\gamma_{gb}}{2\gamma_s}$. Using these boundary conditions, it can be shown[31] that

$$y(x,t) = (tan\beta)(Bt)^{\frac{1}{4}} Z[\frac{x}{(Bt)^{\frac{1}{4}}}] \tag{4.47}$$

where Z can be developed as a power series. This depth of the groove measured in the y direction from the maximum of the surface to the grain boundary and the separation of the two maxima are given by

$$d = 0.973\,(tan\,\beta)(Bt)^{\frac{1}{4}} \text{ and} \tag{4.48}$$

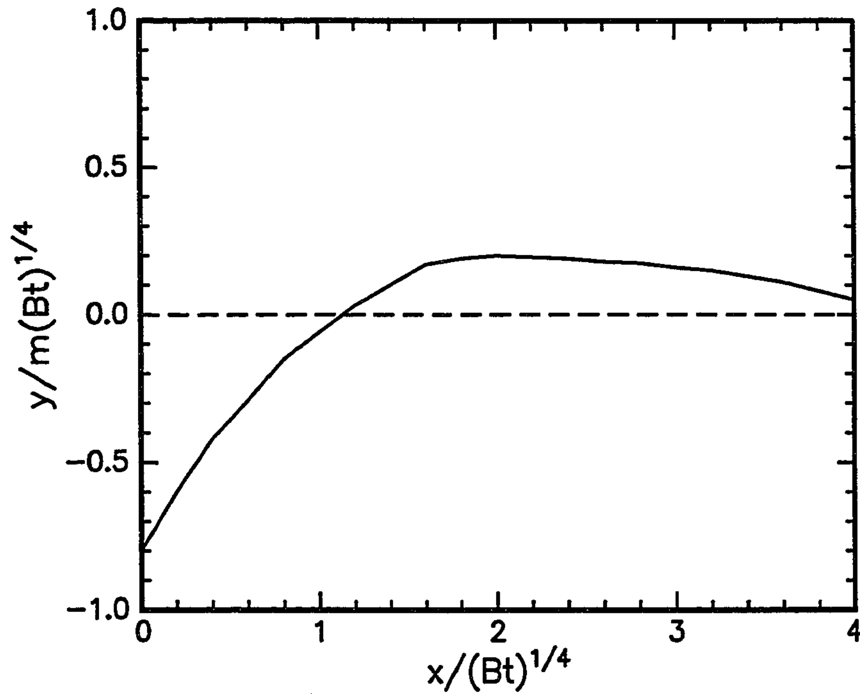


Figure 4.9: A grain boundary groove forming on a surface initially at y = 0. The grain boundary is along the negative y axis.

$$\ell \;=\; 4.6(Bt)^{\frac{1}{4}}, \qquad (4.49)$$

respectively. Hence, a grain boundary intersecting an initially flat surface will form an ever deepening groove of depth d and with maxima to each side separated by ℓ as shown in Figure 4.9.

A moving grain boundary can become stuck at the surface if the magnitude of the angle it makes with the surface normal is less than a critical value θ_c[62]. This can easily be seen if we consider a thermal groove to be represented by a rigid V-notch that remains fixed despite changes in the configuration of the associated boundary. Figure 4.10 shows such a notch with two possible positions for the grain boundary associated with it. A grain boundary lying in the sector bounded by θ_c will be anchored to the notch, since it would be forced to lengthen in order to escape, whereas a boundary with $\theta > \theta_c$ will spontaneously escape from the notch since this will allow it to decrease in length. It is important to note that the effectiveness of the notch as

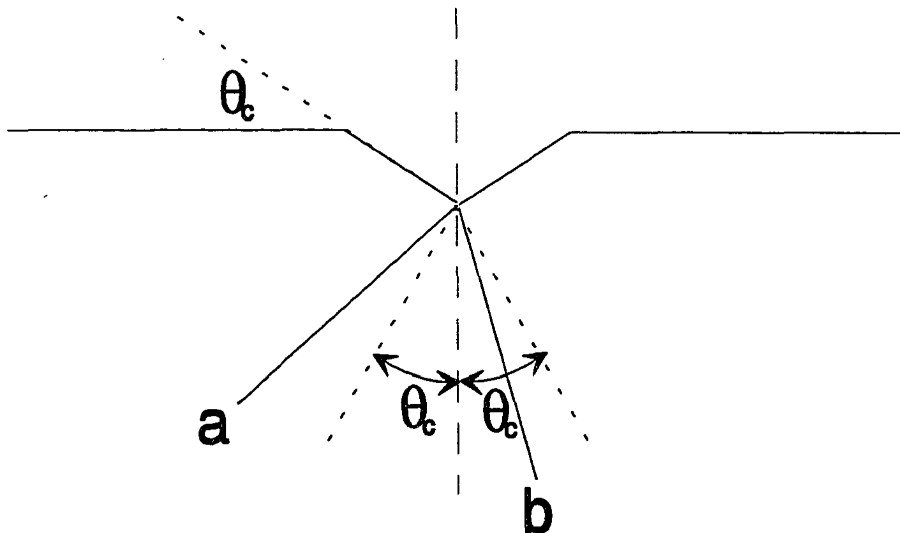


Figure 4.10: An idealized grain boundary groove. In case **a**, the grain boundary may escape the notch, while in case **b** the grain boundary anchored to the notch. The effectiveness of the notch as an anchor depends only upon θ_c, and not on the depth of the groove.

an anchor depends only on θ_c and not on the groove depth. Mullins has rigorously determined that the value of θ_c for a moving grain boundary on an initially flat surface is[62]:

$$\theta_c = \frac{1}{6}\left(\frac{\gamma_{gb}}{\gamma_s}\right). \tag{4.50}$$

A grain boundary in a polycrystalline film could have additional forces placed upon it that could lead to depinning from the groove other than its angle with respect to the surface. During grain growth, grain boundaries migrate to lower the overall energy of the film by shrinking and/or disappearing; the grain boundary can even lengthen if it would result in an overall decrease in the energy.

4.3.2 Grain Boundary Intersecting a General Surface

Grain boundaries can also be explicitly modeled in reflowing high aspect ratio profiles by imposing the following boundary conditions in the region of the grain boundary: (i) that there is a fixed grain boundary groove angle,

$$\vec{x}_+ \times \vec{x}_- = (\vec{x}_+ \cdot \vec{x}_-) tan(\pi - 2\beta) \tag{4.51}$$

with $\sin \beta = \frac{\gamma_{gb}}{2\gamma_s}$, (ii) that there is continuity of the surface current across the grain boundary,

$$\frac{\partial K_+}{\partial s} = \frac{\partial K_-}{\partial s}, \tag{4.52}$$

and (iii) that there is equality of the curvature across the grain boundary,

$$K_+ = K_- \tag{4.53}$$

where + is to the right of the boundary along s and - is to the left. Equation 4.53 is equivalent to assuming the grain boundary has no curvature. These equations, then, are used to solve for the position of the mesh points at, and on each side of, the grain boundary. In finite-element notation,

$$\begin{aligned} (x_1 - x_0)(y_{-1} - y_0) - (x_{-1} - x_0)(y_1 - y_0) &= tan(\pi - 2\beta)[(x_1 - x_0)(x_{-1} - x_0) + \\ &\quad (y_1 - y_0)(y_{-1} - y_0)] && (4.54) \\ K_2 - K_1 &= K_{-1} - K_{-2} && (4.55) \\ K_1 &= K_{-1} && (4.56) \end{aligned}$$

where the point labeled 0 is at the grain boundary-surface intersection, the point labeled 1 is adjacent to the grain boundary on the right, and so on. The form of Equation 4.51 is chosen for its symmetry; this form facilitates the numerical solution of these coupled equations. These equations are solved using a Newton-Raphson method with backtracking[63] and we assume that the points move along the local surface normal. The numerical solution of these equations is described in detail in

Appendix C. Also, note that the curvature K_i depends on the positions of the mesh points $(i - 1)$, i, and $(i + 1)$.

Care must be taken in choosing an appropriate mesh size when adding a grain boundary groove to a finite element simulation. A grain boundary located perpendicular to an otherwise planar surface will form a groove at the grain boundary/surface intersection with depth and width as shown in Equations 4.48 and 4.49, respectively. With a mesh size, Δs, chosen such that $\ell > \Delta s$, it is possible to ensure that the mesh size does not arbitrarily define the groove width.

4.3.3 Some Considerations

The model that has been developed here is a 2-dimensional model. It is useful to step back for a moment and consider our 3-dimensional world. This 2-d model considers the cross-section of a trench with a grain boundary somewhere along the (x,y)-profile. The grain boundary plane is a line in the (x,y) cross-section, but it becomes a plane when the z-direction is added in a 3-d model. We are examining here, then, how a grain boundary that runs along the direction of the trench can affect the kinetics of reflow. In two dimensions, we cannot model the case of a grain boundary whose plane is perpendicular to the trench profile. This case could be very interesting, however. Diffusion along the grain boundary plane into the trench is a likely possibility. The diffusivity along and energy of a grain boundary is extremely dependent upon the type of grain boundary and the misorientation angle of the boundary. Depending on the angle of miscut, the ratio of the grain boundary energy to surface energy for a simple tilt boundary in Cu at 1065°C, γ_{gb}/γ_s, can range from 0 - 0.36, and from 0 - 0.30[64] for a simple twist boundary. The vacancy formation energy for diffusion along a twist grain boundary in Cu varies from 0.14 - 1.42 eV for twist grain boundaries from 8.8 - 43.6°[65], however, the pre-exponentials are not reported. For sputtered, polycrystalline Cu films there should be a large variety of twist grain boundaries along the flat portion of the film, and near the trench there should be a large variety of tilt grain boundaries. The vacancy formation energy varies drastically, but it is clear that

the diffusivity along the grain boundaries can be significant. It is quite possible that the "drapes" of material that we see in Figure 3.1 are due to the high diffusivity along the grain boundaries planes that are perpendicular to the trenches, but the modeling of this phenomena is beyond the scope of this thesis. Also, the grain boundaries in a 3-d film will not always be perpendicular or parallel to the trench. This could possible create additional forces on the grain boundary since the grain boundary may have different forces on it depending upon its initial position along the trench sidewall.

4.3.4 Reflow Including a Single Grain Boundary

The results of adding a grain boundary with γ_{gb} = 300, 600, or 900 ergs/cm^2 at various positions along a profile in a 1 μm by 1 μm trench is shown in Figure 4.11. A typical high angle grain boundary in Cu will have γ_{gb} = 600 ergs/cm^2[33, 66]. Surprisingly, the addition of a single grain boundary can either increase or decrease the reflow time depending upon its position in the initial profile. The addition of a single grain boundary near the bottom of the trench decreases the reflow time by perturbing the hourglass shape that would develop at the bottom of the trench in the absence of a grain boundary. As the position of the single grain boundary is moved up the side of the trench to the inflection point, its effectiveness in perturbing the hourglass region is reduced and the reflow time becomes much closer to that for a continuum surface without a grain boundary. The reflow time tends to be slightly increased for grain boundaries that are placed just outside the top shoulder of the trench. This grain boundary position allows the developing shoulder to become more circular, which slightly increases the stability of the profile, thus slightly increasing the reflow time. Also, note that there is only a very weak dependence on the grain boundary energy, which is not surprising, because the grain boundary's main effect on the evolving profile is to perturb the stability of the lower portion of the hourglass profile.

Figure 4.12 shows the position of the grain boundary for a 1 μm by 1 μm trench with γ_{gb} = 600 ergs/cm^2 as the profile evolves for four of the initial grain boundary

positions shown in Figure 4.11. The simulation assumes that the grain boundary is anchored to the notch; however, this might not be the case. A grain boundary will escape from the groove if it can shorten its length by so doing and, consequently, minimize its energy. Also, the grain boundary will clearly need to be mobile if it is going to keep up with its groove position on the surface–the minimization of the surface energy is, in part, driving the position of the grain boundary.

4.3.5 Reflow Including Multiple Grain Boundaries

The experimentally reflowed films in Chapter 2 and 3 were polycrystalline and therefore included several grain boundaries along any trench profile. Here we will examine the inclusion of two grain boundaries to the reflow model to understand the role and possible interaction of multiple grain boundaries in a reflow process.

Figure 4.13 shows the reflow of four 1 μm by 1 μm trenches with the inclusion of two grain boundaries at different positions in the initial profile. Figure 4.13(a) shows a grain boundary located near the bottom of the initial trench profile and a grain boundary just outside the top shoulder of the trench. These grain boundaries are well separated, and the reflow time is dominated by the grain boundary near the trench bottom. The reflow time is, however, slightly shorter than for this grain boundary alone, since the top grain boundary perturbs the stability of the shoulder as well. The addition of two grain boundaries just outside of the top shoulder of a trench and the addition of two grain boundaries at the top shoulder of a 1 μm by 1 μm trench is shown in Figure 4.13(b) and (c), respectively. The reflow time of (b) is approximately 25% shorter than that in (c), even though the grain boundary positions are relatively close. Clearly, the reflow time is going to depend sensitively on the initial position of the grain boundaries. The left grain boundary in (b) is almost static compared to the right grain boundary; in (c), however, both grain boundaries are moving almost in unison. As the right grain boundary in (b) moves, the regions around it are significantly perturbed by the left grain boundary, whereas in (c), there is only a small relative change between the positions of the left and right boundary.

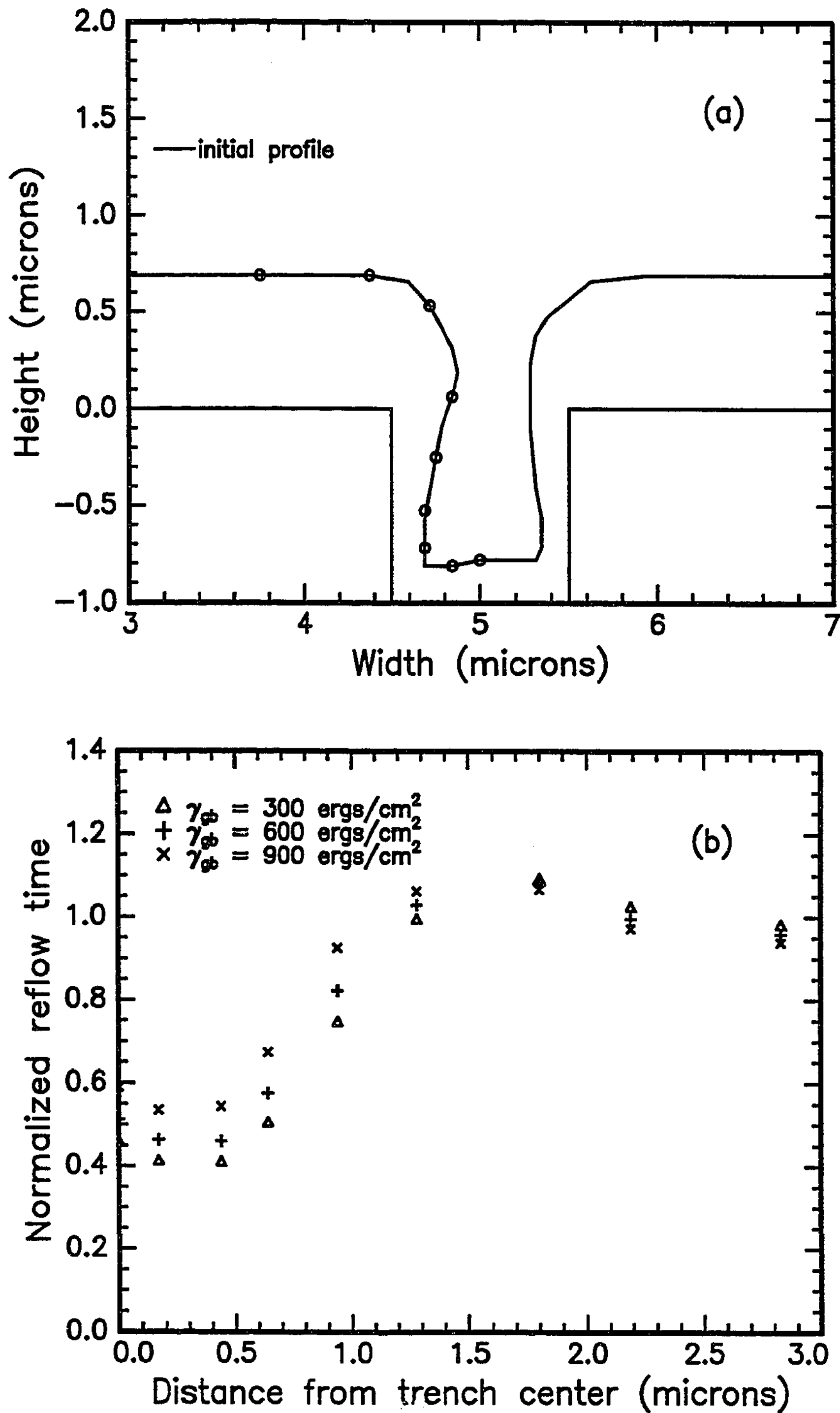
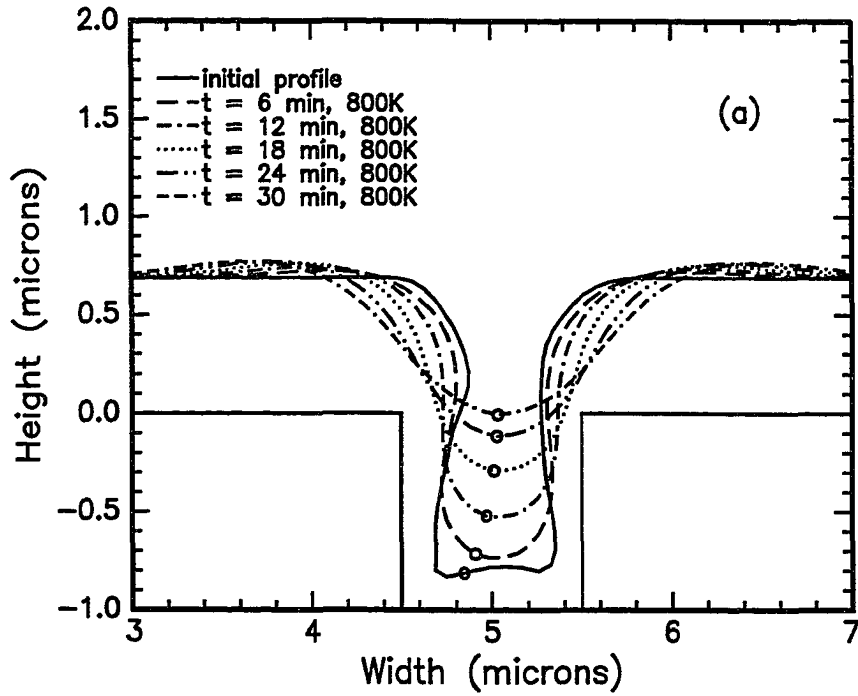
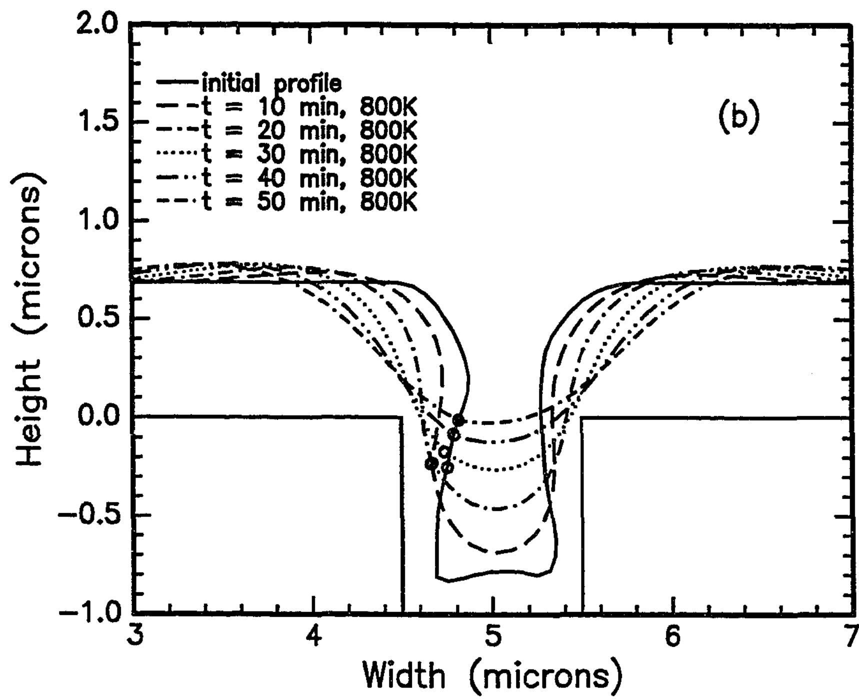


Figure 4.11: (a) Grain boundary position on the initial 1 μm by 1 μm profile and (b) the resulting, normalized reflow times for a grain boundary with γ_{gb} = 300, 600, or 900 ergs/cm^2. The normalized reflow time is the reflow time with inclusion of the grain boundary as indicated in (a) divided by the reflow time for the same profile without a grain boundary.


initial profile
t = 6 min, 800K
t = 12 min, 800K
t = 18 min, 800K
t = 24 min, 800K
t = 30 min, 800K
(a)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7


initial profile
t = 10 min, 800K
t = 20 min, 800K
t = 30 min, 800K
t = 40 min, 800K
t = 50 min, 800K
(b)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7

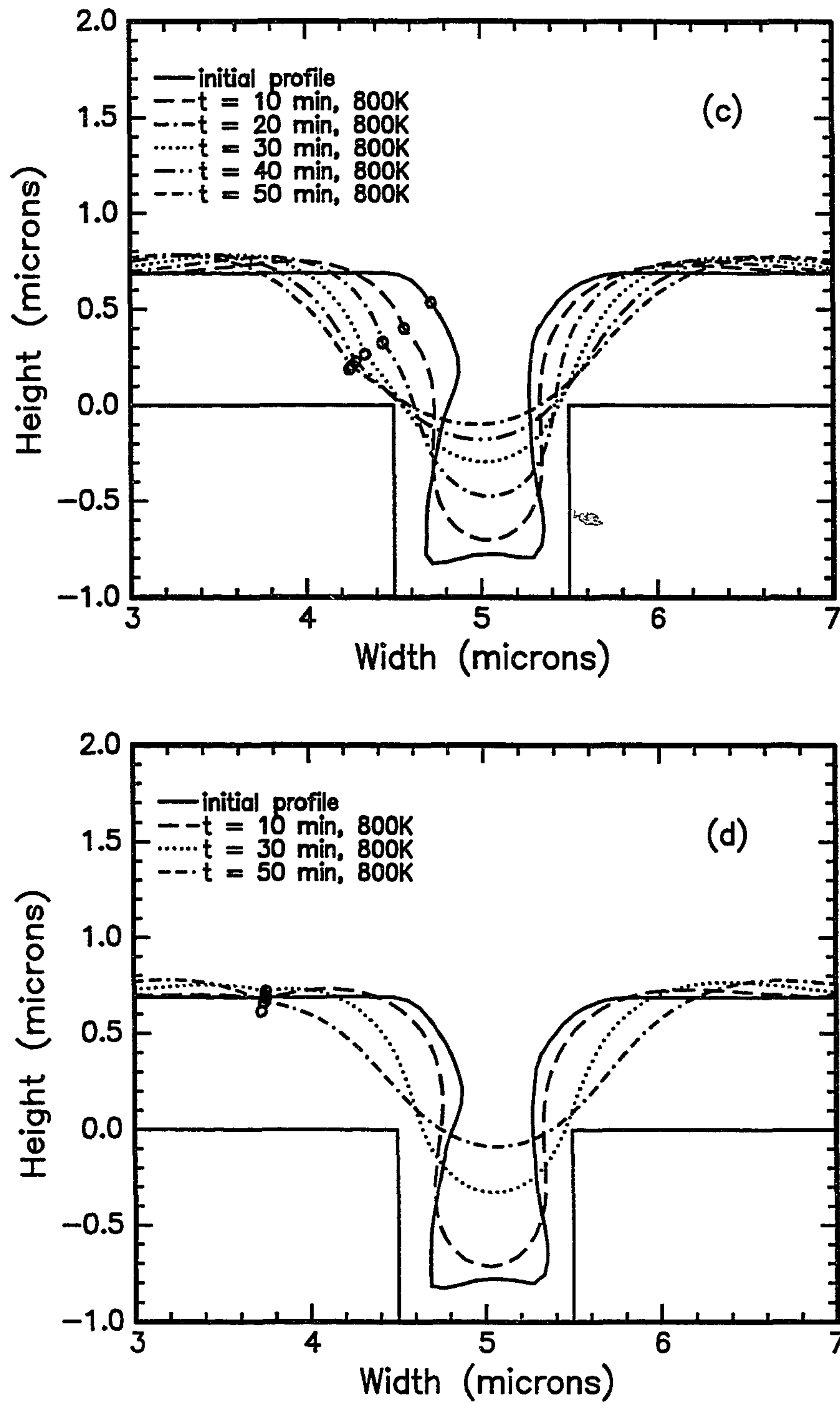


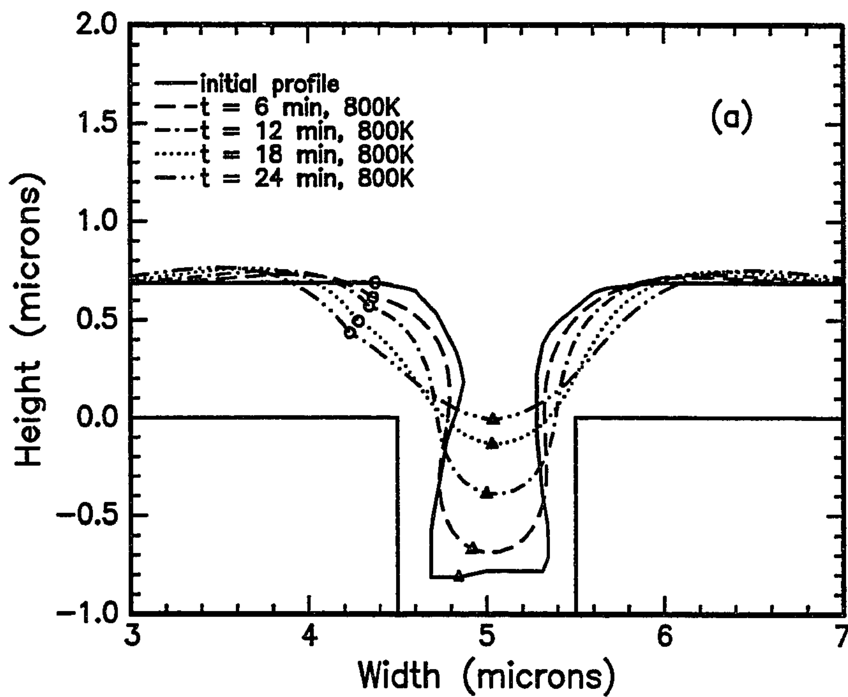
Figure 4.12: Grain boundary position as the profile evolves for a 1 μm by 1 μm trench with $\gamma_{gb} = 600$ ergs/cm^2. The grain boundary position is indicated by a o.

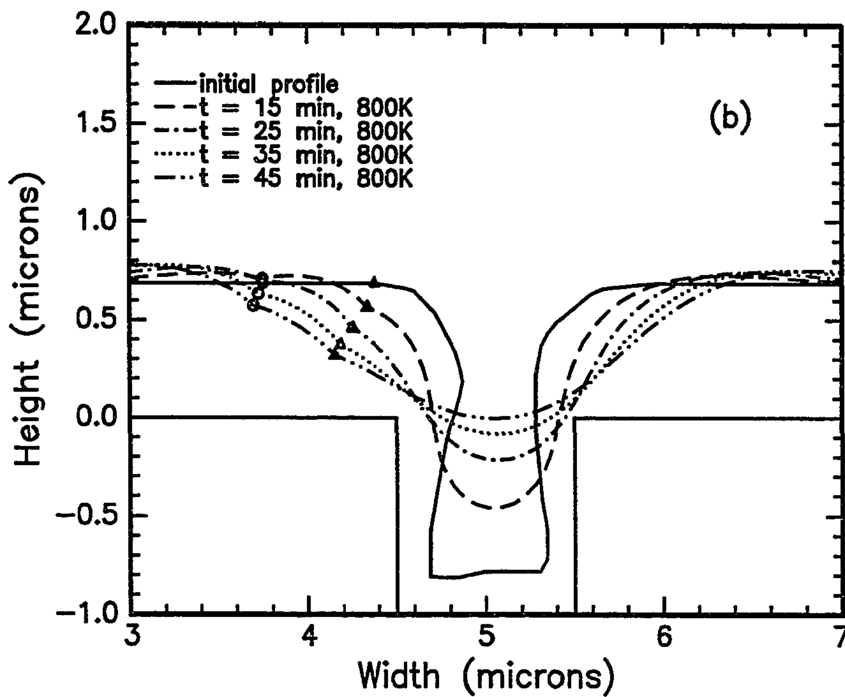
Finally, we consider the motion of two grain boundaries that are in the trench during reflow. The grain boundary positions are quickly approaching one another, and at t = 22 minutes, the grain boundaries are adjacent to each other in the finite element mesh. At this point, we would assume that the grooves would merge into a single grain boundary groove and reflow would proceed as shown in Figure 4.12(a).

4.4 Comparison with Experiment

It is useful at this point to compare these modeling results with the experimental transmission electron microscope (TEM) cross-sections of reflowed Cu in Section 3.2.2 to gain insight into the kinetics and limitations of a post-deposition reflow process and of this model. Clearly, a simple model that includes only surface diffusion, as developed in Section 4.1, is insufficient in predicting many details. It predicts continuously decaying profiles with "shoulders" of material forming on each side of the trench, which is inconsistent with the TEM cross-sections that show some sections of the trenches have reflowed and some sections that have not. This model cannot predict the enhanced stability of the non-reflowed regions. This simple model also predicts the thinning of the sidewall regions during the initial stages of reflow for trenches with aspect ratio greater than 1:1, which seems to be relevant because there has been varied success dependent upon the barrier layer for Al reflow[17, 18]. This model predicts the independence of the reflow in nested trenches, which is a prediction that has not yet been tested. Finally, this model predicts that the filling time, t_{fill}, scales as $w^{1.6}$ where w is the width of the trench, which is a far different scaling law than that obtained from consideration of the small slope case.

The insight gained from the reflow model is greatly enhanced by the addition of an anisotropic surface energy. This model predicts enhanced stability of high aspect ratio structures due to the formation of facets, and for aspect ratios of 2:1 the reflow time is almost doubled with respect to the isotropic surface energy case. The profiles generated by this model are more consistent with the TEM cross-sections, such as the profile shown in Section 3.2.2, which shows flat regions within single grains and


initial profile
t = 6 min, 800K
t = 12 min, 800K
t = 18 min, 800K
t = 24 min, 800K
(a)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7


initial profile
t = 15 min, 800K
t = 25 min, 800K
t = 35 min, 800K
t = 45 min, 800K
(b)
Height (microns)
Width (microns)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
3
4
5
6
7

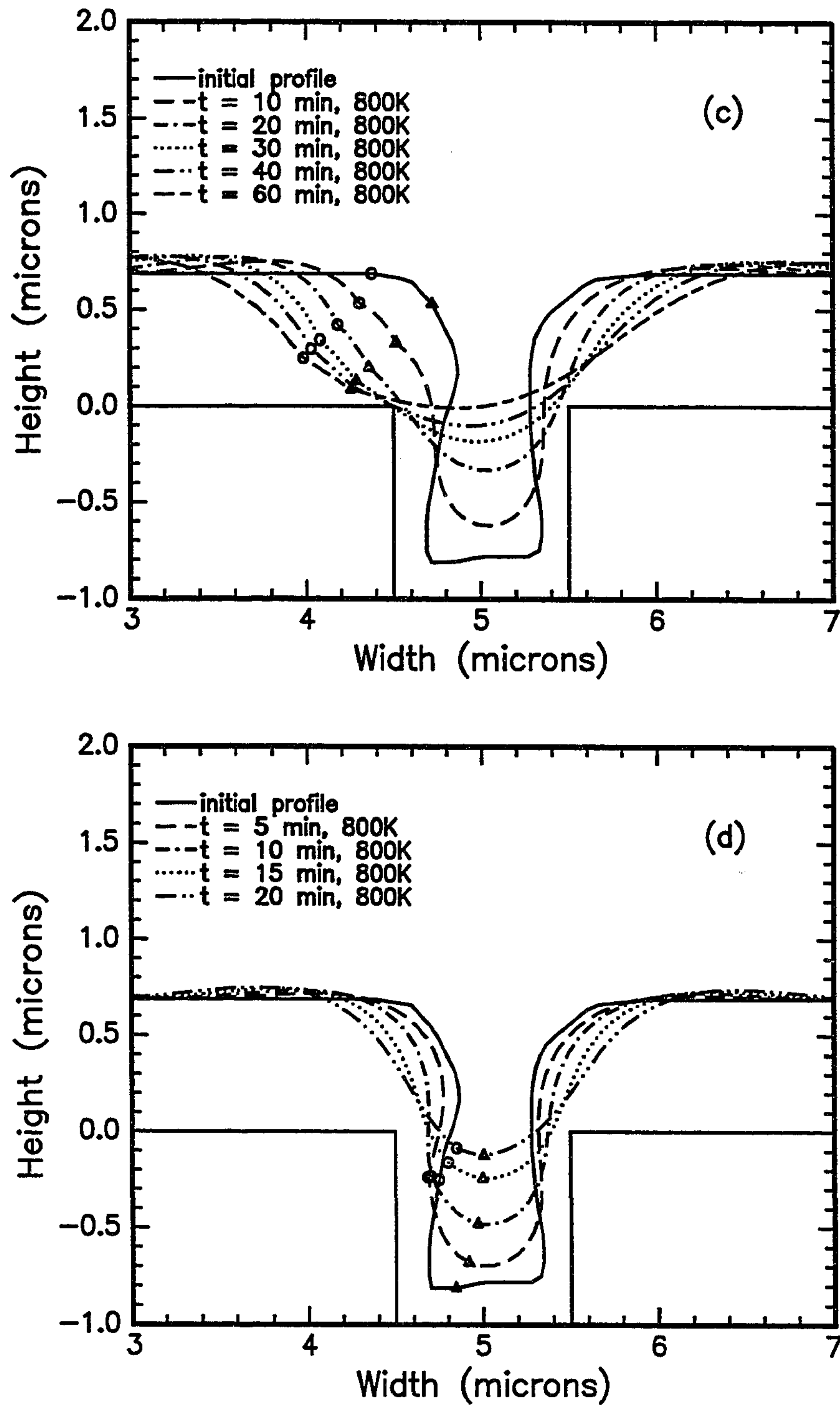


Figure 4.13: Reflowed profiles assuming $\gamma_{gb} = 600$ ergs/cm^2 for both grain boundaries. The symbols indicate the position of the grain boundary on the reflowing profile.

sharp discontinuities in slope at the grain boundaries. Unfortunately, the details of the surface energy plot, $\gamma_s(\theta)$, are not well known at temperatures around 800 K, so it is difficult to make quantitative predictions of the enhanced stability due to the surface energy anisotropy. It is, however, reasonable to conclude that the surface energy anisotropy of Cu is much greater than 2% at temperatures $\leq$ 800 K, and that the driving force for faceting can dominate the kinetics of the reflow. It would also be useful to know the details of the surface energy plot for a Cu surface in a variety of ambients because the presence of ambients can affect the surface energy anisotropy. The inclusion of an ambient during reflow might be able to enhance the ability to reflow Cu by lowering the surface energy anisotropy.

Finally, we have added grain boundaries to the surface diffusion model to see if grain boundaries can limit the kinetics during a reflow process. For the addition of a single grain boundary, the timescales for reflow are extremely dependent upon the position of the grain boundary on the initial profile. The reflow time is decreased by a factor of two when a grain boundary is included near the bottom of a profile, but its effectiveness in speeding reflow is quickly diminished as the grain boundary is moved towards the top edge of the initial profile. This model predicts that an isolated grain boundary does not limit the kinetics, but can enhance them under certain conditions. For the addition of multiple grain boundaries, we have again shown that the timescales are extremely dependent upon the position of the grain boundaries relative to the initial profiles and each other. Grain boundaries initially located near the bottom of the trench tend to converge towards each other as the reflow proceeds. The TEM cross-sections in Section 3.2.2 show only trenches with multiple grain boundaries, but we would predict that these will not limit the kinetics. Rather, what seems to be happening is that the grain boundaries, when *combined* with an anisotropic surface energy can further increase the stability of these structures. The TEM cross-sections have regions containing multiple twin boundaries and faceted surfaces between. These twin boundaries have relatively low energy, so their main effect will be to enhance the stability of the faceted surfaces as shown in the work of Mykura[60, 61].

4.5 Additional Considerations Not Modeled

There are several important microstructural features that have not been included in this reflow model. The two most important of these neglected subjects are, I believe, stress and grain growth during reflow. Both of these subjects depend upon the barrier layer chosen and the details of the deposition; however, it is worthwhile to consider, at least qualitatively, how they might affect reflow.

The prevailing grain growth model predicts a grain growth rate that is a function of three driving forces: minimization of grain boundary energy, surface energy, and elastic energy; the grain growth rate in this model is given by[46, 67]

$$\frac{dr}{dt} = M[\bar{\gamma}_{gb}(\frac{1}{\bar{r}} - \frac{1}{r}) + \frac{2\Delta\gamma_s}{h} + \Delta F_\epsilon] \tag{4.57}$$

where M is the average grain boundary mobility, $\bar{\gamma}_{gb}$ is the average grain-boundary energy, $\bar{r}$ is the average grain radius in the matrix, ΔF_ϵ is the difference in strain energy density between the growing grain and the grains of the matrix, h is the film thickness, and $\Delta\gamma_s = \gamma_s - \bar{\gamma}_s$, where γ_s is the energy of the top and bottom surfaces for the growing grain and $\bar{\gamma}_s$ is the corresponding average energy for the matrix grains. The first term on the right-hand side of Equation 4.57 is the driving force for bulk grain growth when no surface or elastic anisotropy is present; it drives larger grains to consume smaller grains to lower the overall grain boundary area and energy. If there is no orientational dependence of grain size in the as-deposited film, then the driving force for abnormal grain growth will depend only on the competition between the last two terms in Equation 4.57. The second term in Equation 4.57 depends upon the grain's surface energy and can vary with crystallographic orientation. If the second term is the dominant term, then we would expect that the Cu (111) orientation would grow faster than any other orientation since it has the lowest surface energy[47](see Section 4.2).

The third term has been examined extensively in Reference [45] specifically for Cu. The elastic anisotropy of Cu is large and, upon annealing, grains of differing

orientations will have differing strain energy densities. The strain energy density, F, of an individual grain with its (hkl) direction parallel to the surface normal can be expressed as

$$F = \epsilon^2(c_{11} + c_{12} + X - \frac{2(c_{12} - X)^2}{c_{11} + 2X}) \quad (4.58)$$

$$X = (c_{12} + 2c_{44} - c_{11})(h^2k^2 + k^2l^2 + h^2l^2) \quad (4.59)$$

$$1 = h^2 + k^2 + l^2 \quad (4.60)$$

where c_{ij} are the elastic stiffnesses and ϵ is the equal biaxial strain. This expression can be used to calculate ΔF_ϵ only when grains are in a state of pure elastic strain. Using, $c_{11} = 169$ GPa, $c_{12} = 122$ GPa, and $c_{44} = 75.3$ GPa, the Cu (100) grains can be shown to have a significantly lower strain energy density than the surrounding (111) oriented matrix, so that $\Delta F_\epsilon = 146.4\epsilon^2 \times 10^9 J/m^3$[45]. If this term dominants Equation 4.57, it would lead to giant (100) textured grain growth, which has been observed for *part* of the films discussed in Reference [45]. In particular, they have examined Cu films deposited on Ta or W barrier layers at room temperature or at 150°C. To understand why this growth is not observed in all the Cu films, they have added plasticity to their model. Modeling the system as perfectly elastic-plastic gives us:

$$\Delta F_\epsilon = (M_{111} - M_{200})(\Delta\alpha\Delta T)^2 \quad \text{for} \quad 0 \leq \Delta T \leq T^Y_{111} - T_0 \quad (4.61)$$

$$\Delta F_\epsilon = \frac{{\sigma^Y_{111}}^2}{M_{111}} - M_{200}(\Delta\alpha\Delta T)^2 \quad \text{for} \quad T^Y_{111} - T_0 \leq \Delta T \leq T^Y_{100} - T_0 \quad (4.62)$$

$$\Delta F_\epsilon = \frac{{\sigma^Y_{111}}^2}{M_{111}} - \frac{{\sigma^Y_{100}}^2}{M_{100}} \quad \text{for} \quad T^Y_{100} - T_0 \leq \Delta T \quad (4.63)$$

where M_{hkl} is the biaxial modulus for the (hkl) oriented grains, $\Delta\alpha$ is the difference in the thermal expansion coefficients between the substrate and the thin film, $\Delta T = T - T_0$ where T_0 is the temperature at which the film is stress free, σ^Y_{hkl} is the yield strength of the (hkl) orientation, and T^Y_{hkl} is the temperature at which the given orientation will yield. Then, the strain energy difference between the (111) and the

(100) orientation increases with increasing temperature until the (111) orientation yields; as the temperature is increased further, ΔF_ϵ decreases until the (100) orientation yields, and as the temperature is increased further still, ΔF_ϵ becomes a fixed value.

The primary difference between films deposited at room temperature and those deposited at 150°C is the initial grain size. The yield stress of a thin film as a function of grain orientation, film thickness, and grain size is[68]:

$$\sigma_y = \frac{\sin\phi}{b\cos\lambda\cos\phi}\left(\frac{2W_s}{d\sin\phi} + \frac{W_b}{h}\right) \tag{4.64}$$

where ϕ is the angle between the film and slip plane normals, λ is the angle between the film normal and the Burgers vector direction, d is the grain size, h is the film thickness, b is the Burgers vector, and W_s and W_b are the energies per unit length of the dislocation along the sides and bottoms of the grain, respectively. These energies can be approximated by

$$W_s = \frac{b^2\mu_f}{4\pi(1-\nu)}\ln\left(\frac{d}{b}\right) \quad \text{and} \tag{4.65}$$

$$W_b = \frac{b^2}{4\pi(1-\nu)}\frac{2\mu_f\mu_s}{\mu_f+\mu_s}\ln\left(\frac{h}{b}\right) \tag{4.66}$$

where μ_f and μ_s are the shear moduli of the film and substrate, respectively, and ν is Poisson's ratio for the film. For two films with $d_1 < d_2$, then, the films will behave identically until the (111) grains in the d_2 film yield, which fixes the maximum value of ΔF_ϵ. If the strain energy difference is not large enough at the point when the (111) oriented grains yield, the third term will not dominate Equation 4.57 and cannot drive abnormal (100) grain growth in an initially large-grained film. Depending upon the deposition conditions of the Cu film, then, we can expect different driving forces for grain growth to be dominant.

It is also worthwhile to consider different limits in the reflow/grain growth scenario. During all stages of a post-deposition anneal, reflow and grain growth occur simultaneously. Our model of reflow which includes grain boundaries assumes that

the grain boundary is static and only influenced by the motion of the film surface due to capillary-driven surface diffusion. The timescale for filling a 1 μm by 1 μm trench is on the order of 60 minutes for a Cu film at 800 K. During the initial stages of reflow for a 1 μm by 1 μm trench, rapid grain growth occurs because the grain size is not yet comparable with the film thickness where we would expect grain growth to stagnate; however, since the reflow time is long, grain growth will probably stagnate during reflow as the film becomes columnar. The reflow process, then, occurs in both the initial, rapid grain growth limit *and* in the limit where grains have already become columnar and grain growth is relatively slower. The timescale for filling a 0.33 μm by 1 μm trench is on the order of 8 minutes, which is almost an order of magnitude faster than the 1 μm by 1 μm trench. Modeling reflow with the inclusion of grain growth could be a very different process due to the fact that rapid grain growth and reflow are occurring on the same timescales and there will be more interaction/competition between the two processes. These complicated topics deserve much attention, because the experimental results of Cu reflow shown in Chapter 3 seem to indicate dependence on the grain growth mode of the reflowing film.

Chapter 5 Rapid Selective Annealing of Cu Thin Films on Si using Microwaves

A major goal for future integrated circuit fabrication is lowering process temperatures at all stages of fabrication. Conventional thermal annealing processes (such as those used in previous chapters of this thesis) use infrared radiation, heating the film as well as the substrate and potentially creating device reliability problems due to additional and unwanted diffusion of dopants in the device regions. Microwave annealing may provide a novel solution to this problem for a few processes by selective annealing of metallic or other conductive thin films. Selective annealing is possible in the limit where the metal film thickness is approximately equal to the skin depth of the material at microwave frequencies so that the device layer is not actively heated due to the microwaves, and the microwave field is pulsed to maintain a specific temperature profile. Also, this technique is advantageous because the relatively low absorption of silicon compared to the absorption of metals deters substrate heating, and because the thin metallic film shields the rest of the wafer.

We have performed experiments to demonstrate the feasibility of this technique and investigate possible problems. In particular, we have investigated the cavity's quality factor, Q, and the heating characteristics of the silicon as a function of its doping concentration and power input into the microwave cavity. We have also annealed submicron Cu thin films sputtered on patterned SiO_2/Si substrates with microwaves and demonstrated an improvement in trench filling relative to the as-deposited case.

5.1 Thin Film Microwave Annealing Experiments

A sealed, cylindrical, copper microwave cavity, as shown in Figure 5.1, was employed in thin film annealing experiments. The cavity was designed to have a high Q and good

mode separation for the TM010 mode at f = 6.67 GHz. All annealing experiments were performed in this mode. The cavity diameter is 1.350" and the cavity height is 1.855". The unloaded Q of the empty cavity was 13320. Approximately 400 μm thick silicon wafers were cut into 3 mm diameter disks and supported in the center of the cavity with a small, fork-shaped quartz rod. An impurity getter pump was placed into the gas flow line to purify the Ar; during heating, the samples were continuously purged with purified Ar. A fast switch (15 ns) was placed in the power line from the sweep oscillator to the traveling wave tube (TWT) amplifier. Q measurements during heating were performed by switching the microwave cavity power off for a few microseconds and observing the reflected power decay. During sample heating, we tracked the resonant frequency using a control loop that adjusts the input frequency to minimize the reflected power[69]. An optical pyrometer, aligned through a sealed window in the top of the cavity, measured the temperature of the sample.

Cu films were grown in a load lock-equipped ultrahigh vacuum ion-beam sputtering system with base pressure in the low 10^{-9} Torr range as shown in Figure 3.19. 0.8 μm to 1.0 μm thick Cu films were deposited onto patterned SiO_2/Si substrates by Ar ion beam sputtering at a deposition rate of 0.6 Å/sec, estimated from an oscillating-crystal thickness monitor. These substrates were then removed from the vacuum system, cut into 3 mm disks, and annealed in a microwave cavity for 30 seconds with an input power of 1.2 W while being purged with purified Ar.

5.2 Results and Discussion

5.2.1 Silicon Heating Curves

Silicon samples of two different resistivities were examined to measure their Q dependence and temperature dependence as a function of power input into the cavity. Figure 5.2 and Figure 5.3 show the resulting Q measurements and temperature measurements for samples with $\rho = 10.8$ Ω·cm and $\rho = 1000$ Ω·cm, respectively, as a function of power input into the critically-coupled cavity. The pyrometer's minimum

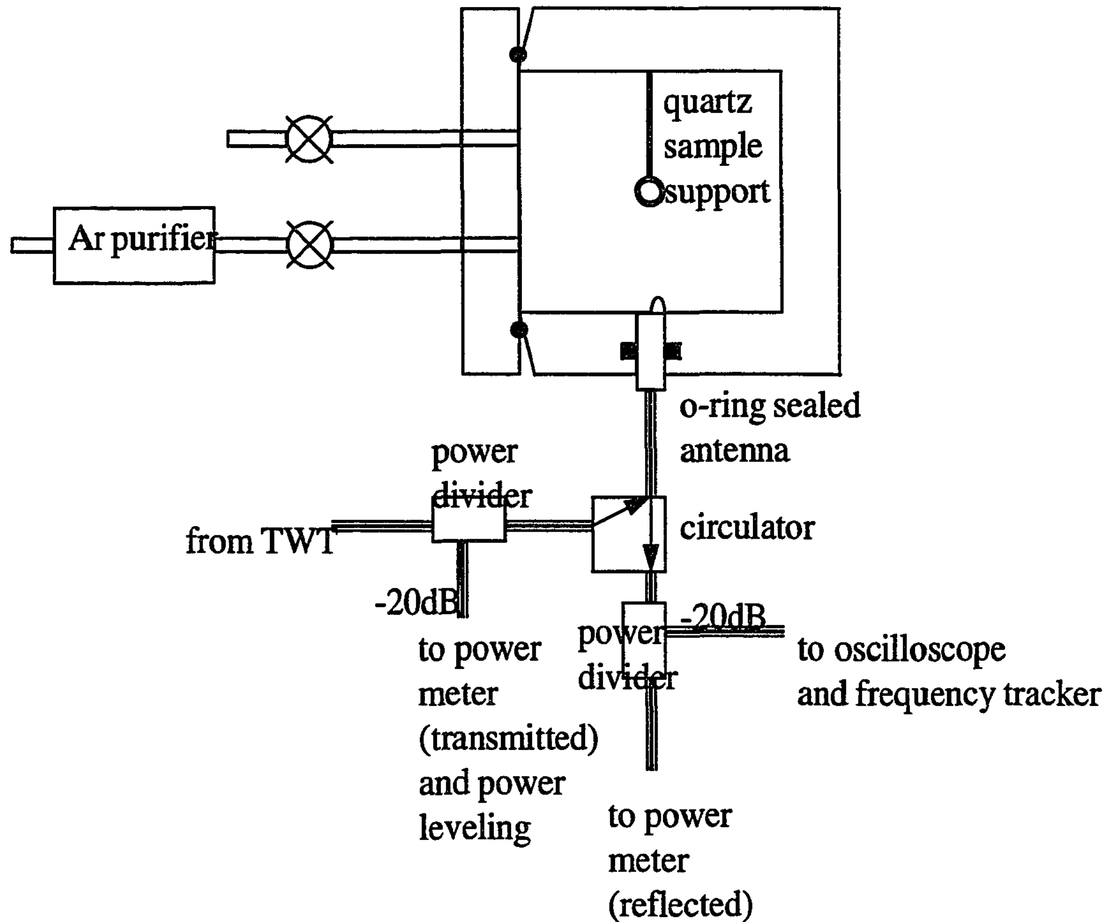


Figure 5.1: Cross-sectional view of the microwave cavity from top. The antenna is sealed with an o-ring and is adjustable to enable critical coupling to the cavity. Above and below the quartz sample support are sealed quartz windows to enable temperature measurements with an optical pyrometer.

resolvable temperature is 300°C; a reading of 300°C in this figure indicates that the sample temperature was at or below this value. For both samples, an emissivity of 0.73 was used. The 10.8 Ω·cm sample has a "normal" heating curve; as the power input into the cavity increases, both the temperature and the Q increase. The sample Q increases at high temperature because the center of the sample is being increasingly shielded as the conductivity of the silicon becomes large. At high temperatures, the rate of temperature increase is quenched because the sample center is no longer being heated due to shielding.

The 1000 Ω·cm sample shows a more anomalous heating behavior. As the power is increased from zero, the Q of the cavity decreases and the temperature correspondingly stays at or below 300°C. In this region, the skin depth is greater than the sample thickness and the absorption of the sample is increasing with increasing temperature; hence the Q decreases. Abruptly at 1.0 W, the sample undergoes thermal runaway, the Q drops from 3000 to 1400, and the temperature simultaneously increases from $\leq$ 300°C to 400°C. At this point, a large impedance adjustment is required (by adjusting the antenna's position in the cavity) so that the cavity remains critically coupled. As the input power is increased further, the sample's quality factor and temperature increase in the same manner as the $\rho = 10.8$ Ω·cm sample. As the power is decreased from 3.0 W towards zero, a similar absorption instability occurs at 0.4 W; at this point, the Q of the sample increases from 700 to 2900 and the temperature remains below 300°C. This hysteretic behavior is shown by the arrows in Figure 5.3.

To understand this anomalous behavior, several coupled, non-linear mechanisms must be considered[70]. Assume we have an isotropic, one-dimensional slab which fills the region $0 < x <$ d with conductivity,

$$\sigma = \sigma_0 f(-1 + \frac{T}{T_0}) \tag{5.1}$$

and that we have the following electric fields in the regions at and around the sample

$$\vec{E} \;=\; E_0[e^{i(kx-\omega t)} + \Gamma e^{-i(kx+\omega t)}]\vec{k} \text{ for } x < 0 \tag{5.2}$$

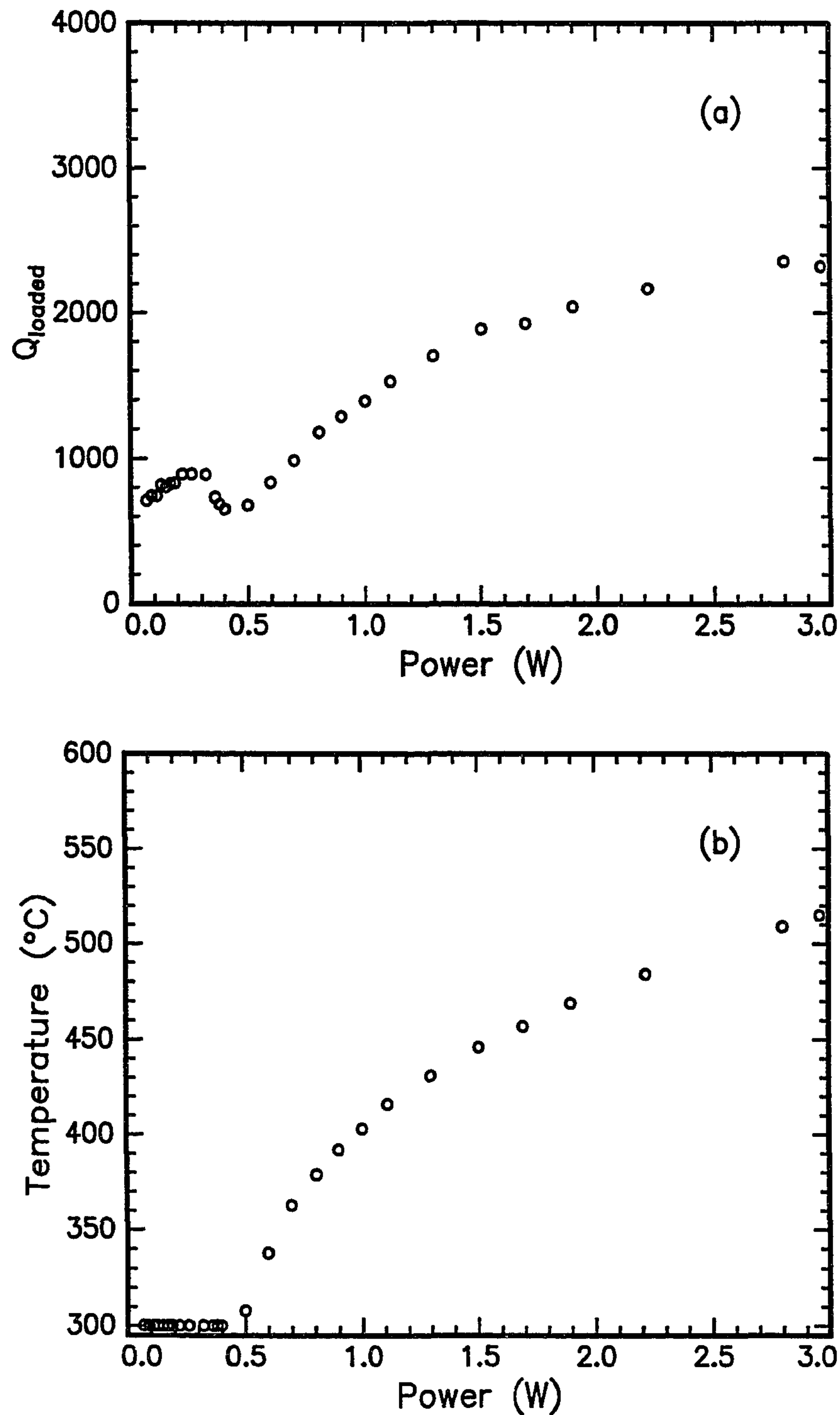


Figure 5.2: (a) Q_{loaded} versus input power and (b) temperature versus input power for Si with resistivity $\rho = 10.8\ \Omega\cdot\text{cm}$. These are values taken after the temperature of the Si had stabilized at the given power level.

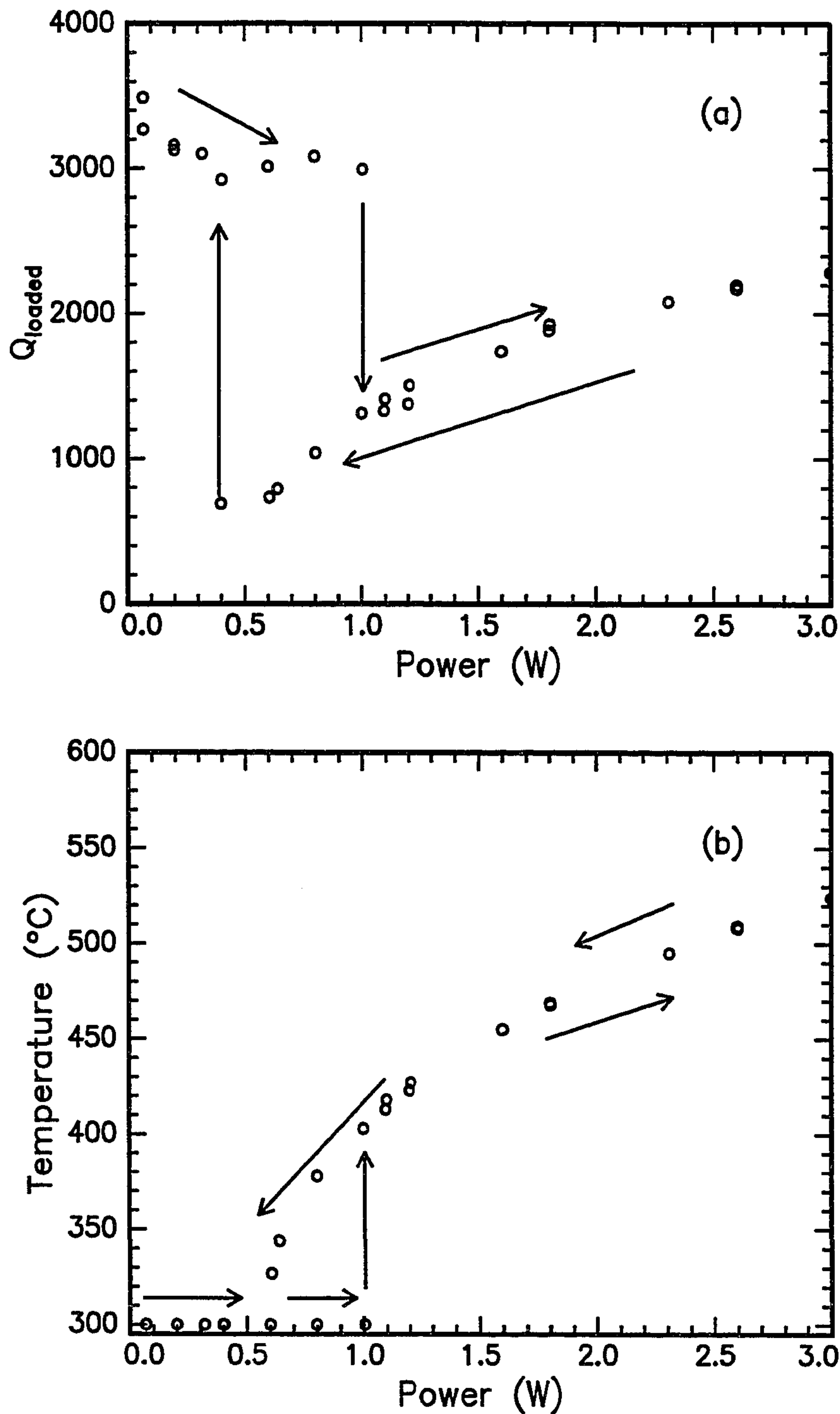


Figure 5.3: (a) Q_{loaded} versus input power and (b) temperature versus input power for Si with resistivity $\rho = 1000$ Ω·cm. The arrows indicate whether the input power was being ramped up or down for a given branch of the heating curve. These are values taken after the temperature of the Si had stabilized at the given power level.

$$\vec{E} = U(x)e^{-\imath\omega t}\vec{k} \text{ for } 0 < x < d \quad (5.3)$$

$$\vec{E} = E_0[\Upsilon e^{\imath(kx-\omega t)}]\vec{k} \text{ for } x > d \quad (5.4)$$

where E_0 is the strength of the incident field, k = $\frac{\omega}{c}$, k$_1$ = $\frac{\omega}{c}\sqrt{\frac{\epsilon_1}{\epsilon_0}}$, Υ is the transmission coefficient, and Γ is the reflection coefficient. The electric field which penetrates the material is given by the real part of Equation 5.3, where $U(x)$ satisfies the equation

$$\frac{d^2U}{dx^2} + k_1^2[1 + \imath\frac{\sigma_0}{\omega\epsilon_1}f(-1 + \frac{T}{T_0})]U = 0 \text{ for } 0 < x < d. \quad (5.5)$$

We must also impose the continuity of the fields and its derivatives at the boundaries of the slab:

$$\frac{dU}{dx} + \imath kU = 2\imath kE_0 \text{ for } x = 0 \text{ and} \quad (5.6)$$

$$\frac{dU}{dx} - \imath kU = 0 \text{ for } x = d. \quad (5.7)$$

Additionally, we must take into account the heating of the sample due to the interaction with the field

$$\rho C_p\frac{\partial T}{\partial t} = K\frac{\partial^2 T}{\partial x^2} + \frac{\sigma_0}{2}f(-1 + \frac{T}{T_0})|U|^2 \quad (5.8)$$

where ρ is the density, C$_p$ is the thermal capacity, and K is the thermal conductivity. Heat balance at the surface requires

$$K\frac{\partial T}{\partial t} = h(T - T_0) + se(T^4 - {T_0}^4) \text{ for } x = 0 \text{ and} \quad (5.9)$$

$$-K\frac{\partial T}{\partial t} = h(T - T_0) + se(T^4 - {T_0}^4) \text{ for } x = d \quad (5.10)$$

where h is the convective heat constant, s is the radiative heat constant, and e is the emissivity of the sample. Finally, we assume that the sample is initially at temperature

$$T(x, 0) = T_0 \text{ for } 0 < x < d. \quad (5.11)$$

The coupled, nonlinear character of this problem is apparent: the electric field propagates through the sample and affects the temperature distribution as shown in Equation 5.8; this changes the electrical conductivity of the material which affects the propagation of the field in the material as shown in Equation 5.5. It is also becoming apparent that this problem is very difficult, but it has been solved by Kriegsmann[70], who solved these coupled equations to first order for samples with exponentially increasing conductivities with temperature and showed that the thickness of the slab and the conductivity model chosen will determine whether or not a sample will undergo thermal runaway.

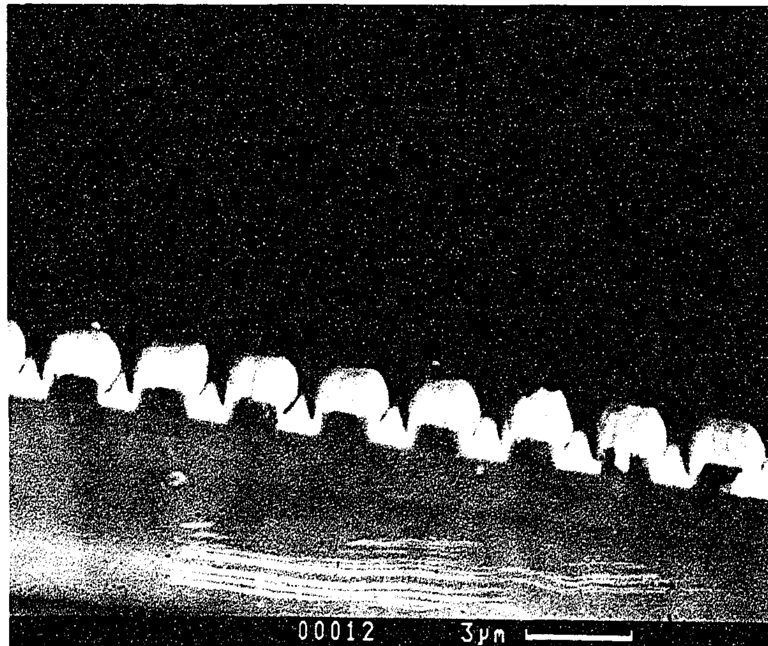
The silicon samples annealed in this study were wafers with thickness d = 400 μm. The skin depth, $\delta = \sqrt{\frac{2}{\mu\omega\sigma}}$, is $\delta_{\rho=10.8\Omega\cdot cm} = 0.2$ cm = 5 d and $\delta_{\rho=1000\Omega\cdot cm} = 2$ cm = 50 d for the samples at room temperature. Kriegsmann's results should certainly be applicable to this case because

$$\sigma \sim e^{\frac{-E_g}{kT}} \tag{5.12}$$

where E_g is the band gap. The low and high resistivity samples have different heating curves because the two samples are in different regimes due to the changing skin depth. As the low resistivity sample is heated, it enters the thick slab limit at a relatively lower temperature than the high resistivity sample, which is in the thin slab limit during much of the annealing cycle. Unfortunately, it is difficult to make quantitative predictions in the regime where the sample makes the transition from the thick slab to the thin slab case.

5.2.2 Cu Reflow

Selective thin film annealing experiments were performed on 0.8 μm to 1.0 μm Cu thin films deposited on a patterned SiO_2/Si sample with grooves 0.8 μm deep and widths ranging from 0.3 μm to 1.0 μm. The resistivity of the Si substrate is approximately 50 Ω·cm. Figure 5.4 shows the patterned sample after Cu deposition in an ion-beam sputtering system. The SEM photos clearly show that the Cu filling



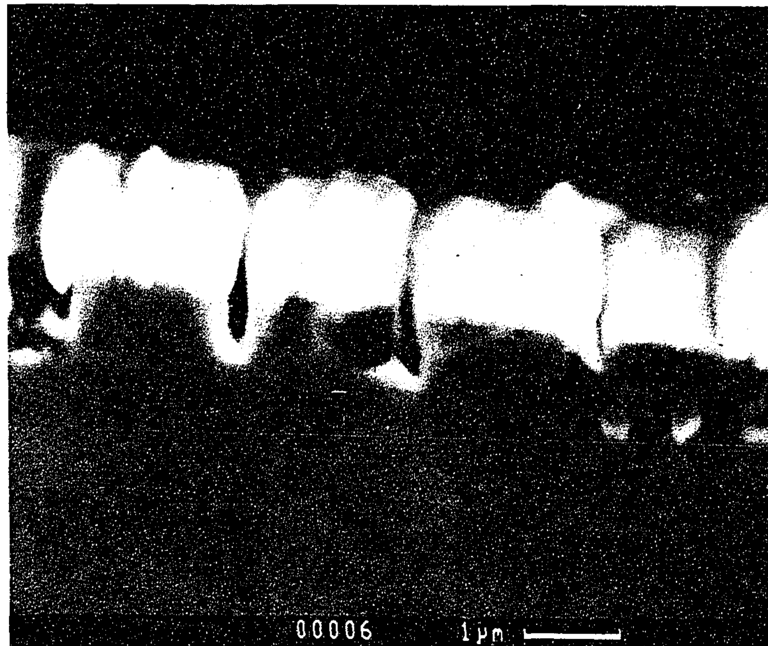
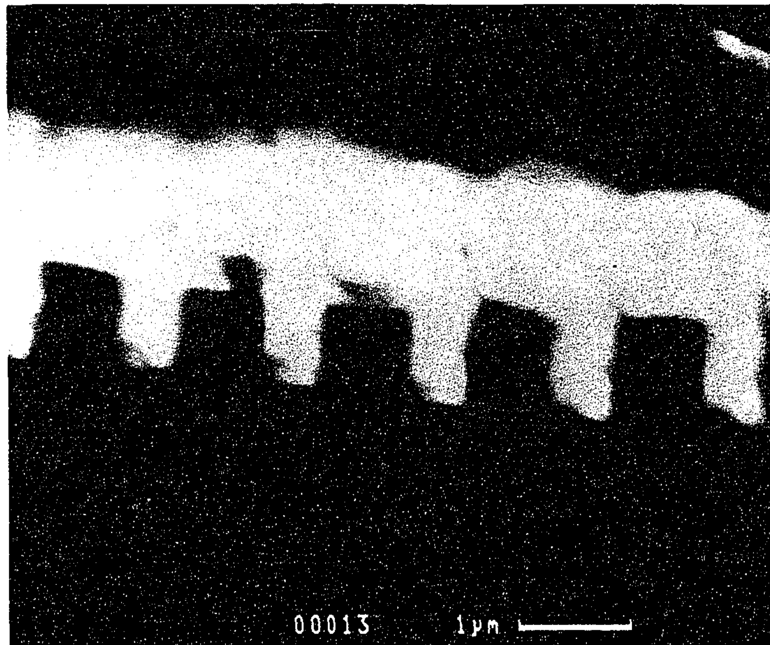


Figure 5.4: Cu films on patterned SiO_2 before microwave annealing. The trench depth is 0.8 μm and the trench widths range from 0.3 μm to 1.2 μm. Note that the trench filling decreases as the aspect ratio increases.



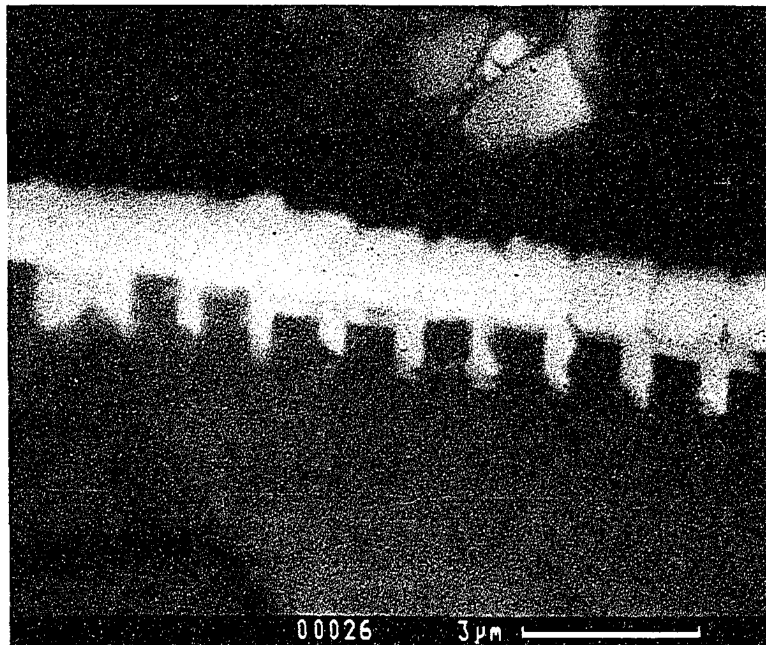


Figure 5.5: Cu films on patterned SiO_2 after microwave annealing. The trenches are now well filled. We believe that the surface cracks are due to the rapid cooling of the samples after annealing.

of these trenches exhibits poor step coverage, which is unacceptable for interconnect metalization. Aspect ratios of approximately 1:1 produce clear gaps in the Cu and, as aspect ratios become greater than 2.5:1, there is very little Cu in the trenches.

Figure 5.5 shows SEM photos of the Cu films after microwave annealing for 30 seconds with an input power of 1.2 W. During the annealing procedure, the temperature of the Cu surface was monitored with the optical pyrometer and also by visual inspection. The pyrometer indicated the Cu film temperature to be less than 300°C (using an emissivity of 0.13) and, in a dark room, the film did not exhibit visible radiation observable by eye, indicating that the temperature was less than 600°C[71]. The annealed Cu films were shiny, indicating a lack of a thick oxide. The annealed films are almost completely planarized. We believe the rapid cooling of the sample caused the surface cracks. Transmission electron micrographs of the Cu films were taken after microwave annealing, and indicate columnar grains and that the average Cu grain size is 0.9 μm. This grain size and grain morphology is consistent with solid-state grain growth, rather than melting and solidification.

These results are rather surprising considering the results of Cu reflow shown in previous chapters, so it is useful to point out some of the similarities and differences between these results and methods and those shown previously. These samples were deposited in an ion-beam sputtering system with base pressure in the mid 10^{-9} Torr range. The deposition rate is much slower in this system than in the magnetron sputtering system described in Section 2.1, and all the precautions that result in ultrapure films (high deposition rate and ultrapure targets) were not taken to such great lengths. Also, the initial profile obtained from an ion-beam sputtering system is very different from that obtained from a magnetron sputtering system. Hence, the purity of these films might be lower than those shown in previous chapters. Also, if we consider the modeling results from Chapter 4, the initial profiles obtained from ion-beam sputtering are more conducive to quick reflow; the low aspect ratio to medium aspect ratio profiles look very much like those obtained from collimated magnetron sputtering (see Figure 4.3(b)). Also, after the Cu films were deposited, they were exposed to air before the post-deposition anneal; the samples shown in

previous sections were annealed *in situ*. Hence, the surface of these exposed films had a copper oxide surface layer; diffusivity data available in the literature[50] indicates that oxygen, at least in trace quantities, seems to enhance the surface diffusion of Cu, although our films most likely have a continuous oxide film. The addition of oxygen to the surface can also greatly modify the surface energy plot, and we have seen in Section 4.2 that this can greatly affect the ability to successfully reflow Cu.

5.3 Summary

Microwave annealing appears to be a promising technique for use in the thermal processing of thin films for integrated circuit applications. As one potential application of this, we investigated reflow of thin Cu films. We were able to fill trenches in patterned SiO_2 substrate with Cu in submicron width grooves with aspect ratios up to 2.5:1. The sometimes unpredictable results of heating bare silicon is a potential problem for this technology that needs to be investigated further. Also, true selective heating can only be achieved when the microwave field is being pulsed with an appropriate duty cycle so that the substrate is not heated due to simple thermal conduction. The experimental set-up described here already includes a fast switch to allow pulsed heating, but additional controls will need to be developed to control the temperature of the sample, and to maximize the temperature difference between the conducting surface film and the device layer beneath.

Chapter 6 Summary and Future Work

We have investigated Cu reflow on refractory metal barrier layers as a possible solution to the nonconformal deposition profiles from a magnetron sputtering system in submicron, high aspect ratio trenches for use in interconnects in integrated circuits.

In Chapter 2, we investigated several techniques to improve the initial Cu coverage obtained from a magnetron sputtering source. We found that a conical (S-gun) target gave sufficient initial Cu coverage for successful reflow in 0.5 μm by 0.5 μm trenches and that the Cu film in these trenches could be reflowed at 500°C in 30 minutes. However, when we attempted this process in 0.5 μm by 1.0 μm trenches, we found that the initial Cu coverage on the sidewalls was extremely thin, and that upon annealing of these films, the Cu film agglomerated along the sidewalls of the trenches. By studying thin Cu films on a flat substrate, we found that 500 Å Cu gave sufficient coverage on *in situ* Mo, Ta, and W underlayers to avoid agglomeration. We also investigated several sputtering techniques to improve the initial sidewall coverage and filling in high aspect ratio trenches. The initial coverage in high aspect ratio trenches showed marked improvement simply by using a planar magnetron sputtering target. Also, we investigated substrate bias as a possible alternative to improved sidewall coverage and filling; the dc substrate bias technique showed regions of good coverage and filling, but the results were extremely non-uniform across the wafer.

In Chapter 3, we studied the mechanisms and kinetics of Cu reflow experimentally. A diffraction experiment was performed on very low aspect ratio trenches (1:10) in an attempt to verify the dominant planarization mechanism and the appropriate kinetic constants. This diffraction experiment demonstrated that the reflow of thin, polycrystalline Cu films cannot be described by simple continuum reflow indicating that the film microstructure is very important. Also, hot-stage TEM experiments were performed to observe the reflow process for 1000 Å Cu films deposited on a W barrier layer. The air-exposed Cu film agglomerated in the trenches of this sample at

temperatures as low as 340°C. The Cu film agglomerated in a spatially inhomogeneous fashion and the Cu beads formed in 5 - 10 seconds on the sample surface. This experiment is interesting since it gave much insight into the dynamics of the reflow process. It also suggested that the agglomerated Cu film could have an orientational relationship between the Cu film and the underlayer, and this could be important to the reflow process.

In Chapter 3, we also examined Cu reflow in 1 μm by 1 μm trenches in ultrahigh vacuum. During the anneal, the samples were monitored in optical diffraction; the diffracted peaks intensities decayed continuously indicating that the Cu film was planarizing during the entire anneal. SEM images taken after the anneal indicate that the reflow was occurring spatially inhomogeneously and the amount of reflow that occurred was dependent upon the underlayer (Ta or W) and the substrate temperature during the Cu deposition (25°C or 150°C). Some sections of the trenches had filled completely and had lengths on the order of 1 μm, and the rest of the sections had filling that was comparable to the as-deposited case. TEM cross-sections of these films indicated that the filled regions were often single grains or grains with boundaries parallel to the trench bottom. Grain boundaries (often twin boundaries) are often tilted approximately 45° with respect to the substrate normal. Also, many of the grains have flat surfaces and sharp discontinuities at the grain boundaries, and we would expect this type of structure to inhibit surface diffusion since a flat surface is a stable solution to the surface diffusion process. The unfilled sections of the trenches often showed groove – anti-groove pairs along that sidewalls with twin boundaries between, and facets sometimes appeared within single grains. These cross-sections strongly indicated that the Cu surface energy was highly anisotropic. Also, many of the unfilled trenches had agglomerated along a single sidewall. XRD taken on the Cu films deposited on Ta indicate that the films have different textures. For the samples deposited at 25°C, the (200) texture increases relative to the (111) texture.

In Chapter 4 we developed a finite-element model to study surface diffusion mediated reflow in high aspect ratio trenches with the inclusion of part of the microstructural features that seem to be important for Cu films. First, we considered reflow of

typical continuum, as-deposited profiles from a magnetron sputtering source. While the diffraction experiments indicate that this simple model is not sufficient for Cu reflow, it may have relevance to Al reflow because the two materials have different surface energy anisotropies and mechanical anisotropies. This model indicates that during the initial stages of reflow, the film along the sidewall of the trenches will be depleted, possibly creating additional agglomeration problems. To this simple model, we have added the complexity of an anisotropic surface energy to the reflow of high aspect ratio profiles. We have found that increasing the anisotropy can dramatically influence the evolution of the profiles. Using an anisotropy of approximately 2% (the measured value for Cu at 1030°C), the reflow time for a 0.5 μm by 1.0 μm trench is almost doubled. If we increase the surface energy anisotropy to approximately 5%, the film forms facets reminiscent of the groove – anti-groove pairs and the film agglomerates along one sidewall. These effects seem extremely relevant when compared with the cross-sectional TEM images in Chapter 3. The surface energy anisotropy will vary depending upon the impurities in the film and the ambient in which the film is being annealed. The dramatic effect that the surface energy anisotropy can have on the reflow process is probably the most exciting result of this work. Finally, we consider reflow with the inclusion of grain boundaries. The position of a single grain boundary near the bottom of the trench decreases the reflow time by almost a factor of 2. As the initial position of the grain boundary is moved up the trench sidewall, the reflow time approaches the value when no grain boundary is included. The inclusion of two grain boundaries in the initial profile also alters the reflow time depending upon their proximity to the trench bottom and to each other. This simulation indicates that grain boundaries can do very little to inhibit reflow in high aspect ratio trenches, however. It should be noted that this model forces the grain boundary to remain anchored to the surface and does not include any additional forces that could be placed on the grain boundary due to grain growth. It would be extremely interesting to include additional forces on the grain boundaries to examine the interaction with the reflow process.

There are several additional elements of these films that we have not modeled,

including film stress and grain growth. The prevalent grain growth model includes terms that drive grains with the minimum surface energy to grow *and* grains with the minimum strain energy density to grow, and for the case of Cu these are the (111) and (100) surfaces, respectively. Our results on Cu/(Ta or W) is consistent the work in References [43] - [45]. They have shown that the dominant term in the grain growth model depends upon the competition between minimizing the surface energy and the strain energy and that the dominant term can change depending upon the barrier layer and the substrate temperature during Cu deposition. Our reflow model does not include any of these complexities, but the experimental results suggest that the dominant mode of grain growth is important. It is also important to note that the reflow behavior during the initial rapid grain growth stage of the film could be very different from the reflow behavior when the film structure has become stagnant, which is the case that our reflow model including grain boundaries simulates. We have seen experimentally that stagnate grain structures with the inclusion of an anisotropic surface energy can have large effects on the ability to reflow by creating stable structures other than the desired one. This suggests that surface defects could enhance the reflow process, because it prevents the grain structure from stagnating; these types of defects could possibly be induced by ion-bombardment during or after deposition of the film.

This research also lends suggestions for future work. The barrier layers in this study were refractory metal barrier layers; their solid solubility with Cu is $< 1\%$. However, we have found that agglomeration on the sidewalls of high aspect ratio trenches can be a difficult problem, and that Al reflow in high aspect ratio trenches (4:1) indicates a dependence upon the barrier layer[17, 18]. Barrier layers which form a thin alloy region at the Cu interface might be helpful in avoiding agglomeration. Also, TEM cross-sections and the model indicate that minimizing the surface energy anisotropy would be prudent. The introductions of ambients into the chamber during the reflow process that can affect/reduce the anisotropy might be advisable. We have successfully reflowed air-exposed samples at temperatures as low as 340°C. It is ironic that we took extreme precautions to maintain an atomically clean surface during

most of the experimental stages of this work because impurities typically reduce the anisotropy of a surface. Of course, the ambient must be chosen so that it doesn't have deleterious side effects (i.e. lowering the surface diffusivity, increasing the electrical resistivity, etc.).

Bibliography

[1] Gopal K. Rao, *Multilevel Interconnect Technology*, McGraw-Hill(1993).

[2] D.B. Fraser, "Metallization" in *VLSI Technology* (1st Edition), ed. by S.M. Sze, McGraw-Hill(1983).

[3] S.P. Murarka, *Metallization Theory and Practice for VLSI and ULSI*, Butterworth-Heinemann(1993).

[4] T.J. Faith, J. Appl. Phys. **52**(7), 4630(1981).

[5] M.B. Anand, T. Matsuno, M. Murota, H. Shibata, M. Kakumu, K. Mori, K. Otsuka, M. Takahashi, N. Kaji, M. Kodera, K. Itoh, R. Aoki, and M. Nagata, Eleventh International VLSI Multilevel Interconnection Conference Proceedings, 15(1994).

[6] M.F. Chisholm, G.A. Dixit, M.K. Jain, R.H. Havemann and T. Weaver, Eleventh International VLSI Multilevel Interconnection Conference Proceedings, 22(1994).

[7] S.P. Murarka, R.J. Gutmann, A.E. Kaloyeros and W.A. Lanford, Thin Solid Films **236**, 257(1993).

[8] Ho-Kyu Kang, Ph.D. Thesis, Stanford University, 1993.

[9] S. Lakshminarayanan, J. Steigerwald, D.T. Price, M. Bourgeois, T.P. Chow, R.J. Gutmann, S.P. Murarka, IEEE Elect. Dev. Lett. **15** (8), 307(1994).

[10] S.P. Murarka, J. Steigerwald, and R.J. Gutmann, MRS Bulletin, pg. 46, June 1993.

[11] J. Tao, N.W. Cheung, C. Hu, H.K. Kang, S.S. Wong, IEEE Elect. Dev. Lett. **13**, 448(1992).

[12] J. Tao, N.W. Cheung, C. Hu, IEEE Elec. Dev. Lett. **14**, 249(1993).

[13] Peter Sigmund, "Sputtering by Ion Bombardment: Theoretical Concepts" in *Sputtering by Particle Bombardment*, ed. by R. Behrisch, Springer-Verlag(1981).

[14] David B. Fraser, "The Sputter and S-Gun Magnetrons" in *Thin Film Processes*, ed. by John L. Vossen and Werner Kern, Academic Press(1978).

[15] H. Windischmann, J. Vac. Sci. Technol. A **9**(4), 2431(1991).

[16] Imran Hashim, Ph.D. Thesis, California Institute of Technology, 1994.

[17] Zheng Xu, Hoa Kieu, Tse-yong Yao and Ivo J. Raaijmakers, Eleventh International VLSI Multilevel Interconnection Conference Proceedings, 158(1994).

[18] N. Ito, Y. Yamada, Y. Murao and D.T.C. Huo, Eleventh International VLSI Multilevel Interconnection Conference Proceedings, 336(1994).

[19] Jason Reid, Ph.D. Thesis, California Institute of Technology, 1995.

[20] *Binary Alloy Phase Diagrams* (2nd Edition), ed. by T. B. Massalski, ASM International(1990).

[21] T. Ono, T. Nakano, and T. Ohta, Appl. Phys. Lett. **64**(12), 1511(1994).

[22] Donald S. Gardner and David B. Fraser, AVS 40th National Symposium, Orlando, FL, November 1993.

[23] SensArray Corporation, Santa Clara, CA custom makes Si wafers with thermocouples embedded in the wafer with a ceramic epoxy for excellent thermal contact between the wafer and thermocouple.

[24] M.L. Gimpl, A.D. McMaster, and N. Fuschillo, J. Appl. Phys. **35**(12), 3572 (1964).

[25] E. Jiran and C.V. Thompson, J. Elect. Mat. **19**(11), 1153(1990).

[26] H. Caswell and Y. Budo, J. Appl. Phys. **35**(3), 644(1964).

[27] L. Bachmann, D.L. Sawyer, and B.M. Siegel, J. Appl. Phys. **36**(1), 304(1966).

[28] D.J. Srolovitz and S.A. Safran, J. Appl. Phys. **60**(1), 247(1986).

[29] D.J. Srolovitz and S.A. Safran, J. Appl. Phys. **60**(1), 255(1986).

[30] S.A. Safran and D.J. Srolovitz, Europhys. Lett. **2**(1), 61(1986).

[31] W.W. Mullins, J. Appl. Phys. **28**(3), 333(1957).

[32] K.T. Miller, F.F. Lange and D.B. Marshall, J. Mater. Res. **5**, 151(1990).

[33] G.L.J. Bailey and H.C. Watkins, Proc. Phys. Soc. B **63**, 350 (1950).

[34] S.M. Rossnagel and J. Hopwood, J. Vac. Sci. Technol. B **12**(1), 449(1994).

[35] S.M. Rossnagel and J. Hopwood, Appl. Phys. Lett. **63**, 3285(1993).

[36] S.M. Rossnagel, D. Mikalsen, H. Kinoshita, and J.J. Cuomo, J. Vac. Sci. Technol. A **9**(2), 261(1991).

[37] C. Nender, I.V. Katardjief, A.M. Barklund, S. Berg and P. Carlsson, Thin Sol. Films **228**, 87(1993).

[38] S. Berg, I.V. Katardjief, C. Nender and P. Carlsson, Thin Sol. Films **241**, 1(1994).

[39] Hellmut Haberland, Martin Mall, Michael Moseler, You Qiang, Thomas Reiners and Yonca Thurmer, J. Vac. Sci. Technol. A **12**(5), 2925(1994).

[40] J.M. Elson, "Low Efficiency Diffraction Grating Theory," Michelson Laboratories, China Lake, 1975.

[41] A.A. Maradudin and D.L. Mills, Phys. Ref. B **11**, 1392(1975).

[42] William W. Mullins, J. Appl. Phys. **30**(1), 77(1959).

[43] Richard P. Vinci and John C. Bravman, Mat. Res. Soc. Symp. Proc **309**, 269(1993).

[44] E.M. Zielinski, R.P. Vinci and J.C. Bravman, Mat. Res. Soc. Symp. Proc. **338**, 307(1994).

[45] E.M. Zielinski, R.P. Vinci, and J.C. Bravman, J. Appl. Phys. **76**(8), 4516(1994).

[46] C.V. Thompson, Annu. Rev. Mater. Sci. **20**, 245(1990).

[47] M. McLean and B. Gale, Philos. Mag. **20**, 1033 (1969).

[48] J.P. Benedict, S.J. Klepeis, W.G. Vandygrift, and Ron Anderson, EMSA Bulletin **19**(2), 74(1989).

[49] G. Neumann and G.M. Neumann, *Surface Self-Diffusion of Metals*, Diffusion Information Center(1972).

[50] *Landolt-Bornstein: Numerical Data and Functional Relationships in Science and Technology* (3rd Edition) **26**, pg. 55, ed. by O. Madelung, Springer-Verlag (1990).

[51] C. Herring, *Physics of Powder Metallurgy*, ed. by Walter E. Kingston, McGraw-Hill(1951).

[52] Andrew Zangwill, *Physics at Surfaces*, Cambridge University Press (1988).

[53] H. Hakkinen, J. Merikoski, M. Manninen, and J. Timonen, Phys. Rev. Lett. **70**(16), 2451(1993).

[54] Edward H. Conrad, Prog. Surf. Sci. **39**, 65(1992).

[55] Jean Lapujoulade, Surf. Sci. Reports **20**, 191(1994).

[56] J. Merikoski, H. Häkkinen, M. Manninen, J. Timonen, K. Kaski, Int. J. Mod. Phys. B **8**(23), 3175(1994).

[57] H.P. Bonzel, N. Freyer, and E. Preuss, Phys. Rev. Lett. **57**(8), 1024(1986).

[58] W.W. Mullins, "Solid Surface Morphologies Governed by Capillarity" in *Metal Surfaces: Structure, Energetics and Kinetics*, Am. Soc. Metals. (1963).

[59] H.P. Bonzel, E. Preuss, B. Steffen, Appl. Phys. A **35**, 1 (1984).

[60] H. Mykura, Acta Met. **5**, 346(1957).

[61] H. Mykura, Acta Met. **9**, 570(1961).

[62] W.W. Mullins, Act. Met. **6**, 414 (1958).

[63] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C* (2nd Edition), Cambridge University Press (1992).

[64] N.A. Gjostein and F.N. Rhines, Acta Met. 7, 319(1959).

[65] Miki Nomura and James B. Adams, J. Mater. Res. **7**(12), 3202(1992).

[66] Richard A. Swalin, *Thermodynamics of Solids*, Second Edition, John Wiley and Sons(1972).

[67] J.E. Sanchez, Jr. and E. Arzt, Scr. Metall. Mater. **27**, 285(1992).

[68] C.V. Thompson, J. Mater. Res. **8**(2), 237(1993).

[69] O. Iny and M. Barmatz, Mat. Res. Soc. Conf. Proc. **347**, 241(1994).

[70] Gregory A. Kriegsmann, J. of Appl. Phys. **71**,4 , 1960 (1992).

[71] *Handbook of Chemistry and Physics*, 43rd Edition, CRC Press, Inc. (1961-1962).

Appendix A Algorithm for Surface Diffusion–Mediated Reflow

The following is a description of the computer program used to model surface diffusion–mediated reflow. It was developed using Borland C++ programming package on a 66 MHz Intel 486DX2.

A.1 Input Parameters

The program is called as shown here:

```
reflow [-T temperature(K)] [-t ending time (min)] [-j time_out (min)]
       [-h trench height (micron)] [-w trench width (micron)]
       [-i input filename] [-o output filename]
       [-s mesh size (micron)]
```

The default values are: T = 800 K, ending time = 60 minutes, output time = 1 minute, trench height = 1 μm, trench width = 1 μm, input filename = "profile", output filename = "reflowed", mesh size = 0.1 μm.

The constants for Cu (diffusion constants, surface energy, etc.) are defined in the program using the `#define` command.

A.2 Subroutines

The following subroutines are used in the reflow program:

curvature Calculates the curvature at each point along the curve.

curvefit Refits a curve so that the curve has a mesh equally spaced in arclength using a cubic spline fit.

err_message Displays an error message if the program is given bad input values.

get_hi Finds the maximum value of the profile.

get_lo Finds the minimum value of the profile.

graphtrench Graphs a trench with dimensions given by the user in the center of the screen for easy comparison with the reflowing profile.

graphprofile Graphs the evolving profile every 100 iterations or whenever the curve has been refit using *curvefit*.

graphtime Graph the current time along the bottom of the screen.

norm Calculates the normal vector to a curve.

pause Forces the program to pause until the user hits return.

printout Prints the time and profile into the output file.

setupgraph Finds and initializes the screen graphics.

tridiag Taken from Numerical Recipes in C, Second Edition, page 51. This subroutine solves the tridiagonal matrix problem–i.e. solve for $\vec{u}$ where M $\vec{u}$ = $\vec{r}$.

A.3 Code

A.3.1 reflow.h

```
/*Define vector as a new type so that I can use dynamic memory.*/
/*The default of 'class' is 'private'.  'class' is equivalent to 'struct'.*/
class vector
   {
   public:
   float x,y;
   };

/*Define types of all subroutines*/
double curvature(vector p1, vector p2, vector p3, float ds);
void tridiag(float *a, float *b, float *c, float *r, float *u, int nsteps);
int curvefit(vector **p, int nsteps, float ds);
vector norm(vector p1,vector p2,vector p3);
void err_message();
```

```
void setupgraph();
void graphtrench(int xmax, int ymax, float h, float w);
void graphprofile(vector *p, int nstep, int xmax, int ymax, int color);
void pause();
float get_hi(vector *p, int nsteps);
float get_lo(vector *p, int nsteps);
void printout(vector *p, int nsteps, float t, FILE *output);
void graphtime(int xmax, int ymax, float t_sec);
```

A.3.2 reflow.cpp

```
//The purpose of this program is to reflow an arbitrary surface by surface
//diffusion. The constants in the system here are for a Cu(111) surface. It is
//written in a combination of ANSI C with a few C++ commands sprinkled in where
//it was needed to make the programming easy. To obtain help with it, type
//'reflow help' from the DOS prompt and it will prompt you for the inputs
//and show you the defaults values of the parameters(unless I forgot to update
//the error_message). This code was developed with the Borland C++ (version 4)
//package; if you're trying to take it somewhere else, you will also need to
//grab the 'reflow.h' file since it defines a new type and all the subroutines
//for this program. Ruth Brain, 7/94.
//Major updates were made to this code in Sept and Oct 1994. I added the whole
//section to refit the curve as the distance between the points changes. I also
//corrected the normal subroutine. RAB 10/94

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
#include "reflow.h"       //The library that I have defined to define my subroutines as well as
                          //set-up 'vector' as a new type.

#define gamma 1800        //surface energy (dyne/cm)
#define Ds0 0.07          //surface diffusion pre-exponential (cm^2/s)
#define Qs  0.82          //surface diffusion activation energy (eV)
#define omega 1.2e-23     //atomic volume (cm^3/atom)
#define nu 1.43e15        //atoms per unit surface area (atoms/cm^2)
#define k 8.62e-5         //Boltzmann's constant (eV/K)
#define e 2.71828         //(d/dx)(e^x) = e^x

int main(int argc,char *argv[])
{
long i;
int j;
int p;
int n;
int flag_time;
char ch;
float time_end;           //The program will run through time_end seconds.
float time_out;           //Every time_out seconds, the current profile will be saved to the output file.
float dist;               //distance between nearest neighbor points--used to see when curve should be refit
int nsteps_s;             //number of steps in arc; program uses a variable mesh(fixed ds)
                          //so nsteps_s will change
double ds;                //element of arc length (in microns)
float dt;                 //element of time (in sec)
float time;               //Tells how much time has passed (in sec) during reflow--
                          //necessary with variable time step

//The next 4 definitions allow those variables to use dynamic memory rather than static
//(since I only have 64k of static)
double *curv=new double[400];     /*curvature at (x(s),y(s))*/
```

```
vector *pos=new vector[400];      /*(x,y) of surface (in microns)*/
vector *newpos=new vector[400];   /*revised (x,y) of surface*/
vector *temp;                     /*temporary pointer used to allow *pos to point at *newpos*/

float Ds;               /*surface diffusivity (in cm^2/s)*/
float T;                /*temperature (in K)*/
float B;                /*coef. for surface diffusion diff. eq.(in micron^4/s)*/
float rate;             /*rate of movement of normal(in microns/s)*/
vector normal;          /*normal to curve*/
float height;           /*trench height (in microns)*/
float width;            /*trench width (in microns)*/

char *filename_i;       /*the filename containing the initial profile's (s,x,y)*/
char *filename_o;       /*                            output                    */

int maxx;               /*number of pixels in graphics window along x axis*/
int maxy;               /*ditto along y axis*/

FILE *input;            /*Input file that contains the initial profile to reflow.*/
FILE *output;           /*Output file that contains time and profile data.*/

/*Put in default values in case they are not entered*/
T = 800.;          //in Kelvin
time_end = 3600.; //in seconds
ds = 0.1;          //in microns
height = 1.0;      //in microns
width = 1.0;       //in microns
filename_i = "profile";
filename_o = "reflowed";

/*Read in command-line arguments.*/
j = 1;
while (j < argc)
   {
   if(argv[j][0] == '-')
      {
      switch(argv[j][1])
         {
         case 'i':   /*input filename*/
            filename_i = argv[j+1];
            j += 2;
            break;
         case 'o':   /*output filename*/
            filename_o = argv[j+1];
            j += 2;
            break;
         case 't':   /*time til end of run: time_end (sec) but entered in minutes*/
            time_end = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'j':   /*spacing between saving data to filename_o; entered in minutes*/
            time_out = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'T':   /*temperature (K)*/
            T = atof(argv[j+1]);
            j += 2;
            break;
         case 's':   /*mesh size (microns)*/
            ds = atof(argv[j+1]);
            j += 2;
            break;
         case 'h':   /*trench height (micron)*/
            height = atof(argv[j+1]);
            j += 2;
            break;
         case 'w':   /*trench width (micron)*/
```

```
            width = atof(argv[j+1]);
            j += 2;
            break;
         default:
            err_message();
            goto finish;
         }
      }
   else
      {
      err_message();
      goto finish;
      }
   }

/*Calculate constants for run*/
/*Ds = surface diffusivity (cm^2/s)*/
Ds = Ds0 * exp(-Qs/(k*T));
/*The constants in front of the expression make the units for B microns^4/s*/
B = (1e16)*(6.242e11)*(Ds * gamma * omega * omega * nu)/(k * T);

/*Read (pos[j].x, pos[j].y) = (x,y) at t=0 from 'profile'.*/
input = fopen(filename_i,"r");
j = 0;
while(fscanf(input,"%f %f\n",&(pos[j].x), &(pos[j].y)) != EOF)
   j++;
nsteps_s = j;

/*Set-up graphics output*/
setupgraph();
/*Finds the maximum number of pixels for the chosen graphics window*/
maxx = getmaxx();
maxy = getmaxy();

/*Graph the trench profile and initial profile.*/
graphprofile(pos,nsteps_s,maxx,maxy,0);
graphtrench(maxx, maxy, height, width);
printf("Initial profile\n");
pause();

//Fit input curve so that it has an equal mesh(i.e. equal ds between points).
nsteps_s = curvefit(&pos, nsteps_s, ds);

//Graph initial profile after refitting.
graphprofile(pos,nsteps_s,maxx,maxy,1);
graphtrench(maxx, maxy, height, width);
printf("Initial profile after refit\n");
pause();

//Open the output file up and put in the time and the initial profile.
output = fopen(filename_o,"w");
printout(pos,nsteps_s,0.,output);

/*Main loop*/
dt = (ds*ds*ds*ds)/(4.*B); //in sec; tmaxstable = (ds^4)/(2.*B)
time = 0.;        //in sec
p = 1;
flag_time = 0;
i=0;
n=1;
while(time < time_end)
   {
   i++;
   //Calculate curvature.
   curv[0] = 0.;
   curv[nsteps_s - 1] = 0.;
   for(j=1;j<nsteps_s-1;j++)
```

```
   curv[j] = curvature(pos[j-1],pos[j],pos[j+1],ds);

//Calculate newpositions after step dt.
tryagain:   //If timestep too large, try this loop again.
newpos[0] = pos[0];
newpos[nsteps_s - 1] = pos[nsteps_s - 1];
for(j=1;j<nsteps_s-1;j++)
   {
   rate = B * (curv[j+1] - 2.*curv[j] + curv[j-1])/(ds*ds);
   normal = norm(pos[j-1],pos[j],pos[j+1]);
   newpos[j].x = pos[j].x + rate*dt*normal.x;
   newpos[j].y = pos[j].y + rate*dt*normal.y;
   if((rate*dt) > 0.1*ds) //Timesteps too large--decrease timestep.
      {
      dt *= 0.95;
      goto tryagain;//If timestep is too large, don't want to finish out loop--go back to beginning.
      }
   }

//Find elapsed time and try to increase timestep every 200 iterations.
time += dt;
if((i%200) == 0)
   dt *= 1.05;

//Put the time and the current profile in the output file every 600 seconds.
if(time >= time_out*p)
   {
   printout(pos,nsteps_s,time,output);
   p++;
   }

//Print times when trench is filled by 10%, 20%, .....
if(get_lo(pos,nsteps_s) >= (-height + .1*n))
   {
   printf("Trench is %i %% filled at t = %5.1f min\n",10*n,time/60.);
   pause();
   n++;
   }
//Is the trench filled?  If so, print the time and ask if you want to continue run.
if((get_lo(pos,nsteps_s) >= 0.)&&(flag_time == 0))
   {
   printf("Trench filled at t = %5.1f min\n",time/60.);
   printf("Do you want to continue run?(y/n)\n");
   ch = '\0';
   while((ch != 'y')&&(ch!= 'n')) ch = getchar();
   if(ch == 'n')
      goto finish;
   printout(pos,nsteps_s,time,output);
   flag_time = 1;
   }

//Swap newpos and pos.
temp = pos;
pos = newpos;
newpos = temp;

//Refit curve if any spacing has moved by >10%.  Ignore spacing to last point since
//it is probably off anyway,i.e. go only to nsteps_s-2.
for(j=0;j<nsteps_s-2;j++)
   {
   dist = sqrt((pos[j].x - pos[j+1].x)*(pos[j].x - pos[j+1].x) +
               (pos[j].y - pos[j+1].y)*(pos[j].y - pos[j+1].y));
   if((fabs(ds - dist)) > (0.1*ds))
      {
      nsteps_s = curvefit(&pos, nsteps_s, ds);
      graphprofile(pos,nsteps_s,maxx,maxy,1);
      graphtrench(maxx, maxy, height, width);
```

```
            }
        }

    //Graph profile every 100 iterations.
    if(i%100 == 0)
        {
        graphprofile(pos,nsteps_s,maxx,maxy,0);
        graphtrench(maxx, maxy, height, width);
        graphtime(maxx,maxy,time);
        }
    }

finish:
printout(pos,nsteps_s, time, output);
fclose(input);
fclose(output);
closegraph();
return(0);
}
/*****************************************************************************/
void printout(vector *p, int nsteps, float t, FILE *output)
{
int i;

fprintf(output,"t=%f\n",t);
for(i=0;i<nsteps;i++)
    fprintf(output,"%7.4f %7.4f\n",p[i].x, p[i].y);

return;
}
/*****************************************************************************/
/*Find maximum point in y and return its value.*/
float get_hi(vector *p, int nsteps)
{
float max;
int j;

max = p[0].y;
for(j=1;j<nsteps;j++)
    {
    if(p[j].y > max)
        max = p[j].y;
    }

return(max);
}
/*****************************************************************************/
/*Find minimum point in y and return its position in array.*/
float get_lo(vector *p, int nsteps)
{
float min;
int j;

min = p[0].y;
for(j=1;j<nsteps;j++)
    {
    if(p[j].y < min)
        min = p[j].y;
    }

return(min);
}
/*****************************************************************************/
/*Calculate arclength*/
float arclength(vector *pos, int nsteps)
{
int j;
```

```
float s,ds,ds2;

s = 0.;
for(j=1;j<nsteps;j++)
   {
   ds2 = (pos[j].x - pos[j-1].x)*(pos[j].x - pos[j-1].x) +
         (pos[j].y - pos[j-1].y)*(pos[j].y - pos[j-1].y);
   ds = sqrt(ds2);
   s += ds;
   }

return(s);
}
/****************************************************************************/
//Fit curve so that ds is constant, function returns the number of steps in new fit.
int curvefit(vector **p, int nsteps, float ds)
{
int i,j;
int nsteps_new;          //The number of steps that will be in the new curve.
float s;
float *a,*b,*c,*rx,*ry;
float h,h1,h2;
float skip;
int flag;
float coef1, coef2, coef3, coef4;
float *d2xds2, *d2yds2;
vector *temp;
vector lastpoint;

//Calculate how many sections to divide curve into.  This code occasionally has problems
//fitting near the end of the profile; I'm not sure why this is, perhaps it is cumulative
//errors in adding up the distances of/to each section.  The neurotic points can be gotten
//rid of by making nsteps_new a little smaller than it really should be--nsteps_new should
//be (int)(s/ds) + 1.  The distance to the last point is always wrong anyway, so usually I
//just make nsteps_new small and hence make the last section too big--the curvature is zero
//at the end anyway, so it shouldn't really introduce any errors.
//Also note below that I have the ability to change nsteps_new to a smaller # if
//i becomes greater than (nsteps-2).  This is needed due to accumulated
//errors in the calculation of the arclength between the new and the old curves.
//Modified 3/95--RAB.
s = arclength(*p,nsteps);
nsteps_new = (int)(s/ds);
//Also, keep the last point from old position array so that curve
//doesn't get lost with many iterations.
nsteps_new++; //Increment nsteps_new for array definition, then decrement it until the end of routine.
lastpoint = (*p)[nsteps-1];

if(nsteps_new > 400)
   {
   printf("nsteps_new = %i     Memory problems!!!!!!!\n",nsteps_new);
   pause();
   }

//Allocate space for these arrays.  I am defining in this fashion since I am
//sometimes short of memory.  They must be deleted at the end of routine.
a = new float[nsteps];
b = new float[nsteps];
c = new float[nsteps];
rx = new float[nsteps];
ry = new float[nsteps];
temp = new vector[nsteps_new];
d2xds2 = new float[nsteps];
d2yds2 = new float[nsteps];
nsteps_new--;  //Temporarily decrement.  Have to figure the last point out separately
               //at the end of program separately.

//Solve set of equations for d2xds2 and d2yds2 found using cubic spline type set-up.
```

```
//Set endpoints to have d2xds2 = d2yds2 = 0.
a[0] = 0.;
b[0]  = 1.;
c[0] = 0.;
rx[0] = 0.;
ry[0] = 0.;
a[nsteps-1] = 0.;
b[nsteps-1] = 1.;
c[nsteps-1] = 0.;
rx[nsteps-1] = 0.;
ry[nsteps-1] = 0.;
for(j=1;j<(nsteps-1);j++)
   {
   h1 = sqrt(((*p)[j].x - (*p)[j-1].x)*((*p)[j].x - (*p)[j-1].x) +
          ((*p)[j].y - (*p)[j-1].y)*((*p)[j].y - (*p)[j-1].y));
   h2 = sqrt(((*p)[j+1].x - (*p)[j].x)*((*p)[j+1].x - (*p)[j].x) +
          ((*p)[j+1].y - (*p)[j].y)*((*p)[j+1].y - (*p)[j].y));

   a[j] = h1/6.;
   b[j] = (h1 + h2)/3.;
   c[j] = h2/6.;

   rx[j] = (*p)[j+1].x/h2 - (*p)[j].x/h2 - (*p)[j].x/h1 + (*p)[j-1].x/h1;
   ry[j] = (*p)[j+1].y/h2 - (*p)[j].y/h2 - (*p)[j].y/h1 + (*p)[j-1].y/h1;
   }
tridiag(a,b,c,rx,d2xds2,nsteps);
tridiag(a,b,c,ry,d2yds2,nsteps);

//Fit curve using cubic spline calculation.
temp[0] = (*p)[0];
i = 0;
skip = 0.;
flag = 0;
for(j=1;j<nsteps_new;j++)
   {
   //Find the correct interval to fit between and calculate what s is for that interval.
   tryagain:
   if(arclength((*p),i+2) >= (arclength(temp,j) + ds))
      {
      if(flag == 0)  //We are in the same interval as for last value of temp.
         s = ds + sqrt(((*p)[i].x - temp[j-1].x)*((*p)[i].x - temp[j-1].x) +
                    ((*p)[i].y - temp[j-1].y)*((*p)[i].y - temp[j-1].y));
      else           //We are in a different interval from last time.
         s = ds - skip;
      //Calculate the total spacing between the two points that we are going to fit.
      h = sqrt(((*p)[i].x - (*p)[i+1].x)*((*p)[i].x - (*p)[i+1].x) +
            ((*p)[i].y - (*p)[i+1].y)*((*p)[i].y - (*p)[i+1].y));
      //Reset the variables for the next iteration.
      skip = 0.;
      flag = 0;
      }
   else   //Incorrect interval.
      {
      //Calculate the value of s that is still left in the interval we are skipping out of.
      if(flag == 0)
         skip += sqrt(((*p)[i+1].x - temp[j-1].x)*((*p)[i+1].x - temp[j-1].x) +
                  ((*p)[i+1].y - temp[j-1].y)*((*p)[i+1].y - temp[j-1].y));
      else
         skip +=sqrt(((*p)[i+1].x - (*p)[i].x)*((*p)[i+1].x - (*p)[i].x) +
                 ((*p)[i+1].y - (*p)[i].y)*((*p)[i+1].y - (*p)[i].y));
      //Go to the next interval.
      i++;
      if(i > (nsteps-2))   //Fixes problem.  See note at beginning of subroutine.
         {
         nsteps_new = j;
         goto quit;
         }
```

```
      //Flag = 1 indicates that we have switched intervals.
      flag = 1;
      //With the new value of i, go back and check to see if we are now in the correct interval.
      goto tryagain;
      }

   //Do the fit.
   coef1 = 1. - (s/h);
   coef2 = 1. - coef1;
   coef3 = (coef1*coef1*coef1 - coef1)*h*h/6.;
   coef4 = (coef2*coef2*coef2 - coef2)*h*h/6.;
   temp[j].x = coef1*(*p)[i].x + coef2*(*p)[i+1].x + coef3*d2xds2[i] + coef4*d2xds2[i+1];
   temp[j].y = coef1*(*p)[i].y + coef2*(*p)[i+1].y + coef3*d2yds2[i] + coef4*d2yds2[i+1];
   }

quit:
//Assign values in temp to (*p) and assign a value to the last point.
for(i=0;i<nsteps_new;i++)
   (*p)[i] = temp[i];
if((fabs((*p)[nsteps_new-1].x - lastpoint.x) < .01)&&(fabs((*p)[nsteps_new-1].y - lastpoint.y)< .01))
   {
   }
else
   {
   nsteps_new++;
   (*p)[nsteps_new-1] = lastpoint;
   }

//Delete all the memory that I allocated.
delete[]a;
delete[]b;
delete[]c;
delete[]rx;
delete[]ry;
delete[]temp;
delete[]d2xds2;
delete[]d2yds2;

return(nsteps_new);
}
/*****************************************************************************/
//Taken from Numerical Recipes in C, Second Edition, page 51.  This solves the
//tridiagonal problem matrix problem for u where Mu = r with a below the
//diagonal, b the diagonal and c above the diagonal.
void tridiag(float *a ,float  *b, float *c, float *r, float *u, int nsteps)
{
int j;
float bet;
float *gam=new float[nsteps];

//printf("Made it to tridiag\n");
//pause();
bet = b[0];
if(bet == 0.)
   printf("Error 1 in tridiag\n");
u[0] = r[0]/bet;

for(j=1;j<nsteps;j++)
   {
   gam[j] = c[j-1]/bet;
   bet = b[j] - a[j]*gam[j];
   if(bet == 0.)
      printf("Error 2 in tridiag: j = %i\n",j);
   u[j] = (r[j] - a[j]*u[j-1])/bet;
   }

for(j=(nsteps-2);j>=0;j--)
```

```
   u[j] -= gam[j+1]*u[j+1];

delete[]gam;
return;
}
/******************************************************************************/
/*curvature at p2 where p1 is point to left, p3 to right*/
double curvature(vector p1, vector p2, vector p3, float ds)
{
float dxds, dyds;
float dx2ds2, dy2ds2;
float sum_sq;
float denom;
float curve;

dxds = (p3.x - p1.x)/(2.*ds);
dyds = (p3.y - p1.y)/(2.*ds);
dx2ds2 = (p3.x - 2.*p2.x + p1.x)/(ds*ds);
dy2ds2 = (p3.y - 2.*p2.y + p1.y)/(ds*ds);
sum_sq = (dxds)*(dxds) + (dyds)*(dyds);
denom = sum_sq * sqrt(sum_sq);
/*The sign of the curvature corresponds to the convention Mullins uses.*/
curve = -(dxds * dy2ds2 - dyds * dx2ds2)/denom;

return(curve);
}
/******************************************************************************/
/*Calculate the normal to the curve at p2 where p1 is to the left, p3 is to the right.*/
/*ycomp = 1; This was deleted to increase speed and all values of ycomp were replaced by 1*/
vector norm(vector p1,vector p2,vector p3)
{
float xcomp,ycomp;
float m;
float normalize;
vector result;

if(p3.x == p1.x) /*Line is vertical*/
   {
   if(p1.y > p3.y)
      {
      result.x = 1.;
      result.y = 0.;
      }
   else
      {
      result.x = -1.;
      result.y = 0.;
      }
   }
else /*Line is not vertical*/
   {
   m = (p3.y - p1.y)/(p3.x - p1.x);
   if(((m > 0.)&&(p1.y > p3.y)) || ((m < 0.)&&(p1.y < p3.y)))
      {
      xcomp = m;
      ycomp = -1.;
      }
   else
      {
      xcomp = -m;
      ycomp = 1.;
      }
   normalize = sqrt(xcomp*xcomp + ycomp*ycomp);
   result.x = xcomp/normalize;
   result.y = ycomp/normalize;
   }
```

```
return(result);
}
/****************************************************************************/
void err_message()
{
printf("usage: reflow [-T temperature(K)] [-t time_end (min)]\n");
printf("\t[-h trench height(micron)] [-w trench width (micron)] [-i filename_i]\n");
printf("\t[-o filename_o] [-s mesh size(micron)] [-j time_out (min)]\n");
printf("Default values:\n");
printf("\t T = 800K\n");
printf("\t time_end = 60 min\n");
printf("\t time_out = 1 min\n");
printf("\t ds = 0.1 microns\n");
printf("\t h = 1 micron\n");
printf("\t w = 1 micron\n");
printf("\t filename_i = profile\n");
printf("\t filename_o = reflowed\n");

return;
}
/****************************************************************************/
void setupgraph()
{
/*request autodetection*/
int gdriver = DETECT, gmode, errorcode;

/*initialize graphics mode*/
initgraph(&gdriver, &gmode, "");

/*read result of initialization*/
errorcode = graphresult();

if (errorcode != grOk) /*an error occured*/
   {
   printf("graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);
   }
return;
}
/****************************************************************************/
/*Make the screen width 2 hours*/
void graphtime(int xmax, int ymax, float t_sec)
{
float t_hr;
float scr_w;   //Screen width in hours.

scr_w = 2.;
setcolor(LIGHTRED);

outtextxy(0,ymax-textheight("0"),"0");
outtextxy((int)(1.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(2.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(3.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(4.*xmax/8.),ymax-textheight("0"),"1");
outtextxy((int)(5.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(6.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(7.*xmax/8.),ymax-textheight("0"),"|");
outtextxy(xmax-textwidth("2"),ymax-textheight("0"),"2");

t_hr = t_sec/3600.;
moveto(0,ymax-2.*textheight("0"));
lineto((int)(xmax*t_hr/scr_w),ymax-2.*textheight("0"));

return;
}
```

```
/*****************************************************************************/
/*Make width of graphics window correspond to 10 microns(adj. with scr_w)*/
/*h = trench height in cm, w = trench width in cm*/
/*For graphics window, lower left at (0,0); upper right at (xmax, ymax)*/
void graphtrench(int xmax, int ymax, float h, float w)
{
float onemicron, scr_w;
float x,y;

setcolor(WHITE);
scr_w = 10.;   /*scale for width of graphics window(in microns)*/
onemicron = (float)(xmax/scr_w);   /*convert 1 micron to pixel units*/

moveto(0,(int)(ymax/2));          /*start halfway up screen*/

x = onemicron*(scr_w/2. - w/2.);
y = ymax/2.;
lineto((int)x,(int)y);

y = ymax/2. + onemicron*h;
lineto((int)x,(int)y);

x = onemicron*(scr_w/2. + w/2);
lineto((int)x,(int)y);

y = (int)(ymax/2);
lineto((int)x,(int)y);

lineto(xmax, (int)(ymax/2));

return;
}
/*****************************************************************************/
void graphprofile(vector *p, int nstep, int xmax, int ymax, int color)
{
float onemicron, scr_w;
float x,y;
int j;

scr_w = 10.;
onemicron = (float)(xmax/scr_w);
if(color == 0)
   setcolor(YELLOW);
else if (color == 1)
   setcolor(LIGHTGREEN);
else
   setcolor(RED);
clearviewport();
moveto(0,(int)(ymax/2. - onemicron*p[0].y));
for(j=1; j<nstep; j++)
   {
   x = onemicron*(p[j].x - p[0].x);
   y = ymax/2. - onemicron*p[j].y;
   lineto((int)x,(int)y);
   }

return;
}
/*****************************************************************************/
void pause()
{
char ch;

ch = '\0';
printf("Press return to continue.....\n");
while(ch != '\n') ch = getchar();
return;
```

```
}
/**********************************************************************************/
```

Appendix B Algorithm for Reflow with an Anisotropic Surface Energy

The following algorithm was used to model surface diffusion–mediated reflow assuming that the surface energy is anisotropic.

B.1 Subroutines

The following subroutine was used in addition to those described in Section A.2:

surfenergyterm Calculates the normalized surface energy term $\frac{1}{\gamma_s(0)}[\gamma_s(\theta) + \frac{\partial^2\gamma_s}{\partial\theta^2}]$.

B.2 Code

B.2.1 reflowan.h

```
/*Define vector as a new type so that I can use dynamic memory.*/
/*The default of 'class' is 'private'.  'class' is equivalent to 'struct'.*/
class vector
   {
   public:
   float x,y;
   };

/*Define types of all subroutines*/
void surfenergyterm(vector *normal, int nsteps, float result[400]);
void curvature(vector *p, float ds, int nsteps, float curve[400]);
void tridiag(float *a, float *b, float *c, float *r, float *u, int nsteps);
int curvefit(vector **p, int nsteps, float ds);
void norm(vector *p, int nsteps, vector **normal);
void err_message();
void setupgraph();
void graphtrench(int xmax, int ymax, float h, float w);
void graphprofile(vector *p, int nstep, int xmax, int ymax, int color);
void pause();
float get_hi(vector *p, int nsteps);
```

```
float get_lo(vector *p, int nsteps);
void printout(vector *p, int nsteps, float t, FILE *output);
void graphtime(int xmax, int ymax, float t_sec);
```

B.2.2 reflowan.cpp

```
//The purpose of this program is to reflow an arbitrary surface by surface
//diffusion.  The constants in the system here are for a Cu(111) surface.  It is
//written in a combination of ANSI C with a few C++ commands sprinkled in where
//it was needed to make the programming easy. To obtain help with it, type
//'reflow help' from the DOS prompt and it will prompt you for the inputs
//and show you the defaults values of the parameters(unless I forgot to update
//the error_message).  This code was developed with the Borland C++ (version 4)
//package; if you're trying to take it somewhere else, you will also need to
//grab the 'reflow.h' file since it defines a new type and all the subroutines
//for this program.  Ruth Brain, 7/94.
//Major updates were made to this code in Sept and Oct 1994.  I added the whole
//section to refit the curve as the distance between the points changes.  I also
//corrected the normal subroutine.  RAB 10/94
//Addition of anisotropic surface energy terms to equation of motion.  See
//comments by surfenergyterm subroutine.  RAB 1/95
//Also, modified some of the subroutines in hopes of speeding up the program.
//None of the fundamental elements of the subroutines have been changed, I only
//set it up so that it would calculate all of the elemnts for the curve at one
//time.  RAB 1/95

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
#include "reflowan.h"        //The library that I have defined to define my subroutines as well as
                             //set-up 'vector' as a new type.

#define gamma100 1800        //surface energy for (100) planes(dyne/cm)
#define Ds0 0.07             //surface diffusion pre-exponential (cm^2/s)
#define Qs 0.82              //surface diffusion activation energy (eV)
#define omega 1.2e-23        //atomic volume (cm^3/atom)
#define nu 1.43e15           //atoms per unit surface area (atoms/cm^2)
#define k 8.62e-5            //Boltzmann's constant (eV/K)
#define e 2.71828            //(d/dx)(e^x) = e^x
#define pi 3.14159

int main(int argc,char *argv[])
{
long i;
int j;
int p;
int flag_time;
char ch;
float time_end;       //The program will run through time_end seconds.
float time_out;       //Every time_out seconds, the current profile will be saved to the output file.
float dist;           //distance between nearest neighbor points--used to see when curve should be refit
int nsteps_s;         //number of steps in arc; program uses a variable mesh(fixed ds) so
                      //nsteps_s will change
float ds;             //element of arc length (in microns)
float dt;             //element of time (in sec)
float time;           //Tells how much time has passed (in sec) during reflow--necessary
                      //with variable time step
float Ds;             //surface diffusivity (in cm^2/s)
float T;              //temperature (in K)
float B;              //coef. for surface diffusion diff. eq.(in micron^4/s)
float rate;           //rate of movement of normal(in microns/s)
float height;         //trench height (in microns)
float width;          //trench width (in microns)
```

```
//The next few definitions allow those variables to use dynamic memory rather than
//static(since I only have 64k of static)
float *curv=new float[400];        /*curvature at (x(s),y(s))*/
vector *pos=new vector[400];       /*(x,y) of surface (in microns)*/
vector *newpos=new vector[400];    /*revised (x,y) of surface*/
vector *temp;                      /*temporary pointer used to allow *pos to point at *newpos*/
float *surfenergy=new float[400];//Contains the surface energy and torque term normalized by
                                   //gamma due to anisotropies in the surface energy (dimensionless)
vector *normal=new vector[400];    /*normal to curve*/

char *filename_i;                  /*the filename containing the initial profile's (s,x,y)*/
char *filename_o;                  /*                               output               */

int maxx;                          /*number of pixels in graphics window along x axis*/
int maxy;                          /*ditto along y axis*/

FILE *input;                       /*Input file that contains the initial profile to reflow.*/
FILE *output;                      /*Output file that contains time and profile data.*/

/*Put in default values in case they are not entered*/
T = 800.;          //in Kelvin
time_end = 3600.; //in seconds
time_out = 600;   //in seconds
ds = 0.1;          //in microns
height = 1.0;      //in microns
width = 1.0;       //in microns
filename_i = "profile";
filename_o = "reflowed";

/*Read in command-line arguments.*/
j = 1;
while (j < argc)
   {
   if(argv[j][0] == '-')
      {
      switch(argv[j][1])
         {
         case 'i':   /*input filename*/
            filename_i = argv[j+1];
            j += 2;
            break;
         case 'o':   /*output filename*/
            filename_o = argv[j+1];
            j += 2;
            break;
         case 't':   /*time til end of run: time_end (sec) but entered in minutes*/
            time_end = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'j':   /*spacing between saving data to filename_o; entered in minutes*/
            time_out = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'T':   /*temperature (K)*/
            T = atof(argv[j+1]);
            j += 2;
            break;
         case 's':   /*mesh size (microns)*/
            ds = atof(argv[j+1]);
            j += 2;
            break;
         case 'h':   /*trench height (micron)*/
            height = atof(argv[j+1]);
            j += 2;
            break;
         case 'w':   /*trench width (micron)*/
            width = atof(argv[j+1]);
```

```
          j += 2;
          break;
        default:
          err_message();
          goto finish;
        }
      }
    else
      {
      err_message();
      goto finish;
      }
  }

/*Calculate diffusion constant for run and rate constant, B (in microns^4/sec)*/
/*Ds = surface diffusivity (cm^2/s)*/
Ds = Ds0 * exp(-Qs/(k*T));
B = (1e16)*(6.242e11)*(Ds * omega * omega * gamma100 * nu)/(k * T);

/*Read (pos[j].x, pos[j].y) = (x,y) at t=0 from 'profile'.*/
input = fopen(filename_i,"r");
j = 0;
while(fscanf(input,"%f %f\n",&(pos[j].x), &(pos[j].y)) != EOF)
   j++;
nsteps_s = j;

/*Set-up graphics output*/
setupgraph();
/*Finds the maximum number of pixels for the chosen graphics window*/
maxx = getmaxx();
maxy = getmaxy();

/*Graph the trench profile and initial profile.*/
graphprofile(pos,nsteps_s,maxx,maxy,0);
graphtrench(maxx, maxy, height, width);
printf("Initial profile\n");
pause();

//Fit input curve so that it has an equal mesh(i.e. equal ds between points).
nsteps_s = curvefit(&pos, nsteps_s, ds);

//Graph initial profile after refitting.
graphprofile(pos,nsteps_s,maxx,maxy,1);
graphtrench(maxx, maxy, height, width);
printf("Initial profile after refit\n");
pause();

//Open the output file up and put in the time and the initial profile.
output = fopen(filename_o,"w");
printout(pos,nsteps_s,0.,output);

/*Main loop*/
dt = 1.;   //in sec
time = 0.; //in sec
p = 1;
flag_time = 0;
i=0;
while(time < time_end)
   {
   i++;

   //Calculate the curvature,the normal to curve, and the surface energy term at each point.
   curvature(pos,ds,nsteps_s,curv);
   norm(pos,nsteps_s,&normal);
   surfenergyterm(normal,nsteps_s,surfenergy);

   //Calculate new positions after step dt.
```

```
tryagain:      //If timestep too large, try this loop again.
newpos[0] = pos[0];
newpos[nsteps_s - 1] = pos[nsteps_s - 1];
for(j=1;j<nsteps_s-1;j++)
   {
   rate = B *(surfenergy[j+1]*curv[j+1] - 2.*surfenergy[j]*curv[j] + surfenergy[j-1]*curv[j-1])/(ds*ds);
   if((rate*dt) > 0.05*ds) //Timesteps too large--decrease timestep.
      {
      dt *= 0.95;
      goto tryagain;   //If timestep is too large, don't want to finish out loop--go back to beginning.
      }
   newpos[j].x = pos[j].x + rate*dt*normal[j].x;
   newpos[j].y = pos[j].y + rate*dt*normal[j].y;
   }

//Find elapsed time and try to increase timestep every 200 iterations.
time += dt;
if((i%200) == 0)
   dt *= 1.05;

//Put the time and the current profile in the output file every time_out seconds.
if(time >= time_out*p)
   {
   printout(pos,nsteps_s,time,output);
   p++;
   }

//Is the trench filled?  If so, print the time and ask if you want to continue run.
if((get_lo(pos,nsteps_s) >= 0.)&&(flag_time == 0))
   {
   printf("Trench filled at t = %5.1f min\n",time/60.);
   printf("Do you want to continue run?(y/n)\n");
   ch = '\0';
   while((ch != 'y')&&(ch!= 'n')) ch = getchar();
   if(ch == 'n')
      goto finish;
   printout(pos,nsteps_s,time,output);
   flag_time = 1;
   }

//Swap newpos and pos.
temp = pos;
pos = newpos;
newpos = temp;

//Refit curve if any spacing has moved by >10%.  Ignore spacing to last point since it is
//probably off anyway,i.e. go only to nsteps_s-2.
for(j=0;j<nsteps_s-2;j++)
   {
   dist = sqrt((pos[j].x - pos[j+1].x)*(pos[j].x - pos[j+1].x) +
               (pos[j].y - pos[j+1].y)*(pos[j].y - pos[j+1].y));
   if((fabs(ds - dist)) > (0.1*ds))
      {
      nsteps_s = curvefit(&pos, nsteps_s, ds);
      graphprofile(pos,nsteps_s,maxx,maxy,1);
      graphtrench(maxx, maxy, height, width);
      }
   }

//Graph profile every 100 iterations.
if(i%100 == 0)
   {
   graphprofile(pos,nsteps_s,maxx,maxy,0);
   graphtrench(maxx, maxy, height, width);
   graphtime(maxx,maxy,time);
   }
}
```

```
finish:
printout(pos,nsteps_s, time, output);
fclose(input);
fclose(output);
closegraph();
return(0);
}
/*****************************************************************************/
//This subroutine calculates the anisotropic surface energy(gamma(theta)) and
//the torque term associated with this(d2(gamma)/d(theta)2). I am using the
//numbers for Cu from (1)M. McLean and B. Gale, Philos. Mag. _20_, pp. 1033-1045
//(1969). I am going to assume that the y-axis points along the (111) direction.
//This is reasonable because most sputtered Cu films show a strong (111) texture.
//Also assume that theta rotates around the [0,1,1bar] axis so that (100) is
//54.7 degrees from (111).
//The other important thing to realize is that there is no really correct way to
//calculate the d2gammadtheta2 term. I have decided to use a similar method of
//that by (2)H.P. Bonzel, E. Preuss and B. Steffen, Appl. Phys. A _35_, pp. 1-8 (1984).
//To calculate the torque term near a cusp in the surface energy requires
//extremely detailed knowledge of the surface energy near the low Miller indice
//planes. Such information doesn't exist. The most important thing to realize
//is that the torque term will only be significant within a few degrees(maybe 1 or
//2 degrees at most) of a low Miller indice plane and that this torque
//should become very great for these low angles, but the torque term should
//be insignificant when far from this low index planes. See discussion in
//reference (2).
//Finally, I will be using the same function form as that given in reference (2)
//and fitting it in separate regions for the cusps at (100) and (111).
void surfenergyterm(vector *normal, int nsteps, float result[400])
{
int j;
float gamma;
float a,b,c;    //Constants used for curve fitting near cusp.
float alpha;
float theta;
float tprime, dtprimedtheta, d2tprimedtheta2;
float g, dgdtprime,d2gdtprime2;
float m,dmdtprime, d2mdtprime2;
float energy,d2;  //energy is gamma(theta), and d2 is the second derivative of gamma wrt theta.

for(j=1;j<nsteps-1;j++)
   {
   //Find the angle between the x-axis and the normal to the curve. Convert to degrees.
   if(normal[j].x == 0.) //normal is in the direction of y-axis
      alpha = pi/2.;
   else                 //arctan only returns values from -pi/2 to pi/2
      if(normal[j].x > 0.)
         alpha = atan(normal[j].y/normal[j].x);
      else
         alpha = pi + atan(normal[j].y/normal[j].x);
   alpha *= 180./pi; //Alpha is between -180 and 180 degrees.
   alpha -= 90.; //Alpha is now the angle between the y-axis(111) and the normal.
   if(alpha < -180.) //Force alpha to be between -180 and 180 degrees.
      alpha += 360.;

   //I've set alpha to be the angle between the y-axis and the normal for ease
   //in calculation in the next section. Assume the y-axis is any of the {111}
   //directions as we tilt away from that, I am forcing mirror symmetry across the typical
   //profile I use, rather than assuming that I have a single crystal grain for
   //the whole profile. This essentially divides the profile into two grains
   //with the "grain boundary" at the bottom center of the trench.

   //Figure out what "region" we're in and assign a value of theta for the region.
   //alpha=0, (1,1bar,1bar); alpha=54.7,(100); alpha=109.4,(111); assume that
   //-alpha has same orientations.
   //I have two regions: (1bar,1,1) and (111) include alpha=[0,27],[81,126] respectively
```

```
   //and (100) which includes alpha=[27,81]. Also assume -alpha => alpha.
   //Try (initially at least to exclude any alpha > 126 so that I don't also
   //have to fit the (011) cusp.
   //Find theta which is the angle that the surface is tilted from the nearest low index plane.
   alpha = fabs(alpha);
   if(alpha > 126.)
      {
      printf("alpha > 126\n");
      pause();
      }
   if((alpha > 27.)&&(alpha < 81.))   //(100)
      {
      theta = alpha - 54.735;
      gamma = gamma100;
      a = 0.007;
      b = 5.;
      c = 225.;
      }
//   else if((alpha >= 126.)&&(alpha <=162.))  //(011)
//      {
//      theta = alpha - 144.735;
//      gamma = gamma100*1.011;
//      a = 0.003;
//      b = 7.;
//      c = 144.;
//      }
   else   //{111}
      {
      gamma = gamma100*0.994;
      a = 0.012;
      b = 7.;
      c = 225.;
      if((alpha <= 27.)&&(alpha >=0.))          //(1bar,1,1)
         theta = alpha;
      else if((alpha >= 81.)&&(alpha <= 126.)) //(111)
         theta = alpha - 2.*54.735;
      else                                      //(1bar,1,1)
         theta = theta - 180.;
      }

   //Fit to the surface energy function and find it's second derivative with respect to theta.
   //Check to see that theta is between -45 and 45 degrees as well(derivative derived for these values).
   if(fabs(theta) > 45.)
      {
      printf("Theta too big!!\n");
      pause();
      }
   theta *= pi/180.;                              //Convert theta back to radians.
   tprime = sqrt(theta*theta + 1./(c*c)) - 1./c;   //Changing c allows one to change curvature near cusp.

   //Find surface energy as a function of theta.
   m = 1. - fabs(cos(tprime) - sin(fabs(tprime)));
   g = 2.*exp(-b*m) - 1.;
   energy = gamma*(1. - a*g); //Surface energy at theta.

   //Find second derivative of surface energy wrt theta, d2.
   //The expressions here for dmdtheta and d2mdtheta2 assume that theta is between -45 and 45 degrees.
   dtprimedtheta = theta/(tprime + 1./c);
   d2tprimedtheta2 = (1. - theta*theta/(theta*theta + 1/(c*c)))/(tprime + 1./c);
   if(tprime > 0.)
      dmdtprime = sin(tprime) + cos(tprime);
   else
      dmdtprime = -sin(tprime) + cos(tprime);
   d2mdtprime2 = cos(tprime) - sin(fabs(tprime));
   dgdtprime = -b*(g+1.)*dmdtprime;
   d2gdtprime2 = -b*(d2mdtprime2*(g+1.) + dmdtprime*dgdtprime);
   d2 = -a*gamma*(dgdtprime*d2tprimedtheta2 + d2gdtprime2*dtprimedtheta*dtprimedtheta);
```

```
   result[j] = (energy + d2)/gamma100; //I am returning the normalized value.
   }

return;
}
/*****************************************************************************/
void printout(vector *p, int nsteps, float t, FILE *output)
{
int i;

fprintf(output,"t=%f\n",t);
for(i=0;i<nsteps;i++)
   fprintf(output,"%7.4f %7.4f\n",p[i].x, p[i].y);

return;
}
/*****************************************************************************/
/*Find maximum point in y and return its value.*/
float get_hi(vector *p, int nsteps)
{
float max;
int j;

max = p[0].y;
for(j=1;j<nsteps;j++)
   {
   if(p[j].y > max)
      max = p[j].y;
   }

return(max);
}
/*****************************************************************************/
/*Find minimum point in y and return its position in array.*/
float get_lo(vector *p, int nsteps)
{
float min;
int j;

min = p[0].y;
for(j=1;j<nsteps;j++)
   {
   if(p[j].y < min)
      min = p[j].y;
   }

return(min);
}
/*****************************************************************************/
/*Calculate arclength*/
float arclength(vector *pos, int nsteps)
{
int j;
float s,ds,ds2;

s = 0.;
for(j=1;j<nsteps;j++)
   {
   ds2 = (pos[j].x - pos[j-1].x)*(pos[j].x - pos[j-1].x) +
         (pos[j].y - pos[j-1].y)*(pos[j].y - pos[j-1].y);
   ds = sqrt(ds2);
   s += ds;
   }

return(s);
}
```

```
/******************************************************************************/
//Fit curve so that ds is constant, function returns the number of steps in new fit.
int curvefit(vector **p, int nsteps, float ds)
{
int i,j;
int nsteps_new;          //The number of steps that will be in the new curve.
float s;
float *a,*b,*c,*rx,*ry;
float h,h1,h2;
float skip;
int flag;
float coef1, coef2, coef3, coef4;
float *d2xds2, *d2yds2;
vector *temp;
vector lastpoint;

//Calculate how many sections to divide curve into.  This code occasionally has problems
//fitting near the end of the profile; I'm not sure why this is, perhaps it is cumulative
//errors in adding up the distances of/to each section.  The neurotic points can be gotten
//rid of by making nsteps_new a little smaller than it really should be--nsteps_new should
//be (int)(s/ds) + 1.  The distance to the last point is always wrong anyway, so usually I
//just make nsteps_new small and hence make the last section too big--the curvature is zero
//at the end anyway, so it shouldn't really introduce any errors.
//Also note below that I have the ability to change nsteps_new to a smaller # if
//i becomes greater than (nsteps-2).  This is needed due to accumulated
//errors in the calculation of the arclength between the new and the old curves.
//Modified 3/95--RAB.
s = arclength(*p,nsteps);
nsteps_new = (int)(s/ds);
//Also, keep the last point from old position array so that curve doesn't get lost
//with many iterations.
nsteps_new++;  //Increment nsteps_new for array definition, then
               //decrement it until the end of routine.
lastpoint = (*p)[nsteps-1];

if(nsteps_new > 400)
   {
   printf("nsteps_new = %i   Memory problems!!!!!!!\n",nsteps_new);
   pause();
   }

//Allocate space for these arrays.  I am defining in this fashion since I am
//sometimes short of memory.  They must be deleted at the end of routine.
a = new float[nsteps];
b = new float[nsteps];
c = new float[nsteps];
rx = new float[nsteps];
ry = new float[nsteps];
temp = new vector[nsteps_new];
d2xds2 = new float[nsteps];
d2yds2 = new float[nsteps];
nsteps_new--;  //Temporarily decrement.  Have to figure the last point out
               //separately at the end of program separately.

//Solve set of equations for d2xds2 and d2yds2 found using cubic spline type set-up.
//Set endpoints to have d2xds2 = d2yds2 = 0.
a[0] = 0.;
b[0]  = 1.;
c[0] = 0.;
rx[0] = 0.;
ry[0] = 0.;
a[nsteps-1] = 0.;
b[nsteps-1] = 1.;
c[nsteps-1] = 0.;
rx[nsteps-1] = 0.;
ry[nsteps-1] = 0.;
for(j=1;j<(nsteps-1);j++)
```

```
   {
   h1 = sqrt(((*p)[j].x - (*p)[j-1].x)*((*p)[j].x - (*p)[j-1].x) +
            ((*p)[j].y - (*p)[j-1].y)*((*p)[j].y - (*p)[j-1].y));
   h2 = sqrt(((*p)[j+1].x - (*p)[j].x)*((*p)[j+1].x - (*p)[j].x) +
            ((*p)[j+1].y - (*p)[j].y)*((*p)[j+1].y - (*p)[j].y));

   a[j] = h1/6.;
   b[j] = (h1 + h2)/3.;
   c[j] = h2/6.;

   rx[j] = (*p)[j+1].x/h2 - (*p)[j].x/h2 - (*p)[j].x/h1 + (*p)[j-1].x/h1;
   ry[j] = (*p)[j+1].y/h2 - (*p)[j].y/h2 - (*p)[j].y/h1 + (*p)[j-1].y/h1;
   }
tridiag(a,b,c,rx,d2xds2,nsteps);
tridiag(a,b,c,ry,d2yds2,nsteps);

//Fit curve using cubic spline calculation.
temp[0] = (*p)[0];
i = 0;
skip = 0.;
flag = 0;
for(j=1;j<nsteps_new;j++)
   {
   //Find the correct interval to fit between and calculate what s is for that interval.
   tryagain:
   if(arclength((*p),i+2) >= (arclength(temp,j) + ds))
      {
      if(flag == 0)  //We are in the same interval as for last value of temp.
         s = ds + sqrt(((*p)[i].x - temp[j-1].x)*((*p)[i].x - temp[j-1].x) +
                       ((*p)[i].y - temp[j-1].y)*((*p)[i].y - temp[j-1].y));
      else   //We are in a different interval from last time.
         s = ds - skip;
      //Calculate the total spacing between the two points that we are going to fit.
      h = sqrt(((*p)[i].x - (*p)[i+1].x)*((*p)[i].x - (*p)[i+1].x) +
              ((*p)[i].y - (*p)[i+1].y)*((*p)[i].y - (*p)[i+1].y));
      //Reset the variables for the next iteration.
      skip = 0.;
      flag = 0;
      }
   else   //Incorrect interval.
      {
      //Calculate the value of s that is still left in the interval we are skipping out of.
      if(flag == 0)
         skip += sqrt(((*p)[i+1].x - temp[j-1].x)*((*p)[i+1].x - temp[j-1].x) +
                      ((*p)[i+1].y - temp[j-1].y)*((*p)[i+1].y - temp[j-1].y));
      else
         skip +=sqrt(((*p)[i+1].x - (*p)[i].x)*((*p)[i+1].x - (*p)[i].x) +
                    ((*p)[i+1].y - (*p)[i].y)*((*p)[i+1].y - (*p)[i].y));
      //Go to the next interval.
      i++;
      if(i > (nsteps-2))//Fixes problem.  See note at beginning of subroutine.
         {
         nsteps_new = j;
         goto quit;
         }
      //Flag = 1 indicates that we have switched intervals.
      flag = 1;
      //With the new value of i, go back and check to see if we are now in the correct interval.
      goto tryagain;
      }

   //Do the fit.
   coef1 = 1. - (s/h);
   coef2 = 1. - coef1;
   coef3 = (coef1*coef1*coef1 - coef1)*h*h/6.;
   coef4 = (coef2*coef2*coef2 - coef2)*h*h/6.;
   temp[j].x = coef1*(*p)[i].x + coef2*(*p)[i+1].x + coef3*d2xds2[i] + coef4*d2xds2[i+1];
```

```
   temp[j].y = coef1*(*p)[i].y + coef2*(*p)[i+1].y + coef3*d2yds2[i] + coef4*d2yds2[i+1];
   }

quit:
//Assign values in temp to (*p) and assign a value to the last point.
for(i=0;i<nsteps_new;i++)
   (*p)[i] = temp[i];
if((fabs((*p)[nsteps_new-1].x - lastpoint.x) < .01)&&(fabs((*p)[nsteps_new-1].y - lastpoint.y)< .01))
   {
   }
else
   {
   nsteps_new++;
   (*p)[nsteps_new-1] = lastpoint;
   }

//Delete all the memory that I allocated.
delete[]a;
delete[]b;
delete[]c;
delete[]rx;
delete[]ry;
delete[]temp;
delete[]d2xds2;
delete[]d2yds2;

return(nsteps_new);
}
/*****************************************************************************/
//Taken from Numerical Recipes in C, Second Edition, page 51.  This solves the
//tridiagonal problem matrix problem for u where Mu = r with a below the
//diagonal, b the diagonal and c above the diagonal.
void tridiag(float *a ,float  *b, float *c, float *r, float *u, int nsteps)
{
int j;
float bet;
float *gam=new float[nsteps];

//printf("Made it to tridiag\n");
//pause();
bet = b[0];
if(bet == 0.)
   printf("Error 1 in tridiag\n");
u[0] = r[0]/bet;

for(j=1;j<nsteps;j++)
   {
   gam[j] = c[j-1]/bet;
   bet = b[j] - a[j]*gam[j];
   if(bet == 0.)
      printf("Error 2 in tridiag: j = %i\n",j);
   u[j] = (r[j] - a[j]*u[j-1])/bet;
   }

for(j=(nsteps-2);j==0;j--)
   u[j] -= gam[j+1]*u[j+1];

delete[]gam;
return;
}
/*****************************************************************************/
/*Calculate curvature along profile*/
void curvature(vector *p, float ds, int nsteps, float curve[400])
{
int j;
float dxds, dyds;
float dx2ds2, dy2ds2;
```

```
float sum_sq;
float denom;

curve[0] = 0.;
curve[nsteps - 1] = 0.;
for(j=1;j<nsteps-1;j++)
   {
   dxds = (p[j+1].x - p[j-1].x)/(2.*ds);
   dyds = (p[j+1].y - p[j-1].y)/(2.*ds);
   dx2ds2 = (p[j+1].x - 2.*p[j].x + p[j-1].x)/(ds*ds);
   dy2ds2 = (p[j+1].y - 2.*p[j].y + p[j-1].y)/(ds*ds);
   sum_sq = (dxds)*(dxds) + (dyds)*(dyds);
   denom = sum_sq * sqrt(sum_sq);
   /*The sign of the curvature corresponds to the convention Mullins uses.*/
   curve[j] = -(dxds * dy2ds2 - dyds * dx2ds2)/denom;
   }

return;
}
/********************************************************************************/
/*Calculate the normal to the curve(neglects endpoints since this aren't needed).*/
void norm(vector *p, int nsteps, vector **result)
{
float xcomp,ycomp;
float m;
float normalize;
int j;

//This routine calculates the normal for everything but the endpoints.
for(j=1;j<nsteps-1;j++)
   {
   if(p[j+1].x == p[j-1].x)   /*Line is vertical*/
      {
      if(p[j-1].y > p[j+1].y)
         {
         xcomp = 1.;
         ycomp = 0.;
         }
      else
         {
         xcomp = -1.;
         ycomp = 0.;
         }
      }
   else                        /*Line is not vertical*/
      {
      m = (p[j+1].y - p[j-1].y)/(p[j+1].x - p[j-1].x);
      if(((m > 0.)&&(p[j-1].y > p[j+1].y)) || ((m < 0.)&&(p[j-1].y < p[j+1].y)))
         {
         xcomp = m;
         ycomp = -1.;
         }
      else
         {
         xcomp = -m;
         ycomp = 1.;
         }
      }
   normalize = sqrt(xcomp*xcomp + ycomp*ycomp);
   (*result)[j].x = xcomp/normalize;
   (*result)[j].y = ycomp/normalize;
   }

return;
}
/********************************************************************************/
void err_message()
```

```
{
printf("usage: reflow [-T temperature(K)] [-t time_end (min)]\n");
printf("\t[-h trench height(micron)] [-w trench width (micron)] [-i filename_i]\n");
printf("\t[-o filename_o] [-s mesh size(micron)] [-j time_out (min)]\n");
printf("Default values:\n");
printf("\t T = 800K\n");
printf("\t time_end = 60 min\n");
printf("\t time_out = 10 min\n");
printf("\t ds = 0.1 microns\n");
printf("\t h = 1 micron\n");
printf("\t w = 1 micron\n");
printf("\t filename_i = profile\n");
printf("\t filename_o = reflowed\n");

return;
}
/******************************************************************************/
void setupgraph()
{
/*request autodetection*/
int gdriver = DETECT, gmode, errorcode;

/*initialize graphics mode*/
initgraph(&gdriver, &gmode, "");

/*read result of initialization*/
errorcode = graphresult();

if (errorcode != grOk)   /*an error occured*/
   {
   printf("graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);
   }

return;
}
/******************************************************************************/
/*Make the screen width 2 hours*/
void graphtime(int xmax, int ymax, float t_sec)
{
float t_hr;
float scr_w; //Screen width in hours.

scr_w = 2.;
setcolor(LIGHTRED);

outtextxy(0,ymax-textheight("0"),"0");
outtextxy((int)(1.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(2.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(3.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(4.*xmax/8.),ymax-textheight("0"),"1");
outtextxy((int)(5.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(6.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(7.*xmax/8.),ymax-textheight("0"),"|");
outtextxy(xmax-textwidth("2"),ymax-textheight("0"),"2");

t_hr = t_sec/3600.;
moveto(0,ymax-2.*textheight("0"));
lineto((int)(xmax*t_hr/scr_w),ymax-2.*textheight("0"));

return;
}
/******************************************************************************/
/*Make width of graphics window correspond to 10 microns(adj. with scr_w)*/
/*h = trench height in cm, w = trench width in cm*/
```

```
/*For graphics window, lower left at (0,0); upper right at (xmax, ymax)*/
void graphtrench(int xmax, int ymax, float h, float w)
{
float onemicron, scr_w;
float x,y;

setcolor(WHITE);
scr_w = 10.;   /*scale for width of graphics window(in microns)*/
onemicron = (float)(xmax/scr_w);   /*convert 1 micron to pixel units*/

moveto(0,(int)(ymax/2));   /*start halfway up screen*/

x = onemicron*(scr_w/2. - w/2.);
y = ymax/2.;
lineto((int)x,(int)y);

y = ymax/2. + onemicron*h;
lineto((int)x,(int)y);

x = onemicron*(scr_w/2. + w/2);
lineto((int)x,(int)y);

y = (int)(ymax/2);
lineto((int)x,(int)y);

lineto(xmax, (int)(ymax/2));

return;
}
/******************************************************************************/
void graphprofile(vector *p, int nstep, int xmax, int ymax, int color)
{
float onemicron, scr_w;
float x,y;
int j;

scr_w = 10.;
onemicron = (float)(xmax/scr_w);
clearviewport();
moveto(0,(int)(ymax/2. - onemicron*p[0].y));
for(j=1; j<nstep; j++)
   {
   if(color == 0)
      setcolor(YELLOW);
   else
      setcolor(LIGHTGREEN);
   x = onemicron*(p[j].x - p[0].x);
   y = ymax/2. - onemicron*p[j].y;
   lineto((int)x,(int)y);
   }

return;
}
/******************************************************************************/
void pause()
{
char ch;

ch = '\0';
printf("Press return to continue.....\n");
while(ch != '\n') ch = getchar();
return;
}
/******************************************************************************/
```

Appendix C Algorithm for Reflow with Grain Boundary/Surface Interactions

The following algorithm was used to model surface diffusion–mediated reflow with the inclusion of grain boundaries.

C.1 Subroutines

The following subroutines are used here in addition to those described in Section A.2:

dfunction Calculate the derivatives of the three boundary condition functions.

dkidxyi Calculate the derivative of the curvature, K_i, with respect to x_i or y_i.

dkidxyj Calculate the derivative of the curvature, K_i, with respect to x_j or y_j.

function Calculate the values of the three boundary condition functions.

newton Solves three coupled, non-linear equations using an adaptation of the Newton-Raphson method with backtracking. See Numerical Recipes, 2nd edition, page 379-382.

precurvefit Divides the profile into two regions that are separated by the grain boundary before implementing the curvefit subroutine.

C.2 Code

C.2.1 reflowgb.h

```
//Define vector as a new type so that I can use dynamic memory.
//The default of 'class' is 'private'.  'class' is equivalent to 'struct'.
//Because of my definition of vector, I had to change the numerical recipes
//name of it's 'vector' routine to 'vect'.
class vector
   {
   public:
   float x,y;
   int label;
   };

/*Define types of all subroutines*/
float curvature(vector p1, vector p2, vector p3);
void tridiag(float *a, float *b, float *c, float *r, float *u, int nsteps);
void precurvefit(vector **p, int *nsteps_s, float ds, int *j_gb);
int curvefit(vector **p, int nsteps, float ds);
vector norm(vector p1, vector p2, vector p3);
void err_message();
void setupgraph();
void graphtrench(int xmax, int ymax, float h, float v);
void graphprofile(vector *p, int nstep, int xmax, int ymax);
void pause();
float get_hi(vector *p, int nsteps);
float get_lo(vector *p, int nsteps);
void printout(vector *p, int nsteps, float t, FILE *output);
void graphtime(int xmax, int ymax, float t_sec);
void newton(vector **np, vector **p, int j_gb);
float arclength(vector *pos, int nsteps);
void function(vector *p, int j, float *f);
void dfunction(vector *p, int j, float **dfdx, char axis);
float dkidxyi(vector *p, int j, char axis);
float dkidxyj(vector *p, int i, int j, char axis);

/*These are external subroutines pulled from Numerical Recipes.*/
extern "C" void lubksb(float **a, int n, int *indx, float b[]);
extern "C" void ludcmp(float **a, int n, int *indx, float *d);
extern "C" float *vect(long nl, long nh);
extern "C" int *ivector(long nl, long nh);
extern "C" float **matrix(long nrl, long nrh, long ncl, long nch);
extern "C" void free_vector(float *v, long nl, long nh);
extern "C" void free_ivector(int *v, long nl, long nh);
extern "C" void free_matrix(float **m, long nrl, long nrh, long ncl, long nch);
```

C.2.2 reflowgb.cpp

```
//The purpose of this program is to reflow an arbitrary surface by surface
//diffusion.  The constants in the system here are for a Cu(111) surface.  It is
//written in a combination of ANSI C with a few C++ commands sprinkled in where
//it was needed to make the programming easy. To obtain help with it, type
//'reflow help' from the DOS prompt and it will prompt you for the inputs
//and show you the defaults values of the parameters(unless I forgot to update
//the error_message).  This code was developed with the Borland C++ (version 4)
//package; if you're trying to take it somewhere else, you will also need to
//grab the 'reflow.h' file since it defines a new type and all the subroutines
//for this program.  Ruth Brain, 7/94.
//Major updates were made to this code in Sept and Oct 1994.  I added the whole
//section to refit the curve as the distance between the points changes.  I also
//corrected the normal subroutine.  RAB 10/94
//Add grain boundaries.  RAB 2/95
#include <stdio.h>
```

```
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
#include "reflowgb.h"      //The library that I have defined to define my subroutines as well
                           //as set-up 'vector' as a new type.

#define gamma_gb 600.      //grain boundary energy (dyne/cm)
#define gamma_s 1800.      //surface energy (dyne/cm)
#define Ds0 0.07           //surface diffusion pre-exponential (cm^2/s)
#define Qs  0.82           //surface diffusion activation energy (eV)
#define omega 1.2e-23      //atomic volume (cm^3/atom)
#define nu 1.43e15         //atoms per unit surface area (atoms/cm^2)
#define k 8.62e-5          //Boltzmann's constant (eV/K)
#define e 2.71828          //(d/dx)(e^x) = e^x

void main(int argc,char *argv[])
{
long i;
int j;
int j_gb;   /*pos[j_gb].label = 0, i.e. we're at the grain boundary*/
int p;
int flag;
int flag_time;
int flag_fit;
float dist;
char ch;
float time_end;         //The program will run through time_end seconds.
float time_out;         //Every time_out seconds, the current profile will be saved
                        //to the output file.
int nsteps_s;           //number of steps in arc; program uses a variable
                        //mesh(fixed ds) so nsteps_s will change
float ds;               //element of arc length (in microns)
float dt;               //element of time (in sec)
float time;             //Tells how much time has passed (in sec) during reflow--necessary
                        //with variable time step
//The next 4 definitions allow those variables to use dynamic memory rather
//than static(since I only have 64k of static). This type of definition also allows me
//to change where they are pointing when I switch newpos and pos.
float *curv=new float[400];        /*curvature at (x(s),y(s))*/
vector *pos=new vector[400];       /*(x,y) of surface (in microns)*/
vector *newpos=new vector[400];    /*revised (x,y) of surface*/
vector *temp;                      /*temporary pointer used to allow *pos to point at *newpos*/
float Ds;               /*surface diffusivity (in cm^2/s)*/
float T;                /*temperature (in K)*/
float B;                /*coef. for surface diffusion diff. eq.(in micron^4/s)*/
float rate;             /*rate of movement of normal(in microns/s)*/
vector normal;          /*normal to curve*/
float height;           /*trench height (in microns)*/
float width;            /*trench width (in microns)*/

char *filename_i;       /*the filename containing the initial profile's (s,x,y)*/
char *filename_o;       /*                          output                      */

int maxx;               /*number of pixels in graphics window along x axis*/
int maxy;               /*ditto along y axis*/

FILE *input;            /*Input file that contains the initial profile to reflow.*/
FILE *output;           /*Output file that contains time and profile data.*/

/*Put in default values in case they are not entered*/
T = 800.;          //in Kelvin
time_out = 600.;  //in seconds
time_end = 3600.; //in seconds
ds = 0.1;          //in microns
```

```
height = 1.0;     //in microns
width = 1.0;      //in microns
filename_i = "profile";
filename_o = "reflowed";

/*Read in command-line arguments.*/
j = 1;
while (j < argc)
   {
   if(argv[j][0] == '-')
      {
      switch(argv[j][1])
         {
         case 'i':   /*input filename*/
            filename_i = argv[j+1];
            j += 2;
            break;
         case 'o':   /*output filename*/
            filename_o = argv[j+1];
            j += 2;
            break;
         case 't':   /*time til end of run: time_end (sec) but entered in minutes*/
            time_end = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'j':   /*spacing between saving data to filename_o; entered in minutes*/
            time_out = atof(argv[j+1])*60.;
            j += 2;
            break;
         case 'T':   /*temperature (K)*/
            T = atof(argv[j+1]);
            j += 2;
            break;
         case 's':   /*mesh size (microns)*/
            ds = atof(argv[j+1]);
            j += 2;
            break;
         case 'h':   /*trench height (micron)*/
            height = atof(argv[j+1]);
            j += 2;
            break;
         case 'w':   /*trench width (micron)*/
            width = atof(argv[j+1]);
            j += 2;
            break;
         default:
            err_message();
            goto finish;
         }
      }
    else
      {
      err_message();
      goto finish;
      }
   }

/*Calculate constants for run*/
/*Ds = surface diffusivity (cm^2/s)*/
Ds = Ds0 * exp(-Qs/(k*T));
/*The constants in front of the expression make the units for B microns^4/s*/
B = (1e16)*(6.242e11)*(Ds * gamma_s * omega * omega * nu)/(k * T);

/*Read (pos[j].x, pos[j].y, pos[j].label) = (x,y,grain label) at t=0 from 'profile'.*/
input = fopen(filename_i,"r");
j = 0;
while(fscanf(input,"%f %f %i\n",&(pos[j].x), &(pos[j].y), &(pos[j].label)) != EOF)
```

```
    {
    if(pos[j].label == 0)
       j_gb = j;
    j++;
    }
nsteps_s = j;

/*Set-up graphics output*/
setupgraph();
/*Finds the maximum number of pixels for the chosen graphics window*/
maxx = getmaxx();
maxy = getmaxy();

/*Graph the trench profile and initial profile.*/
graphprofile(pos,nsteps_s,maxx,maxy);
graphtrench(maxx, maxy, height, width);
printf("Initial profile\n");
pause();

//Fit input curve so that it has an equal mesh(i.e. equal ds between points).
precurvefit(&pos,&nsteps_s,ds,&j_gb);

//Graph initial profile after refitting.
graphprofile(pos,nsteps_s,maxx,maxy);
graphtrench(maxx, maxy, height, width);
printf("Initial profile after refit\n");
pause();

//Open the output file up and put in the time and the initial profile.
output = fopen(filename_o,"w");
printout(pos,nsteps_s,0.,output);

/*Main loop*/
dt = (ds*ds*ds*ds)/(4.*B);    //in sec; equations become unstable if dt>= (ds^4)/(2*B), i.e. eq
                              //unstable as diffuse across several cells in one step
time = 0.;   //in sec
p = 1;
flag_time = 0;
i=0;
while(time < time_end)
    {
    //Calculate curvature.
    curv[0] = 0.;
    curv[nsteps_s-1] = 0.;
    for(j=1;j<nsteps_s-1;j++)
       curv[j] = curvature(pos[j-1],pos[j],pos[j+1]);

    //Calculate newpositions after step dt.
    //Boundary conditions give me fixed endpoints.
    newpos[0] = pos[0];
    newpos[nsteps_s - 1] = pos[nsteps_s - 1];
    tryagain:  //If timestep too large, try this loop again.
    for(j=1;j<nsteps_s-1;j++)
       {
       if(pos[j-1].label == pos[j+1].label)   //If we are not near grain boundary, reflow as usual.
          {
          rate = B * (curv[j+1] - 2.*curv[j] + curv[j-1])/(ds*ds);
          normal = norm(pos[j-1],pos[j],pos[j+1]);
          newpos[j].x = pos[j].x + rate*dt*normal.x;
          newpos[j].y = pos[j].y + rate*dt*normal.y;
          newpos[j].label = pos[j].label;
          if((rate*dt) > 0.02*ds)   //Timesteps too large--decrease timestep.
             {
             dt *= 0.95;
             goto tryagain;   //If timestep is too large, don't want to finish out
                              //loop--go back to beginning.
             }
```

```
      }
   else   //The points on each side of boundary and the boundary itself must
          //be fit by special boundary conditions.  Take old value as first guess to new value.
      newpos[j] = pos[j];
   }
//Move the grain boundary region according the the boundary conditions.  Here we have
//used an initial guess(i.e. the old position) and put in newpos.
newton(&newpos,&pos,j_gb);

//Find elapsed time and try to increase timestep every 200 iterations.
time += dt;
if((i%200) == 0)
   dt *= 1.05;

//Put the time and the current profile in the output file every time_out seconds.
if(time >= time_out*p)
   {
   printout(pos,nsteps_s,time,output);
   p++;
   }

//Is the trench filled?  If so, print the time and ask if you want to continue run.
if((get_lo(pos,nsteps_s) >= 0.)&&(flag_time == 0))
   {
   printf("Trench filled at t = %5.1f min\n",time/60.);
   printf("Do you want to continue run?(y/n)\n");
   ch = '\0';
   while((ch != 'y')&&(ch!= 'n')) ch = getchar();
   if(ch == 'n')
      goto finish;
   printout(pos,nsteps_s,time,output);
   flag_time = 1;
   }

//Swap newpos and pos.
temp = pos;
pos = newpos;
newpos = temp;

//Refit curve if any spacing has moved by >5%.  Ignore spacing to first and last points
//since it is probably off anyway,i.e. go only to nsteps_s-2.
j=1;
flag_fit = 0;
while((j<nsteps_s-2)&&(flag_fit == 0))
   {
   dist = sqrt((pos[j].x - pos[j+1].x)*(pos[j].x - pos[j+1].x) +
               (pos[j].y - pos[j+1].y)*(pos[j].y - pos[j+1].y));
   if((fabs(ds - dist)) > (0.05*ds))
       flag_fit = 1;
   j++;
   }

//Fit curves separately on each side of grain boundary, starting at the grain
//boundary so that there is no change in the grain boundary position.
if(flag_fit == 1)
   {
   precurvefit(&pos,&nsteps_s,ds,&j_gb);
   graphprofile(pos,nsteps_s,maxx,maxy);
   graphtrench(maxx, maxy, height, width);
   }

//Graph profile every 100 iterations.
if(i%100 == 0)
   {
   graphprofile(pos,nsteps_s,maxx,maxy);
   graphtrench(maxx, maxy, height, width);
   graphtime(maxx,maxy,time);
```

```
      }

   i++;
   }

finish:
printout(pos,nsteps_s,time,output);
fclose(input);
fclose(output);
closegraph();
return;
}
/******************************************************************************/
//The derivative of the curvature at p[i] wrt p[i].x(axis='x') or p[i].y(axis='y').
float dkidxyi(vector *p,int j,char axis)
{
float num,denom;
float temp;
float result;

if(axis == 'x')
   num = -8.*(p[j+1].y - p[j-1].y);
else
   num = 8.*(p[j+1].x - p[j-1].x);

temp = (p[j+1].x - p[j-1].x)*(p[j+1].x - p[j-1].x) + (p[j+1].y - p[j-1].y)*(p[j+1].y - p[j-1].y);
denom = temp*sqrt(temp);
result = num/denom;

return(result);
}
/******************************************************************************/
//The derivative of the curvature at p[j] wrt p[j+-1].x(axis='x') or p[j+-1].y(axis='y').
float dkidxyj(vector *p, int i, int j, char axis)
{
float term1,term2;
float sign;
float result;

if(j == (i+1))
   sign = -1.;
else if (j == (i-1))
   sign = 1.;
else
   printf("Problem with dkidxyj routine.\n");

term1 = (p[i+1].x - p[i-1].x)*(p[i+1].x - p[i-1].x) + (p[i+1].y - p[i-1].y)*(p[i+1].y - p[i-1].y);

if(axis == 'x')
   {
   term2 = 4.*sign*(p[i+1].y - 2.*p[i].y + p[i-1].y) + 4.*(p[i+1].y - p[i-1].y);
   result = 3.*sign*curvature(p[i-1],p[i],p[i+1])*(p[i+1].x - p[i-1].x)/term1;
   }
else
   {
   term2 = -4.*sign*(p[i+1].x - 2.*p[i].x + p[i-1].x) - 4.*(p[i+1].x - p[i-1].x);
   result = 3.*sign*curvature(p[i-1],p[i],p[i+1])*(p[i+1].y - p[i-1].y)/term1;
   }

result += term2/(term1*sqrt(term1));

return(result);
}
/******************************************************************************/
//Evaluate the three functions at grain boundary.
//f[1] is the condition that there be a fixed groove angle.
//f[2] is the condition that the surface currents across gb should be equal.
```

```
//f[3] is the condition that the curvatures across gb should be equal.
//When newton has solved the equations correctly, f[1]=f[2]=f[3]=0
void function(vector *p, int j, float *f)
{
float m;
float dot,cross;
float tana;

m = gamma_gb/(2.*gamma_s);
tana = m*sqrt(1. - m*m)/(m*m - 0.5);

cross = (p[j+1].x - p[j].x)*(p[j-1].y - p[j].y) - (p[j+1].y - p[j].y)*(p[j-1].x - p[j].x);
dot = (p[j+1].x - p[j].x)*(p[j-1].x - p[j].x) + (p[j+1].y - p[j].y)*(p[j-1].y - p[j].y);
f[1] = cross - dot*tana;
f[2] = curvature(p[j-3],p[j-2],p[j-1]) - curvature(p[j-2],p[j-1],p[j]) -
       curvature(p[j],p[j+1],p[j+2]) + curvature(p[j+1],p[j+2],p[j+3]);
f[3] = curvature(p[j-2],p[j-1],p[j]) - curvature(p[j],p[j+1],p[j+2]);

//printf("f[1] = %f     f[2] = %f     f[3] = %f\n",f[1],f[2],f[3]);
return;
}
/*****************************************************************************/
void dfunction(vector *p, int j, float **jac, char axis)
{
float m;
float tana;

m=gamma_gb/(2.*gamma_s);
tana = m*sqrt(1. - m*m)/(m*m - 0.5);

if(axis == 'x')   //Derivatives of f[1],f[2],f[3] wrt p[j-1].x,p[j].x,p[j+1].x
   {
   jac[1][1] = -(p[j+1].y - p[j].y) - tana*(p[j+1].x - p[j].x);
   jac[1][2] = (p[j+1].y - p[j-1].y) + tana*(p[j+1].x - 2.*p[j].x + p[j-1].x);
   jac[1][3] = (p[j-1].y - p[j].y) - tana*(p[j-1].x - p[j].x);
   }
else   //Derivatives of f[1],f[2],f[3] wrt p[j-1].y,p[j].y,p[j+1].y
   {
   jac[1][1] = (p[j+1].x - p[j].x) - tana*(p[j+1].y - p[j].y);
   jac[1][2] = -(p[j+1].x - p[j-1].x) + tana*(p[j+1].y - 2.*p[j].y + p[j-1].y);
   jac[1][3] = -(p[j-1].x - p[j].x) - tana*(p[j-1].y - p[j].y);
   }

jac[2][1] = dkidxyj(p,j-2,j-1,axis) - dkidxyi(p,j-1,axis);
jac[2][2] = -dkidxyj(p,j-1,j,axis) - dkidxyj(p,j+1,j,axis);
jac[2][3] = dkidxyj(p,j+2,j+1,axis) - dkidxyi(p,j+1,axis);

jac[3][1] = dkidxyi(p,j-1,axis);
jac[3][2] = dkidxyj(p,j-1,j,axis) - dkidxyj(p,j+1,j,axis);
jac[3][3] = -dkidxyi(p,j+1,axis);

//printf("jac[1][1] = %f\njac[1][2] = %f\njac[1][3] = %f\n",jac[1][1],jac[1][2],jac[1][3]);
//printf("jac[2][1] = %f\njac[2][2] = %f\njac[2][3] = %f\n",jac[2][1],jac[2][2],jac[2][3]);
//printf("jac[3][1] = %f\njac[3][2] = %f\njac[3][3] = %f\n",jac[3][1],jac[3][2],jac[3][3]);

return;
}
/*****************************************************************************/
//An adaptation of the Newton-Raphson method for solving non-linear, coupled
//equations.  See Numerical Recipes, 2nd edition, page 379-382.  Here, I assume
//that the (x,y) only move along the normal to the surface.
//Also, the only way I could get this subroutine to work with the NR subroutines
//is by defining things in NRs weird way, i.e. all vectors go from (1,n) rather
//than (0,n-1).  That's why here and in function and dfunction routines everything
//is labeled one higher than in the rest of the program.

void newton(vector **np, vector **pos, int j_gb)
```

```
{
char ch;
char axis;
float lambda;
int i,j,n;
float *f;
double ff_new,ff_old;
float **jac;
float *p;
float errf,tolf;
float err_xory,tol_xory;
vector normal[4];
vector temp[4];
float g[4];
float sum;
int *indx;  //These are needed for the matrix solution to keep track of row switches, etc.
float d;

//These subroutine calls make my definitions compatible with the NR definitions.
indx=ivector(1,3);
p=vect(1,3);
f=vect(1,3);
jac=matrix(1,3,1,3);

//Acceptable tolerances for solution.
tol_xory = 0.000002;
tolf = 0.0001;

//Calculate the normals to the three points that I want to solve.
for(j=1;j<=3;j++)
   normal[j] = norm((*pos)[j_gb-3+j],(*pos)[j_gb-2+j],(*pos)[j_gb-1+j]);

//If the normal at the grain boundary is closer to the y-axis than the x-axis then we'll
//fit the points by varying y otherwise we'll fit by varying x.
axis = 'x';
if(normal[2].x == 0.)
   axis = 'y';
else
   {
   if((fabs(normal[2].y/normal[2].x)) >= 1.)
      axis = 'y';
   }
//printf("axis = %c\n",axis);

//Main loop.
i=0;
while(i <= 200)  //200 trials before I give up finding a solution.
   {
   i++;

   //Calculate the functional values.
   //jac=dfdx if axis=x,jac=dfdy if axis=y; jac is the matrix of df(i)/dx(j) or df(i)/dy(j)
   function(*np,j_gb,f);
   dfunction(*np,j_gb,jac,axis);

   //Figure out what the error is in the value of the functions and decide whether to quit or not.
   errf = 0.;
   ff_old = 0.;
   for(j=1;j<=3;j++)
      {
      errf += fabs(f[j]);
      ff_old += 0.5*f[j]*f[j];
      temp[j] = (*np)[j_gb-2+j];
      sum = 0.;
      for(n=1;n<=3;n++)
         sum += jac[n][j]*f[n];
      g[j] = sum;
```

```
      }

//    printf("ff_old = %f\n",ff_old);
//    printf("errf = %f\n",errf);
   if(errf <= tolf)
      goto solved;

   //Solve for dx in the matrix equation (J)(dx) = -f were J is dfdx.  The result ends up in p.
   for(j=1;j<=3;j++)
      p[j] = -f[j];
   ludcmp(jac,3,indx,&d);
   lubksb(jac,3,indx,p);

//    printf("p[1] = %f, p[2] = %f, p[3] = %f\n",p[1],p[2],p[3]);

   //Figure out what error is in the changing of x's or y's, and update the
   //(x,y) pairs taking the full Newton-Raphson step.
   err_xory = 0.;
   lambda = 1.;
   loopagain:
   for(j=1;j<=3;j++)
      {
      p[j] *= lambda;
      err_xory += fabs(p[j]);
      if(axis == 'x')
         {
         (*np)[j_gb-2+j].x = temp[j].x + p[j];
         (*np)[j_gb-2+j].y = temp[j].y + (normal[j].y/normal[j].x)*p[j];
         }
      else
         {
         (*np)[j_gb-2+j].y = temp[j].y + p[j];
         (*np)[j_gb-2+j].x = temp[j].x +(normal[j].x/normal[j].y)*p[j];
         }
      }
   if(err_xory <= tol_xory)
      goto solved;

   //Calculate f*f.
   function(*np,j_gb,f);
   ff_new = 0.;
   for(j=1;j<=3;j++)
      ff_new += 0.5*f[j]*f[j];
   if((ff_new > (ff_old + 0.0001*(g[1]*p[1] + g[2]*p[2] + g[3]*p[3])))&&(ff_new != 0.))
      {
      lambda = 0.5;
      goto loopagain;
      }
   }

if(i >= 200)
   {
   printf("newton was unable to solve in %i iterations\n",i);
   printf("errxory = %f     errf = %f\n",err_xory,errf);
   printf("Do you want to continue run?(y/n)\n");
   ch = '\0';
   while((ch != 'y')&&(ch!= 'n')) ch = getchar();
   if(ch == 'n')
      pause();
   for(j=1;j<=3;j++)
      {
      printf("norm[%i].x = %f   .y = %f\n",j,normal[j].x,normal[j].y);
      (*np)[j_gb-2+j] = (*pos)[j_gb-2+j];
      normal[j] = norm((*pos)[j_gb-2],(*pos)[j_gb],(*pos)[j_gb+2]);
      }
   pause();
   }
```

```
solved:

free_ivector(indx,1,3);
free_vector(p,1,3);
free_vector(f,1,3);
free_matrix(jac,1,3,1,3);
return;
}
/****************************************************************************/
/*Find maximum point in y and return its value.*/
float get_hi(vector *p, int nsteps)
{
float max;
int j;

max = p[0].y;
for(j=1;j<nsteps;j++)
   {
   if(p[j].y > max)
      max = p[j].y;
   }

return(max);
}
/****************************************************************************/
/*Find minimum point in y and return its position in array.*/
float get_lo(vector *p, int nsteps)
{
float min;
int j;

min = p[0].y;
for(j=1;j<nsteps;j++)
   {
   if(p[j].y < min)
      min = p[j].y;
   }

return(min);
}
/****************************************************************************/
/*Calculate arclength*/
float arclength(vector *pos, int nsteps)
{
int j;
float s,ds,ds2;

s = 0.;
for(j=1;j<nsteps;j++)
   {
   ds2 = (pos[j].x - pos[j-1].x)*(pos[j].x - pos[j-1].x) +
         (pos[j].y - pos[j-1].y)*(pos[j].y - pos[j-1].y);
   ds = sqrt(ds2);
   s += ds;
   }

return(s);
}
/****************************************************************************/
//This clumsy looking routine sets up a curve that includes a grain boundary
//to be refit using curvefit.  It will modify **p, *nsteps_s and *j_gb.
void precurvefit(vector **p, int *nsteps, float ds, int *j_gb)
{
int j;
int nsteps_s1,nsteps_s2;
vector *t1=new vector[400];
```

```
vector *t2=new vector[400];

//Put everything to the left of the grain boundary in t1 in reverse order since
//We don't want the grain boundary to move and want the spacing to be accurate
//near the grain boundary--curvefit will put some small error at the endpoint.
for(j=0;j<=(*j_gb);j++)
   t1[j] = (*p)[(*j_gb)-j];
nsteps_s1 = curvefit(&t1,(*j_gb)+1,ds);

//Put everything to the right of the grainboundary in t2 and refit it.
for(j=(*j_gb);j<(*nsteps);j++)
   t2[j-(*j_gb)] = (*p)[j];
nsteps_s2 = curvefit(&t2,(*nsteps)-(*j_gb),ds);

//Dump these results back into (*p).
for(j=0;j<nsteps_s1;j++)
   (*p)[j] = t1[nsteps_s1-1-j];
for(j=1;j<nsteps_s2;j++)        //t2[0] = t1[0]
   (*p)[j+nsteps_s1-1] = t2[j];

//Modify nsteps_s and j_gb to their new values.
(*nsteps) = nsteps_s1 + nsteps_s2 - 1;
(*j_gb) = nsteps_s1 - 1;

//Delete the memory that I allocated.
delete[]t1;
delete[]t2;
return;
}
/*****************************************************************************/
//Fit curve so that ds is constant, function returns the number of steps in new fit.
int curvefit(vector **p, int nsteps, float ds)
{
int i,j;
int nsteps_new;          //The number of steps that will be in the new curve.
float s;
float *a,*b,*c,*rx,*ry;
float h,h1,h2;
float skip;
int flag;
float coef1, coef2, coef3, coef4;
float *d2xds2, *d2yds2;
vector *temp;
vector lastpoint;

//Calculate how many sections to divide curve into.  This code occasionally has problems
//fitting near the end of the profile; I'm not sure why this is, perhaps it is cumulative
//errors in adding up the distances of/to each section.  The neurotic points can be gotten
//rid of by making nsteps_new a little smaller than it really should be--nsteps_new should
//be (int)(s/ds) + 1.  The distance to the last point is always wrong anyway, so usually I
//just make nsteps_new small and hence make the last section too big--the curvature is zero
//at the end anyway, so it shouldn't really introduce any errors.
//Also note below that I have the ability to change nsteps_new to a smaller # if
//i becomes greater than (nsteps-2).  This is needed due to accumulated
//errors in the calculation of the arclength between the new and the old curves.
//Modified 3/95--RAB.
s = arclength(*p,nsteps);
nsteps_new = (int)(s/ds);
//Also, keep the last point from old position array so that curve doesn't
//get lost with many iterations.
nsteps_new++;  //Increment nsteps_new for array definition, then decrement it until
               //the end of routine.
lastpoint = (*p)[nsteps-1];

if(nsteps_new > 400)
   {
   printf("nsteps_new = %i   Memory problems!!!!!!!\n",nsteps_new);
```

```
   pause();
   }

//Allocate space for these arrays.  I am defining in this fashion since I am
//sometimes short of memory.  They must be deleted at the end of routine.
a = new float[nsteps];
b = new float[nsteps];
c = new float[nsteps];
rx = new float[nsteps];
ry = new float[nsteps];
temp = new vector[nsteps_new];
d2xds2 = new float[nsteps];
d2yds2 = new float[nsteps];
nsteps_new--;  //Temporarily decrement.  Have to figure the last point out
               //separately at the end of program separately.

//Solve set of equations for d2xds2 and d2yds2 found using cubic spline type set-up.
//Set endpoints to have d2xds2 = d2yds2 = 0.
a[0] = 0.;
b[0]  = 1.;
c[0] = 0.;
rx[0] = 0.;
ry[0] = 0.;
a[nsteps-1] = 0.;
b[nsteps-1] = 1.;
c[nsteps-1] = 0.;
rx[nsteps-1] = 0.;
ry[nsteps-1] = 0.;
for(j=1;j<(nsteps-1);j++)
   {
   h1 = sqrt(((*p)[j].x - (*p)[j-1].x)*((*p)[j].x - (*p)[j-1].x) +
            ((*p)[j].y - (*p)[j-1].y)*((*p)[j].y - (*p)[j-1].y));
   h2 = sqrt(((*p)[j+1].x - (*p)[j].x)*((*p)[j+1].x - (*p)[j].x) +
            ((*p)[j+1].y - (*p)[j].y)*((*p)[j+1].y - (*p)[j].y));

   a[j] = h1/6.;
   b[j] = (h1 + h2)/3.;
   c[j] = h2/6.;

   rx[j] = (*p)[j+1].x/h2 - (*p)[j].x/h2 - (*p)[j].x/h1 + (*p)[j-1].x/h1;
   ry[j] = (*p)[j+1].y/h2 - (*p)[j].y/h2 - (*p)[j].y/h1 + (*p)[j-1].y/h1;
   }
tridiag(a,b,c,rx,d2xds2,nsteps);
tridiag(a,b,c,ry,d2yds2,nsteps);

//Fit curve using cubic spline calculation.
temp[0] = (*p)[0];
i = 0;
skip = 0.;
flag = 0;
for(j=1;j<nsteps_new;j++)
   {
   //Find the correct interval to fit between and calculate what s is for that interval.
   tryagain:
   if(arclength((*p),i+2) >= (arclength(temp,j) + ds))
      {
      if(flag == 0)  //We are in the same interval as for last value of temp.
         s = ds + sqrt(((*p)[i].x - temp[j-1].x)*((*p)[i].x - temp[j-1].x) +
                      ((*p)[i].y - temp[j-1].y)*((*p)[i].y - temp[j-1].y));
      else   //We are in a different interval from last time.
         s = ds - skip;
      //Calculate the total spacing between the two points that we are going to fit.
      h = sqrt(((*p)[i].x - (*p)[i+1].x)*((*p)[i].x - (*p)[i+1].x) +
              ((*p)[i].y - (*p)[i+1].y)*((*p)[i].y - (*p)[i+1].y));
      //Reset the variables for the next iteration.
      skip = 0.;
      flag = 0;
```

```
      }
   else   //Incorrect interval.
      {
      //Calculate the value of s that is still left in the interval we are skipping out of.
      if(flag == 0)
         skip += sqrt(((*p)[i+1].x - temp[j-1].x)*((*p)[i+1].x - temp[j-1].x) +
                     ((*p)[i+1].y - temp[j-1].y)*((*p)[i+1].y - temp[j-1].y));
      else
         skip +=sqrt(((*p)[i+1].x - (*p)[i].x)*((*p)[i+1].x - (*p)[i].x) +
                    ((*p)[i+1].y - (*p)[i].y)*((*p)[i+1].y - (*p)[i].y));
      //Go to the next interval.
      i++;
      if(i > (nsteps-2))//Fixes problem.  See note at beginning of subroutine.
         {
         nsteps_new = j;
         goto quit;
         }
      //Flag = 1 indicates that we have switched intervals.
      flag = 1;
      //With the new value of i, go back and check to see if we are now in the correct interval.
      goto tryagain;
      }

   //Do the fit.
   coef1 = 1. - (s/h);
   coef2 = 1. - coef1;
   coef3 = (coef1*coef1*coef1 - coef1)*h*h/6.;
   coef4 = (coef2*coef2*coef2 - coef2)*h*h/6.;
   temp[j].x = coef1*(*p)[i].x + coef2*(*p)[i+1].x + coef3*d2xds2[i] + coef4*d2xds2[i+1];
   temp[j].y = coef1*(*p)[i].y + coef2*(*p)[i+1].y + coef3*d2yds2[i] + coef4*d2yds2[i+1];
   temp[j].label = (*p)[i+1].label;
   }

quit:
//Assign values in temp to (*p) and assign a value to the last point.
for(i=0;i<nsteps_new;i++)
   (*p)[i] = temp[i];
if((fabs((*p)[nsteps_new-1].x - lastpoint.x) < .01)&&
   (fabs((*p)[nsteps_new-1].y - lastpoint.y)< .01))
   {
   }
else
   {
   nsteps_new++;
   (*p)[nsteps_new-1] = lastpoint;
   }

//Delete all the memory that I allocated.
delete[]a;
delete[]b;
delete[]c;
delete[]rx;
delete[]ry;
delete[]temp;
delete[]d2xds2;
delete[]d2yds2;

return(nsteps_new);
}
/*******************************************************************************/
//Taken from Numerical Recipes in C, Second Edition, page 51.  This solves the
//tridiagonal problem matrix problem for u where Mu = r with a below the
//diagonal, b the diagonal and c above the diagonal.
void tridiag(float *a ,float  *b, float *c, float *r, float *u, int nsteps)
{
int j;
float bet;
```

```
float *gam=new float[nsteps];

bet = b[0];
if(bet == 0.)
   printf("Error 1 in tridiag\n");
u[0] = r[0]/bet;

for(j=1;j<nsteps;j++)
   {
   gam[j] = c[j-1]/bet;
   bet = b[j] - a[j]*gam[j];
   if(bet == 0.)
      printf("Error 2 in tridiag: j = %i\n",j);
   u[j] = (r[j] - a[j]*u[j-1])/bet;
   }

for(j=(nsteps-2);j==0;j--)
   u[j] -= gam[j+1]*u[j+1];

delete[]gam;
return;
}
/*****************************************************************************/
/*curvature at p2 where p1 is point to left, p3 to right*/
//This assumes that all points are equally spaced by some ds--which has been removed
//from the equations because of the symmetry this creates.
float curvature(vector p1, vector p2, vector p3)
{
float dxds, dyds;
float dx2ds2, dy2ds2;
float sum_sq;
float denom;
float curve;

dxds = (p3.x - p1.x)/2.;
dyds = (p3.y - p1.y)/2.;
dx2ds2 = p3.x - 2.*p2.x + p1.x;
dy2ds2 = p3.y - 2.*p2.y + p1.y;
sum_sq = (dxds)*(dxds) + (dyds)*(dyds);
denom = sum_sq * sqrt(sum_sq);

/*The sign of the curvature corresponds to the convention Mullins uses.*/
curve = -(dxds * dy2ds2 - dyds * dx2ds2)/denom;

return(curve);
}
/*****************************************************************************/
/*Calculate the normal to the curve at p2 where p1 is to the left, p3 is to the right.*/
/*ycomp = 1; This was deleted to increase speed and all values of ycomp were replaced by 1*/
vector norm(vector p1,vector p2,vector p3)
{
float xcomp,ycomp;
float m;
float normalize;
vector result;

if(p3.x == p1.x) /*Line is vertical*/
   {
   if(p1.y > p3.y)
      {
      result.x = 1.;
      result.y = 0.;
      }
   else
      {
      result.x = -1.;
      result.y = 0.;
```

```
      }
   }
else /*Line is not vertical*/
   {
   m = (p3.y - p1.y)/(p3.x - p1.x);
   if(((m > 0.)&&(p1.y > p3.y)) || ((m < 0.)&&(p1.y < p3.y)))
      {
      xcomp = m;
      ycomp = -1.;
      }
   else
      {
      xcomp = -m;
      ycomp = 1.;
      }
   normalize = sqrt(xcomp*xcomp + ycomp*ycomp);
   result.x = xcomp/normalize;
   result.y = ycomp/normalize;
   }

return(result);
}
/*****************************************************************************/
void err_message()
{
printf("usage: reflow [-T temperature(K)] [-t time_end (min)]\n");
printf("\t[-h trench height(micron)] [-w trench width (micron)] [-i filename_i]\n");
printf("\t[-o filename_o] [-s mesh size(micron)] [-j time_out (min)]\n");
printf("Default values:\n");
printf("\t T = 800K\n");
printf("\t time_end = 60 min\n");
printf("\t time_out = 10 min\n");
printf("\t ds = 0.1 microns\n");
printf("\t h = 1 micron\n");
printf("\t w = 1 micron\n");
printf("\t filename_i = profile\n");
printf("\t filename_o = reflowed\n");

return;
}
/*****************************************************************************/
void setupgraph()
{
/*request autodetection*/
int gdriver = DETECT, gmode, errorcode;

/*initialize graphics mode*/
initgraph(&gdriver, &gmode, "");

/*read result of initialization*/
errorcode = graphresult();

if (errorcode != grOk) /*an error occured*/
   {
   printf("graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);
   }
return;
}
/*****************************************************************************/
/*Make the screen width 2 hours*/
void graphtime(int xmax, int ymax, float t_sec)
{
float t_hr;
float scr_w; //Screen width in hours.
```

```
scr_w = 2.;
setcolor(LIGHTRED);

outtextxy(0,ymax-textheight("0"),"0");
outtextxy((int)(1.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(2.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(3.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(4.*xmax/8.),ymax-textheight("0"),"1");
outtextxy((int)(5.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(6.*xmax/8.),ymax-textheight("0"),"|");
outtextxy((int)(7.*xmax/8.),ymax-textheight("0"),"|");
outtextxy(xmax-textwidth("2"),ymax-textheight("0"),"2");

t_hr = t_sec/3600.;
moveto(0,ymax-2.*textheight("0"));
lineto((int)(xmax*t_hr/scr_w),ymax-2.*textheight("0"));

return;
}
/******************************************************************************/
/*Make width of graphics window correspond to 10 microns(adj. with scr_w)*/
/*h = trench height in cm, w = trench width in cm*/
/*For graphics window, lower left at (0,0); upper right at (xmax, ymax)*/
void graphtrench(int xmax, int ymax, float h, float w)
{
float onemicron, scr_w;
float x,y;

setcolor(WHITE);
scr_w = 10.;   /*scale for width of graphics window(in microns)*/
onemicron = (float)(xmax/scr_w);   /*convert 1 micron to pixel units*/

moveto(0,(int)(ymax/2));   /*start halfway up screen*/

x = onemicron*(scr_w/2. - w/2.);
y = ymax/2.;
lineto((int)x,(int)y);

y = ymax/2. + onemicron*h;
lineto((int)x,(int)y);

x = onemicron*(scr_w/2. + w/2);
lineto((int)x,(int)y);

y = (int)(ymax/2);
lineto((int)x,(int)y);

lineto(xmax, (int)(ymax/2));

return;
}
/******************************************************************************/
void graphprofile(vector *p, int nstep, int xmax, int ymax)
{
float onemicron, scr_w;
float x,y;
int j;

scr_w = 10.;
onemicron = (float)(xmax/scr_w);

clearviewport();
moveto(0,(int)(ymax/2. - onemicron*p[0].y));
for(j=1;j<nstep;j++)
   {
   if(p[j].label == 1)
```

```
      setcolor(YELLOW);
   else if (p[j].label == 2)
      setcolor(LIGHTGREEN);
   else
      setcolor(LIGHTRED);
   x = onemicron*(p[j].x - p[0].x);
   y = ymax/2. - onemicron*p[j].y;
   lineto((int)x,(int)y);
   }

return;
}
/*******************************************************************************/
void printout(vector *p, int nsteps, float t, FILE *output)
{
int i;

fprintf(output,"t=%f\n",t);
for(i=0;i<nsteps;i++)
   fprintf(output,"%7.4f   %7.4f\n",p[i].x, p[i].y);

return;
}
/*******************************************************************************/
void pause()
{
char ch;

ch = '\0';
printf("Press return to continue.....\n");
while(ch != '\n') ch = getchar();
return;
}
/*******************************************************************************/
```