Coupled Mesoscale—Continuum Simulations of Copper Electrodeposition in a Trench

Timothy O. Drews, Eric G. Webb, David L. Ma, Jay Alameda, Richard D. Braatz, and Richard C. Alkire
Dept. of Chemical and Biomolecular Engineering and National Center for Supercomputing Applications,
University of Illinois at Urbana-Champaign, Urbana, IL 61801

Copper electrodeposition in submicron trenches involves phenomena that span many orders of magnitude in time and length scales. In the present work, two codes that simulate electrochemical phenomena on different time and length scales were externally linked. A Monte Carlo code simulated surface phenomena in order to resolve surface roughness evolution and trench in-fill. A 2-D finite difference code simulated transport phenomena in the diffusion boundary layer outside the trench. The continuum code passed fluxes to the Monte Carlo code, which passed back a concentration to the continuum code. A numerical instability that arises in the multiscale linked code was suppressed by filtering the concentration data passed from the Monte Carlo code to the finite difference code. The resulting simulation results were self-consistent for a sufficiently small amount of filtering. © 2004 American Institute of Chemical Engineers *AIChE J,* 50: 226–240, 2004

Keywords: *multiscale simulations, interconnect, copper electrodeposition*

Introduction

Electrodeposition involves phenomena that are simultaneously important over a wide range of time and length scales. For example, the current and potential distribution between electrodes depends on heterogeneous and homogeneous reactions that occur among the species that move by migration, diffusion, and convection. At the same time, atomic-to-nanoscale events associated with metal nucleation and growth must be controlled to achieve the deposit properties and growth morphologies that determine product quality. While numerically stable codes are available for modeling continuum and noncontinuum phenomena, the linkage of such codes to form multiscale simulations can be numerically unstable due to temporal mismatch at the interface between the two codes. In this work, we report on a coupled simulation model that utilized a first-order filter to eliminate the numerical instability, while at the same time maintaining self-consistency.

The shift from Al to Cu for on-chip interconnections in microelectronic devices occurred because of the need for increased speed (Andricacos, 1999), reduced cost (Gau et al., 2000), reduced resistivity (Harper et al., 1999), and reduced processing temperature (Gau et al., 2000). An electrodeposition process is used to deposit copper in submicron trenches to fabricate interconnects. The electrochemical process requires simultaneous control of deposit uniformity over the wafer surface, shape evolution during trench in-fill at the submicron scale, and nucleation/morphology at the nanoscale.

The discovery of new applications where control of events at the nanoscale is critical to product quality is now driving the development of interlocking multiscale design tools that bridge both continuum and noncontinuum phenomena. Such applications are characterized by precision processing, rapid innovation, and incorporation of new knowledge at different scales. The engineering challenge associated with rapid innovation of new products is, in part, to link understanding of molecular events that control product quality with the events at larger scales that also influence processing (Alkire et al., 1998). In the case of electrodeposition of interconnects, recent fundamental advances have been made in experimental areas (Horkans et al., 2000; Moffatt et al., 2000; Kolb, 2002; Budevski, 2000) as well as continuum computational methods (Moffat et al., 2001; Merchant et al., 2000; Andricacos et al., 1998; Georgiadou et al., 2001; Gill et al., 2001; West, 2000; Soukane et al., 2002).

Correspondence concerning this article should be addressed to R. C. Alkire at r-alkire@uiuc.edu.

The manufacturability of these components that are formed by electrodeposition depends critically on precise control of interfacial conditions during processing (Datta et al., 2000).

A multiscale electrodeposition simulation was recently reported in a two-part publication (Pricer et al., 2002a,b) in which a Monte Carlo code describing noncontinuum events associated with surface roughness evolution was linked internally to a one-dimensional (1-D) continuum finite difference code for describing diffusive transport. In Part I, the Monte Carlo portion of the code simulated the molecular mechanism for copper electrodeposition in additive-free solution, and gave predictions of roughness evolution on an initially flat copper surface. In Part II, the mechanistic hypothesis was extended to include additives that follow the Kardos-Foulke blocking mechanism, and simulations were performed in a trench geometry such as used for on-chip interconnects. This work provided a "proof of concept" for the use of a noncontinuum approach for simulating additive effects on morphology evolution. The work was extended to a more complex 3-additive system (Drews et al., 2003).

External linkage of multiscale codes is desirable, since improvements can then be made in a component code at one scale without affecting the codes at other scales. External code linkage has been performed recently in a variety of applications, although the stability of the code linkage was not analyzed in these works. A linked Monte Carlo and continuum transport/reaction model of boundary-layer phenomena was used in a multiscale integration hybrid algorithm to simulate a unimolecular surface reaction (Vlachos, 1997). A multiscale metal film growth simulation was created by incorporating molecular dynamic data into a level-set model of the growing film (Hansen et al., 2000). A molecular dynamics code was coupled to a Monte Carlo code to improve feature-scale simulations of copper ionized PVD in a trench; the molecular dynamics code provided copper-ion sticking probabilities to the Monte Carlo code that were a function of the position in the trench, and the Monte Carlo code simulated the trench filling (Coronell et al., 2000). Linked codes were also used to simulate metal–oxide–semiconductor field-effect transistors (MOSFET); a Monte Carlo code calculated an equilibrium density on the device and passed that information to a finite-element continuum code that computed a potential-field distribution that was passed back to the Monte Carlo code (Hadji et al., 1999). The code sequence was run until the electron density and the potential field converged. Finally, multiscale low-pressure CVD behavior was simulated by linking a reactor-scale code, a feature-scale code, and a mesoscale code that mediated the linkage between the other codes (Gobbert et al., 1997). The reactor-scale code passed concentration, temperature, and pressure information to the mesoscale code, which resolved the information to a finer mesh and passed the information to the feature-scale code. The feature-scale code returned flux information via the mesoscale code to the reactor-scale code. For computational efficiency, the reactor-scale code ran multiple iterations before calling the other codes for updated information, a strategy that does not affect the accuracy of the simulations provided that the smaller-scale codes are called before too much change has taken place in the reactor-scale code.

In the present work, we report on implementing an external linkage of a Monte Carlo electrodeposition code with a 2-D finite difference code. Simulation results are compared to simulations performed with a 1-D continuum code linked internally to the Monte Carlo code as previously reported (Pricer et al., 2002a,b). No numerical instability is observed in the 1-D/Monte Carlo simulations, whereas numerical instability is observed in 2-D/Monte Carlo simulations, where the 2-D code is called less often due to computational constraints. The use of digital filtering to improve the numerical stability of the 2-D/Monte Carlo linked codes is described. Simulations are reported to illustrate how the amount of filtering affects the linkage between the codes, as well as to check that the linked simulation results are self-consistent for a sufficiently small amount of filtering.

Table 1. Parameters Used in the Monte Carlo and Continuum Codes in the Coupled Simulations

Model	Parameter	Value
MC	Cu^{2+} bulk-diffusion rate	6.0×10^8 nm^2/s
MC	Cu^{2+} adsorption rate	150 nm/s
MC	Cu^{2+} adsorption transfer coefficient	0.339
MC	Cu^{2+} surface-diffusion rate	2.0×10^8 nm^2/s
MC	Cu^{2+} step energy barrier	-1.5×10^{-20} J
MC	Cu^{2+} broken-face energy barrier	-5.0×10^{-22} J
MC	Cu^{2+} new-face energy barrier	5.0×10^{-22} J
MC	Cu^{2+} incorporation rate	2.0×10^4 nm^2/s
MC	Cu^{2+} incorporation transfer coefficient	-0.4
MC	Cu^{2+} incorporation transfer-coefficient contribution from Cu	0.2
CONT	H^+ bulk-diffusion rate	9.312×10^{-5} cm^2/s
CONT	SO_4^{2-} bulk-diffusion rate	1.0×10^{-5} cm^2/s
CONT	Cu^{2+} bulk-diffusion rate	0.72×10^{-5} cm^2/s

Abbreviations: MC = Monte Carlo; CONT = continuum.

Reaction Mechanism for Cu Electrodeposition

The reaction mechanism used in this article was the same as (Pricer et al., 2002a) for additive-free copper. The Cu^{2+} ions diffuse to the surface, where they react by a two-step process. The first step involves a one-electron transfer and adsorption of Cu^+ onto the surface

$$Cu^{2+} + e^- \rightarrow Cu^+ \tag{1}$$

The cuprous adions move by surface diffusion to a second location where they react to become incorporated into the crystal lattice.

$$Cu^+ + e^- \rightarrow Cu \tag{2}$$

The parameters associated with the base case (Pricer et al., 2002a) were used in the present study and are listed in Table 1. The cupric-ion adsorption-rate constant used in this study differs from that used by Pricer because the constant used here was fit to experimental data generated by a related experimental study with a similar diffusion boundary layer thickness as that simulated here (Drews et al., 2003).

Simulation Methods

This work simulates the electrodeposition of copper onto the inner surface of a submicron rectangular trench with an (additive-free) electrolytic solution containing 0.5 M $CuSO_4$ and 1

M H_2SO_4. The kinetic Monte Carlo code, used to simulate near-surface behavior, provided concentrations to the continuum code, while the continuum code, which simulated transport of the species in a diffusion layer region, passed fluxes back to the Monte Carlo code. Simulations were performed under galvanostatic (constant current) control, although the code was also able to simulate potentiostatic as well as potentiodynamic (time-dependent) control.

Monte Carlo code

The 3-D kinetic Monte Carlo code simulated mesoscale behavior in the near-surface region during deposit growth with use of cubic mesoparticles that represented clusters of molecules of various species in the deposition bath. A mesoscale simulation approach has been described elsewhere (Maroudas, 2000), and is important for simulating surface roughness evolution that cannot be accurately described by a continuum approach. The mesoparticle approach has been widely used (Birdsall and Langdon, 1985; Bird, 1994; Lu et al., 2001). In the present work, mesoparticles were assumed to be homogeneous in both phase and composition. The user specifies the species, the steps of the reaction sequence in which they participate, the parameters required for each action, and the operating variables.

The Monte Carlo simulation domain had periodic boundary conditions in the x- and y-directions, an impenetrable boundary at the bottom of the trench (in the z-direction), and a link to the continuum code at the top boundary, where the flux of mesoparticles into the domain from the continuum region was obtained as described in the next section. The initial trench aspect ratio used for the simulations reported here was 2:1. The trench width of 0.50 μm was used for the simulations presented here, and the mesoparticle size was 12.5 nm, which gave a simulation space that was 70 mesoparticles wide, 120 mesoparticles high, and 6 mesoparticles deep, within which the trench was 40 mesoparticles wide and 80 mesoparticles high. The additional height in the Monte Carlo space allowed the deposit to grow on the external surface adjacent to the trench.

At a given Monte Carlo time step, a mesoparticle can only make a maximum of one move. The moves that a mesoparticle makes are a function of the location of the particle in the simulation space and the number and type of nearest neighbors. All actions are computed as frequencies, with units of s^{-1} for homogeneity.

Bulk diffusion is used to move species through the bulk electrolyte, and it is typically the most computationally demanding part of the simulations. Bulk-diffusion coefficients are generally large compared to reaction rates, so the bulk-diffusion coefficients typically dictate the dynamics of the system, since the Monte Carlo time step is computed as 1/(frequency of an action). The movement of a particle by bulk diffusion to one of its six nearest-neighbor sites is simulated by a random-walk mechanism

$$f = \frac{6D}{L_b^2} \tag{3}$$

Surface diffusion is used to specify how a particle can maneuver along a solid surface. Surface-diffusion coefficients are typically not as high as bulk-diffusion coefficients, and particles tend to only surface diffuse for a short amount of time before they incorporate into the surface. The movement of a particle by surface diffusion is also simulated by a random-walk mechanism

$$f = \frac{6D_s e^{\Delta E/kT}}{L_b^2} \tag{4}$$

The ΔE is computed at each Monte Carlo time step as a function of the local surface morphology and makes it more facile or difficult for a particle to diffuse into one location or another depending on the value of ΔE.

The code can model adsorption with and without charge transfer. If the adsorption occurs with charge transfer, a mesoparticle of charge is consumed and the number of mesoparticles of charge consumed in the simulations is tracked and used to compute the current in the system. The reaction is a combination of Arrhenius and Tafel kinetics

$$f = \frac{6k_{rxn}e^{(\Delta E/kT)-(\alpha nF\eta/RT)}}{L_b} \tag{5}$$

The energy barrier in the reaction frequency is computed as a function of the six nearest neighbors of the particle.

The Monte Carlo time step must be small enough to capture the full dynamics of the system, so the time step is dictated by the time scale of action of the fastest species. To compute the time step, all of the possible frequencies for each species are computed. Then the inverse of each summed frequency is computed. The Monte Carlo time step t_{MC} is then selected as the smallest inverse summed frequency

$$t_{MC_j} = \frac{1}{\left(\sum_{i=1}^{k} f_i\right)_j} \tag{6}$$

$$t_{MC} = \min_j (t_{MC_j}) \tag{7}$$

In the simulations presented here the Monte Carlo time step was $\sim 4.34 \times 10^{-8}$ s, and the simulations involved $\sim 1.04 \times 10^9$ time steps. During the Monte Carlo simulation, mesoparticles are selected in the Monte Carlo domain and the simulation code looks at the "menu" of possible moves defined for the species, which is called the probability array

$$\bar{\rho}_j = t_{MC} \sum_{i=1}^{k} f_i \tag{8}$$

where $\bar{\rho}_j$ is the probability array for species j that contains a vector of probability bands. If all of the elements of $\bar{\rho}_j$ are summed, then the value of the summation will be less than 1 for all species, except the fastest reacting species in the system. If the summation is less than 1, then there is a nonzero probability of rejecting moves in the code for all species except the fastest reacting species in the system. In Eq. 8, each $f_i t_{MC}$ defines a "probability band" or bounds on an action. The values of $f_i t_{MC}$ have to be between 0 and 1. To determine what move a particle is going to make at a certain time step, the probability array is first constructed for that species. Then a random-number generator is called and it returns a random number between 0 and 1. The probability bands within which that number falls dictate the action that the particle takes at that time step. For all species other than the fastest moving species, there is a region in the probability array for which "do nothing" is an option, that is, the upper probability band is less than 1. If a random number is selected that is between this upper probability band and 1, then the particle does not do anything during the current time step. If a particle does not do anything during the current time step, then it has implicitly rejected a move. Once a move is selected, another random number is computed that tells the species to which free space it should move.

The galvanostatic simulations reported here required development of a controller to manipulate the system overpotential to obtain the desired current density (Drews, 2001). In addition, step-potential programs were used to determine the dynamics in the system and to tune the controller to compensate for the dynamics.

In a previous study, dimensionless scaling arguments were used to show that fluid convection and ohmic solution resistance are not important within submicron trenches (Takahashi et al., 1999). Moreover, concentration gradients would be expected in a trench only if the parameter

$$\xi_D = \frac{2 i_{i,0} L^2}{C_0 D w n F} \tag{9}$$

is greater than 1. For a typical trench simulation in this study (0.50 μm deep with deposition occurring at 15 mA/cm^2) the ξ_D parameter is 0.01 for cupric ions. The small value of ξ_D indicates that one would expect insignificant cupric ion depletion in the trench during electrolysis. In view of these physical expectations, and in order to reduce the computational time involved in tracking many species, the assumption was made for the Monte Carlo region to include only cupric ions in the volume of the trench. While the cupric concentration gradients are expected to be small in this investigation, it is important to retain bulk diffusion in the code for future applications, which will include consideration of dilute species for which concentration variations may be significant. The cupric ions move through the electrolyte in the Monte Carlo domain via a random-walk mechanism that simulates diffusion.

Finite difference codes

The finite difference model solved a system of coupled nonlinear partial differential equations and algebraic equations. Originally developed (Webb et al., 2002) for another application, the code was modified to address continuum transport in the diffusion layer outside the Monte Carlo region. Figure 1 shows the geometry of the continuum domain, along with the Monte Carlo region in the inset. The height of the continuum domain corresponded to the diffusion-layer thickness, which was 50 μm. A mesh of unequally spaced nodes was used. A necklace of ten nodes existed at the interface between the Monte Carlo and continuum codes. The width of the Monte Carlo-continuum boundary was set to 0.875 μm, of which 0.50 μm was the width of the trench and 0.1875 μm on either side of the trench were the trench walls.

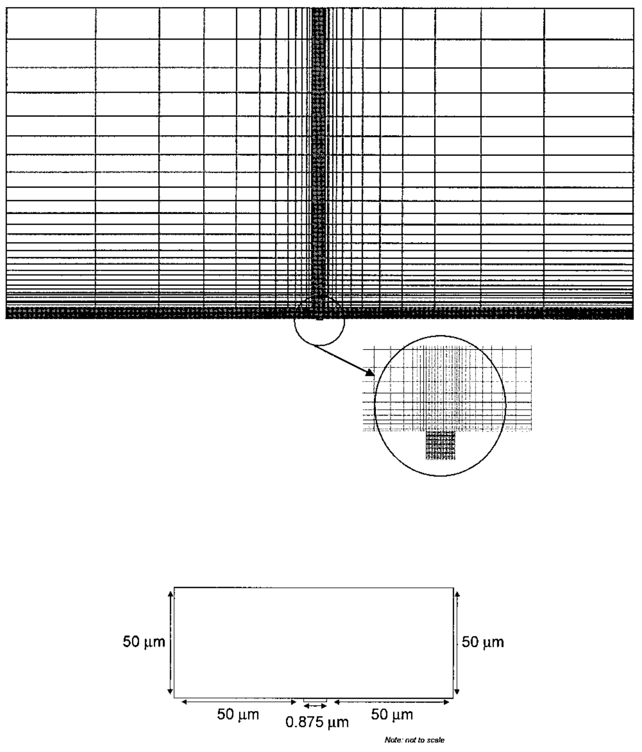


Figure 1. Mesh and size of computational domain used for solution in the finite difference PETSc code for a 0.50-μm trench with a 50-μm diffusion boundary layer (bottom diagram not to scale).

The dilute solution approximation was used to describe transport by diffusion and migration, and the solution was stagnant. The transport parameters were assumed to be independent of concentration. Elemental material balances were used for each species, i, containing the base species, k

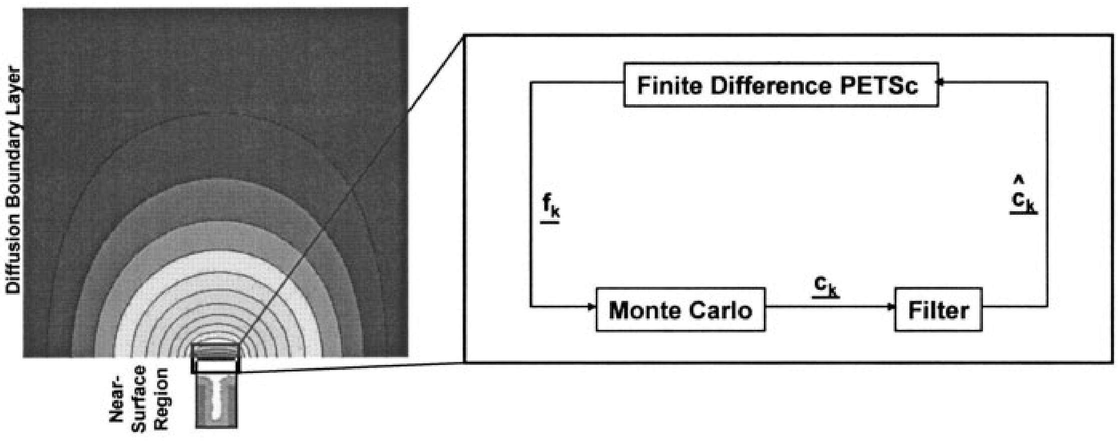
$$\sum_k \left(\frac{\partial c_i}{\partial t}\right)_k = \sum_k (-s_i^k \nabla \times N_i)_k \tag{10}$$

The equations for reaction equilibria describing relationships among the concentrations of the chemical species

$$\frac{c_i^\alpha c_j^\beta}{c_k^\delta} = K_{\text{eq}} \tag{11}$$

along with the electrical neutrality assumption

$$\sum_i z_i c_i = 0 \tag{12}$$



$\underline{f}_k$ – Vector of fluxes passed at time step k

$\underline{c}_k$ – Vector of concentrations passed at time step k

$\hat{\underline{c}}_k$ – Vector of filtered concentrations passed at time step k

Figure 2. Code linkage between the Monte Carlo code and the finite difference PETSc code.

completed the equation set. For a system involving n species, the set of Eqs. 10–12 provide $n + 1$ equations for the $n + 1$ unknowns (n species plus the solution potential).

At the upper, left, and right boundaries, the concentrations of all species were ascribed their bulk values, and zero potential with respect to the reference electrode was specified

$$c_i = c_i^\infty$$

$$\Phi = 0 \tag{13}$$

At the boundary linked to the Monte Carlo code, the fluxes of the nonreactive ions were equal to zero

$$\boldsymbol{n} \cdot \boldsymbol{N}_i = 0 \tag{14}$$

Additionally, for the boundary linked to the Monte Carlo code, the cupric ion flux was given by

$$N_i = -z_i u_i F c_i \nabla\Phi - D_i \nabla c_i \tag{15}$$

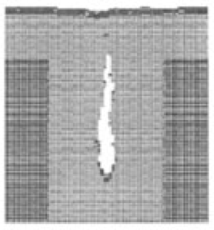
The boundary conditions for the linked boundary were completed with the addition of the equation of electroneutrality

$$\sum_i z_i c_i = 0 \tag{16}$$

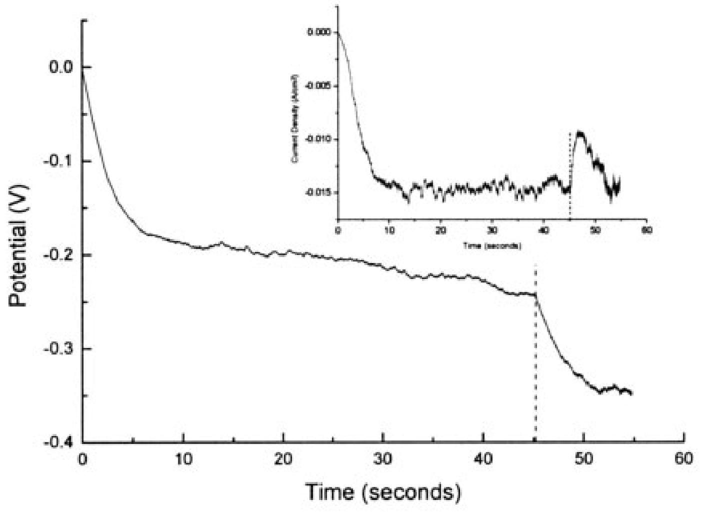
In summary, the set of equations given by Eqs. 10–12 along with the boundary conditions given by Eqs. 13–16 form a system of coupled nonlinear partial differential equations mixed with coupled nonlinear algebraic equations. The set of equations was solved simultaneously to obtain the cupric-ion flux at the boundary linked to the Monte Carlo code.

The system of coupled nonlinear partial differential equations and algebraic equations was solved numerically by the finite difference method. Second-order centered finite differences, in two-dimensions, were used to approximate the solution of the set of nonlinear partial differential equations by transforming them into a set of nonlinear algebraic equations. Second-order forward- and backward-difference approximations were used on the boundaries. The finite difference form of the partial differential equations was obtained from partial Taylor series expansion by using unequally spaced nodes, and a fully implicit scheme was implemented to step forward in time.

A line-search Newton method was used for the nonlinear set of equations where the Jacobian was hand coded. The Jacobian matrix was stored in the AIJ sparse matrix format. The resulting linear system from application of Newton's method was solved using the GMRES Krylov subspace method with ASM preconditioner. The model used PETSc (Portable, Extensible Toolkit for Scientific Computation) (http://www.mcs.anl.gov/petsc), which was developed for the parallel solution of systems of equations resulting from the discretization of partial differential equations and is particularly useful for this application, due to the large system of equations that must be solved. PETSc provided an efficient set of tools to solve the system of linear and nonlinear equations and boundary conditions, an efficient storage method to reduce memory requirements for the very large and very sparse matrix associated with the set of equations, and an efficient method to parallelize the structured grid with a distributed array. PETSc allowed for easy change of the solution method for both the nonlinear system and the resulting linear system of equations for determination of the most efficient solution technique for the set of equations. This model extended our previous 1-D finite difference model due the fact that it is a 2-D code that solves for the migration contribution to the flux. Additionally, the 2-D code is necessary to accurately simulate the effect of diffusion-limited additives in a trench, in which significant 2-D variations in the additive concentrations would be present. In the simulations presented



a.



b.

Figure 3. (a) Final trench fill image, and (b) potential-time and current density-time (inset) curves for a 0.50-μm trench at 15 mA/cm^2, with a 50-μm diffusion boundary layer; linked 2-D code simulation with no filtering at the Monte Carlo-continuum interface.

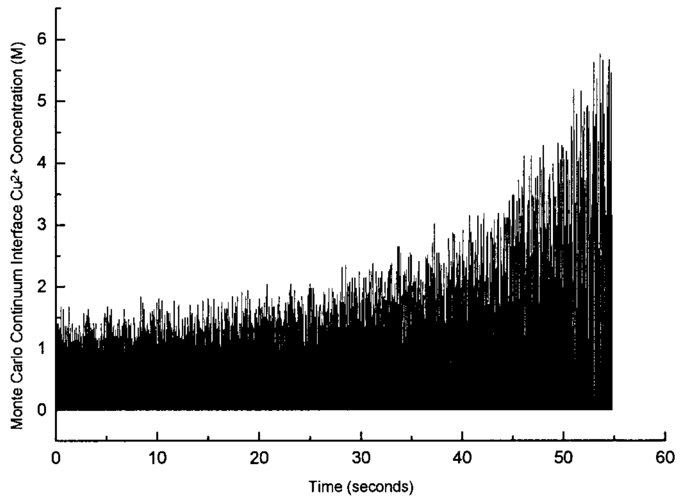
here, the finite difference code computes steady-state flux information to pass to the Monte Carlo model, although it is capable of transient simulations.

Code Linking

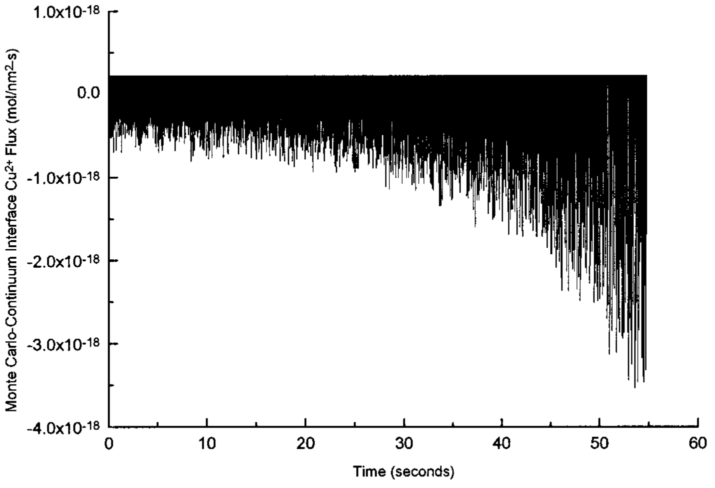
The Monte Carlo code and the finite difference code were not written explicitly to be linked to one another, but were externally linked by a series of event-handling mechanisms. Three scripts, used to instruct which code to run when, consisted of a master script that was executed to start the simulation, a script that instructed the Monte Carlo code how to operate, and a script that instructed the 2-D continuum code how to operate.

A typical simulation proceeded by the process illustrated in Figure 2. The user sets the frequency, or time length, over which the Monte Carlo code runs until it calls the external code for updated flux information. The user executes the master script and, in turn, the master script executes the script that starts the Monte Carlo code. The Monte Carlo code is the "driving code" in the process and it runs until the user-specified time length is reached, which is 0.005 s in the simulations performed here. At this point, the Monte Carlo code writes a linkage file that contains filtered concentration data at the upper boundary in the Monte Carlo domain. The script that runs the Monte Carlo code detects the existence of the linkage file and alerts that master script that the linkage file exists. The Monte Carlo code then sits in standby mode while the following actions occur.

The master script executes the script that controls the finite difference code. The finite difference code runs and uses the concentration data from the linkage file as a lower boundary condition. It then computes a steady-state flux, which is written to a different linkage file. The process then happens in reverse. The finite difference code terminates and the script that runs the finite difference code signals the master script that the flux information has been written to a linkage file. When the flux information is written, the master script signals the script that runs the Monte Carlo code that the flux has been computed. The Monte Carlo code is activated, reads in the flux information, and redistributes the flux along the top boundary of the



a.



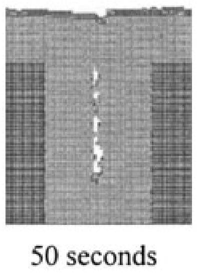
b.

Figure 4. (a) Monte Carlo–continuum interface concentrations as a function of time, and (b) Monte Carlo–continuum interface fluxes as a function of time for a 0.50-μm trench at 15 mA/cm^2, with a 50-μm diffusion boundary layer; linked 2-D code simulation with no filtering at the Monte Carlo–continuum interface.

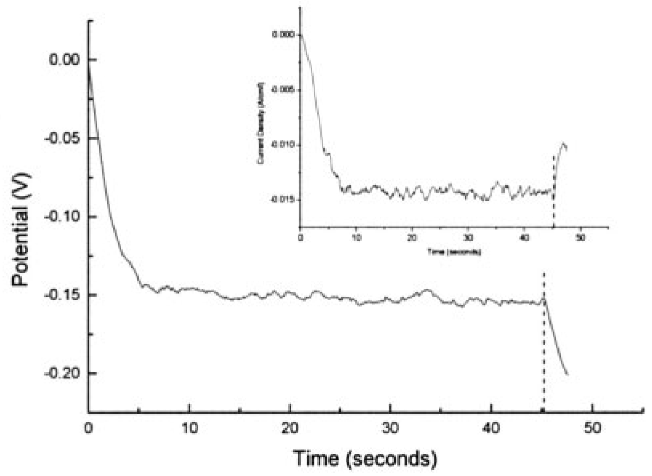
Monte Carlo domain. The flux is redistributed in the Monte Carlo domain by converting the flux into an equivalent number of mesoparticles and randomly placing them in the top layer of the Monte Carlo domain (that is, at the Monte Carlo–continuum interface). A constant flux could be employed across all nodes in the Monte Carlo domain if it were possible to vary the amount of material in a mesoparticle. The Monte Carlo code then progresses in time until the user-specified time length is reached again, at which time the entire process is repeated. When the Monte Carlo simulation is complete, a file is written by the Monte Carlo code that signals all of the scripts to terminate.

The concentrations passed to the finite difference code are computed by averaging a species' concentration over the top ten layers in the Monte Carlo domain. The fluxes that are passed to the Monte Carlo code from the finite difference code are computed by averaging the flux values computed over all ten boundary nodes. The variability in the fluxes computed over the inner eight nodes is low, while the flux computed on the outer nodes tends to vary slightly.

Simulations with the linked Monte Carlo/2-D continuum code were run on a Silicon Graphics Power Challenge with 10 R8000 CPUs at Indiana University. These simulations required approximately four days to run, with a 0.50-μm trench. Simulations of complex additive chemistries are more computationally demanding and could require several times longer to perform than the additive-free simulations presented here.



a.



b.

Figure 5. (a) Final trench fill image and (b) potential-time and current density-time (inset) curves for a 0.50-μm trench at 15 mA/cm^2, with a 50-μm diffusion boundary layer; linked code simulation with α_{filter} = 0.2 at the Monte Carlo continuum interface.

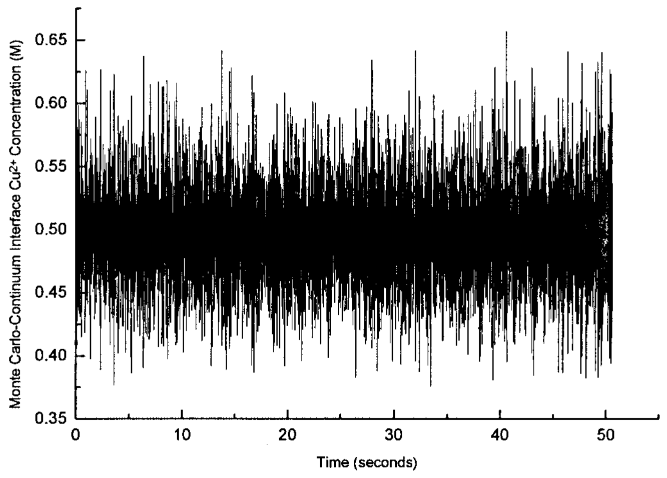
Calling the 2-D continuum code at each Monte Carlo time step would cost ($\sim$1.04 $\times$ 10^9 time steps)(20 CPU s) = $\sim$656 CPU years per simulation run, which is computationally infeasible. The frequency with which the 2-D continuum code was called was dictated by the feasibility of the total computation time required to perform a simulation. The 2-D continuum code should be called as frequently as possible, but not so often as to slow the simulations down to where they require an unreasonable amount of time to complete. Calling the 2D-continuum code for every 0.005 s of simulation time took $\sim$1.8 CPU days for each simulation run.

To gain insights into the dynamic behavior of the linked Monte Carlo/2-D simulations, simulations were also run with the 2-D code replaced by the 1-D finite difference code reported previously (Pricer et al., 2002a,b). The 1-D code was internally linked to the Monte Carlo code, that is, it was hard-coded within the Monte Carlo model. The 1-D code was called every 1 $\times$ 10^{-6} s and required less than a second ($\sim$0.004 s) to run.

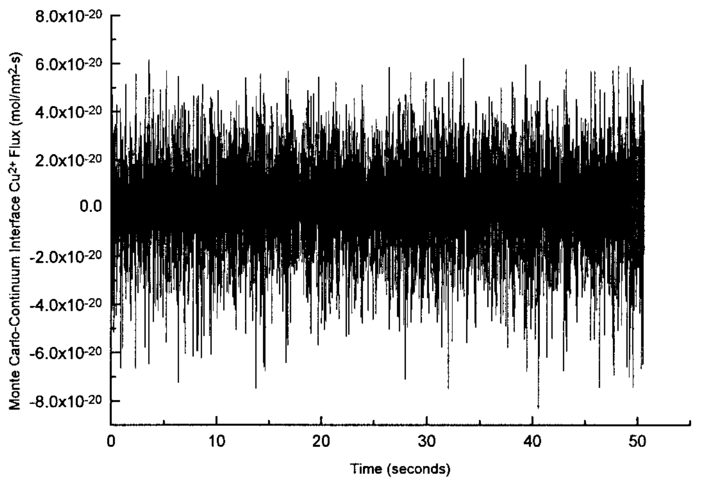
Simulations with the internally linked Monte Carlo/finite difference code were carried out on a Linux Condor pool (*http://www.cs.wisc.edu/condor*) at the National Center for Supercomputing Applications at the University of Illinois and at various Condor pools at the University of Wisconsin. Condor is an ideal system to run parameter studies that require large numbers of simulations. The Condor system seeks unused processors where it sends jobs to run. A 0.50-μm trench simulation normally required two to three days to complete on Condor.

Numerical Stability of Multiscale Linked Codes

Both the Monte Carlo code and the 2-D continuum codes were numerically stable when run as stand-alone codes. However, the linked code was numerically unstable. This numerical instability became more problematic as the 2-D continuum code was called less frequently. Calling the 2-D continuum code more frequently is not computationally feasible due to its large computational expense ($\sim$20 s). To explore the dynamic behavior of the linked codes when the continuum code is called more frequently, the 2-D continuum code was replaced by the 1-D continuum code, whose computational time is small enough that it can be run at each Monte Carlo time step. This linked Monte Carlo/1-D continuum code was numerically stable. The preceding observations suggest that the numerical instability in the linked Monte Carlo/2-D continuum code is



a.



b.

Figure 6. Monte Carlo–continuum interface concentrations as a function of time, and (b) Monte Carlo–continuum interface fluxes as a function of time for a 0.50-μm trench at 15 mA/cm^2, with a 50-μm diffusion boundary layer; linked code simulation with α_{filter} = 0.2 at the Monte Carlo–continuum interface.

due to temporal mismatch between the two codes, that is, that the input to the Monte Carlo code is updated much less often than the Monte Carlo time step. It is hypothesized that these numerical instability problems will be common between linked multiscale codes and would be expected in the case where more complex additive chemistries are simulated in the multiscale code presented here as well, since this temporal mismatch will also occur in the complex additive simulations. To suppress this numerical instability, the concentrations that are passed to the finite difference PETSc continuum code are digitally filtered, which acts to stabilize the codes. In this manner, the concentration passed from the Monte Carlo code to the continuum code contains concentration information from previous time steps. Only a fraction of the current time step information is included in the concentration. This fraction is the filter constant α_{filter}, where $0 < \alpha_{\text{filter}} < 1$ and

$$(\text{concentration passed})_i = \alpha_{\text{filter}}(\text{calculated concentration})_i + (1 - \alpha_{\text{filter}})(\text{concentration passed})_{i-1}. \quad (17)$$

This, which is a discrete-time first-order filter, will not affect the linked simulation results for α_{filter} sufficiently close to 1. In static codes, underrelaxation can be used to pass only part of the information from the current step and underrelaxation may affect the numerical stability of an algorithm, but will not affect

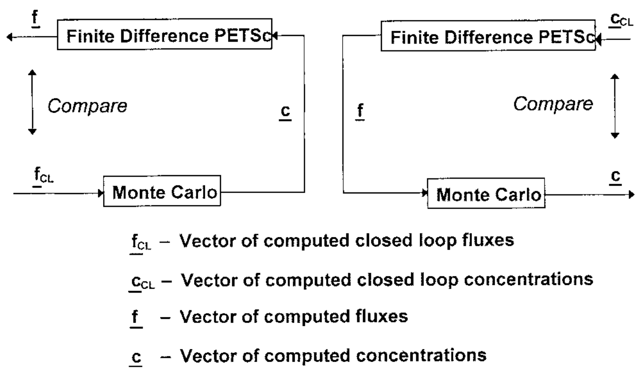

f
Finite Difference PETSc
Compare
c
f_{CL}
Monte Carlo
c_{CL}
Finite Difference PETSc
f
Compare
c
Monte Carlo
f_{CL} – Vector of computed closed loop fluxes
c_{CL} – Vector of computed closed loop concentrations
f – Vector of computed fluxes
c – Vector of computed concentrations

Figure 7. Methodology for proving self-consistency in the fluxes passed from the finite difference code to the Monte Carlo code and in the concentrations passed from the Monte Carlo code to the finite difference code.

the numerical accuracy of the results, provided that the algorithm converges. Here the term "filter" is used because this is a dynamic coupled simulation described by a dynamic filter equation that is iterated with the dynamic simulation equations, and the filter will affect the numerical accuracy of the simulations. Also, this filter was introduced based on a control system's interpretation of the dynamically coupled codes, in which a filter is introduced to suppress instabilities in a system of coupled dynamic equations. As a general rule, the code is more stable for lower α_{filter}. The filter constant should be selected to be just small enough to stabilize the linked codes.

A self-consistency check is performed, where an uncoupled simulation is run without any filtering while using the computed boundary conditions and compared to the filtered results.

Results and Discussion

This section contains the results of the simulations with the linked 2-D continuum code and the Monte Carlo code, and the linked 1-D continuum code and Monte Carlo code. The results of the simulations with the linked 2-D continuum-Monte Carlo code are provided first, followed by the simulations performed with the linked 1-D continuum-Monte Carlo code, which are used for verification. All simulations were run with galvanostatic control at -15 mA/cm^2.

Deposition experiments, reported elsewhere, were carried out with a rotating disk electrode for which the diffusion boundary layer thickness, δ_0, can be computed as a function of the rotation rate, ω

$$\delta_0 = 1.61 D_0^{1/3} \omega^{-1/2} \nu^{1/6}, \tag{18}$$

where D_0 is the diffusion coefficient and ν is the kinematic viscosity. The diffusion boundary-layer thickness did not have a significant effect on the simulations presented here because concentration gradients in the system were small (i.e., $\xi_D = 0.01$). In the system under investigation, a diffusion boundary layer thickness of 50 μm was used as the base case, which corresponds to a rotation rate of about 100 rpm.

Externally linked 2-D finite difference code

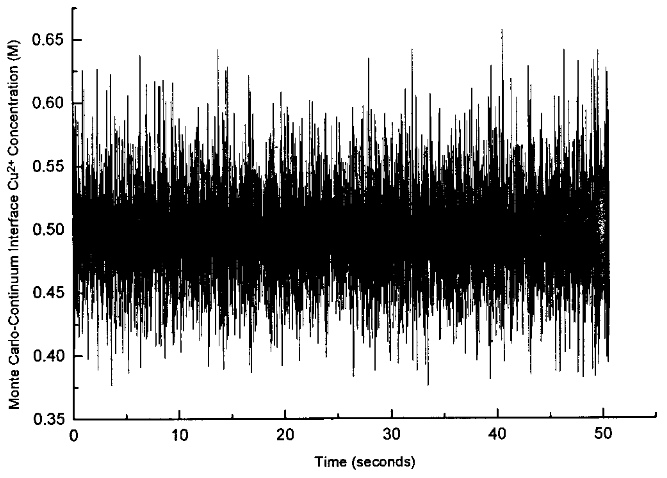
Trenches of varying diffusion boundary-layer thickness were simulated with various filter constant values to examine how the filter constant value affects the accuracy. The simulation results without filtering, corresponding to a filter constant value of 1, for the externally linked codes are shown in Figure 3. In Figure 3a, it may be seen that the trench fills in a conformal manner. Figure 3b shows that the potential never really settles to a steady-state potential; it is continually pushed more negative over time.

Numerical instability in the linked code is illustrated in Figure 4, which shows the concentrations passed to the continuum code by the Monte Carlo code and the fluxes passed to the Monte Carlo code by the continuum code. As time progresses in the simulations, the code linkage becomes numerically unstable and these quantities increase exponentially. The concentrations passed have lower bounds of zero. We suggest that such numerical instabilities are common when dynamically linking multiscale codes, that is, codes of widely varying time and length scales. Fluxes passed to the Monte Carlo code blow up over time as the concentrations blow up over time; the fluxes are computed as a function of the concentration information passed to the continuum code.

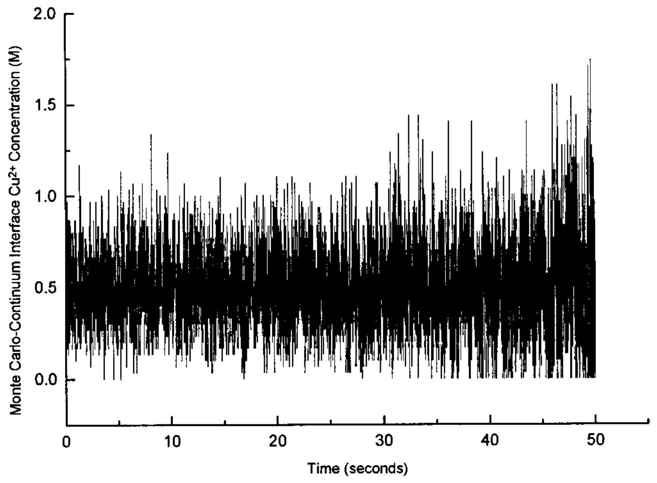
Simulations were performed for various filter constant values. The results from a simulation performed with a filter constant of 0.2 are presented in Figure 5. This value of the filter constant was selected because no numerical instability was observed in the code linkage and the results will be later shown to be self-consistent. A diffusion boundary layer thickness of 50 μm was used. Simulations with a filter constant of 0.1 and 0.2 are very similar (Drews, 2001), meaning that this small change in filter constant does not significantly affect the simulation results. The trench fills in the same manner, with a similar shaped void for a filter constant value of 0.1 and 0.2. The steady-state potentials are very similar, with steady state being attained after ~5 s of simulation. In Figure 6, the effect of the filter on the concentrations and fluxes passed to the continuum code is shown. Increasing the filter constant from 0.1 to 0.2 results in a higher amplitude of noise in the passed quantities (Drews, 2001).

Increasing the filter constant to 0.50 and 0.75 resulted in numerical instability that was very similar, but not as severe, to that observed when the passed concentrations were not filtered (Drews, 2001). The potential is maintained at a steady-state value for most of the simulation, but begins to emulate the trend seen in the unfiltered potential-time curve after ~35 s of simulation time. That is, the system is continuously pushed more cathodic until the trench pinches off. This suggests that the trends seen in the potential-time curves are in response to the code linkage instability.

Self-Consistency Checks. When quantities are filtered in simulations, it is possible that the filtering can affect the physical results predicted by the simulations. In the case of the linked Monte Carlo–finite difference codes, the concentrations passed from the Monte Carlo code to the finite difference code are filtered, which suppresses the numerical instability in the code linkage, provided that the amount of filtering is large enough. It was shown previously that if the amount of filtering was not high enough (i.e., the filter constant was small), then the numerical instability in the linkage would still be encountered, albeit noticeable at later times in the simulation.



a.



b.

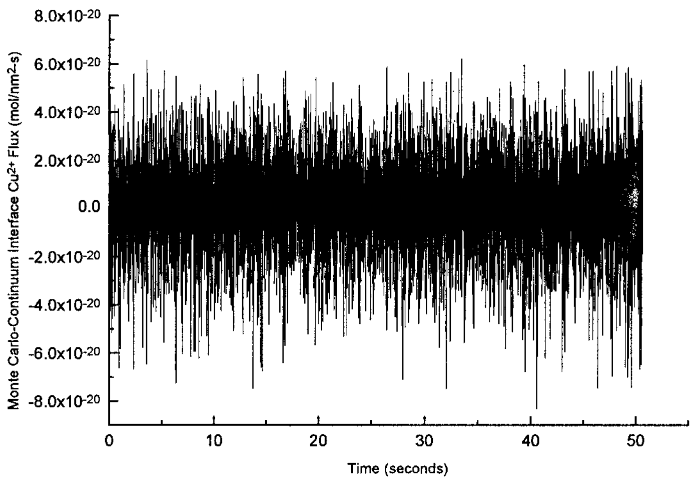
Figure 8. (a) Concentrations passed from the Monte Carlo code with filtering (closed-loop), and (b) concentrations passed from the Monte Carlo code (open-loop).

Concentration self-consistency in the concentrations passed from the Monte Carlo code to the finite difference code. In Figure 8b, some of the concentrations after 35 s of deposition attain the minimum allowed value of 0 M.

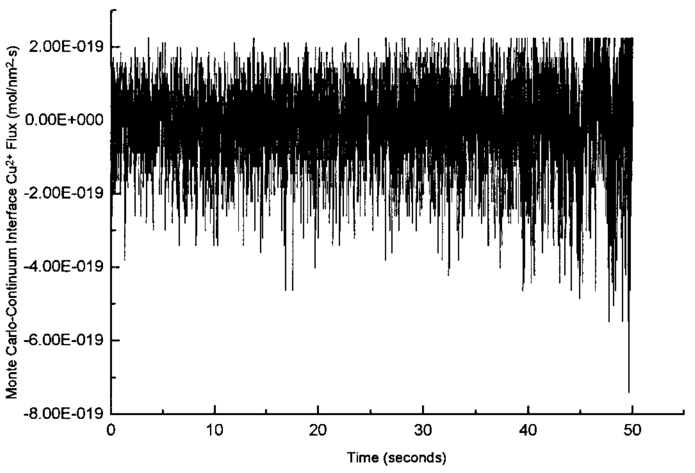
The desired amount of filtering is such that the filtering is large enough to eliminate the numerical instability in the code linkage, but small enough that accurate physical outcomes are simulated. Previously it was shown that by monitoring the information passed between the codes, one can determine whether the numerical instability arises in the code linkage. A method to assess the potential effects of the filtering on the accuracy of the simulation results is to perform self-consistency simulations. The methodology with which the self-consistency simulations were performed with these codes is shown in Figure 7.

The filtered concentrations and fluxes that are exchanged between the Monte Carlo code and finite difference code are recorded at all times during the simulation where the codes are linked together. Then the code linkage is broken so that the codes can be run independent of the other code, but take as input the output from the other code.

For the self-consistency check of the fluxes (the self-consistency check for the concentrations is performed in the exact same manner), the closed loop (note that the terms “closed loop” and “open loop” are used in the control sense) fluxes are fed into the Monte Carlo code and a simulation is performed with these fluxes. The concentrations that would be passed to the finite difference code are recorded at each



a.



b.

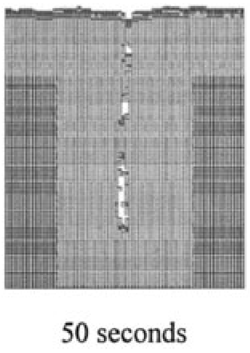
Figure 9. (a) Fluxes passed from the finite difference code with filtering (closed-loop), and (b) fluxes passed from the finite difference code (open-loop).

Flux self-consistency in the fluxes passed from the finite difference code to the Monte Carlo code. In Figure 9b, some of the fluxes after 35 s of deposition attain the maximum possible flux, which corresponds to a boundary condition of 0 M Cu^{2+} passed from the Monte Carlo code.

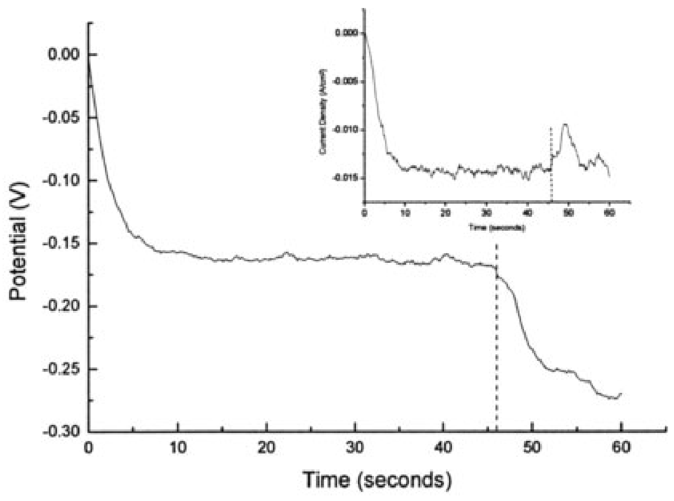
time step that a new flux is read into the Monte Carlo code. These concentrations are then fed into the finite difference code without being filtered, and the fluxes that are computed by the finite difference code are recorded and compared to the closed-loop fluxes initially fed into the Monte Carlo code. If the fluxes that are computed from the open-loop simulations just described do not drift away from the values of the fluxes computed from the closed-loop simulation, then the fluxes are self-consistent. The fluxes computed from the open-loop simulations will be noisier than the closed-loop fluxes because the noise in the simulations is compounded in the Monte Carlo and then the finite difference simulation. The results from the concentration and flux self-consistency checks are shown in Figures 8 and 9, respectively.

1-D internal continuum code

The same simulations that were run with the 2-D continuum code were run with the 1-D continuum code. For a given trench width, the trenches all pinched off at almost the identical time, regardless of the diffusion boundary-layer thickness used. The



a.



b.

Figure 10. (a) Final trench fill image, and (b) potential-time and current-density-time (inset) curves for a 0.50-μm trench at -15 mA/cm^2, with a 50-μm diffusion boundary layer; continuum code is the 1-D finite difference code.

potential-time curves are almost identical for varying diffusion-layer thickness, which is similar to the experimentally observed results observed on flat-surface rotating-disk electrode experiments. Furthermore, the continuum diffusion profiles showed that the bulk concentration was maintained at the upper continuum boundary. However, this concentration was not maintained just at the boundary; often the bulk concentration was seen much further inside the diffusion layer. Additionally, the time step for calling the continuum code had to be made very small in order to maintain numerical stability in the continuum code.

Trench fill images for the three diffusion boundary-layer thicknesses show that the trenches fill almost identically, in a conformal manner, regardless of the diffusion boundary-layer thickness. The results for the simulation with a 50-μm diffusion boundary layer thickness are shown in Figure 10. The potential-time and current density-time curves show that the trenches pinch off between about 44 and 46 s for all three diffusion boundary-layer thicknesses. The steady-state potential is about the same for all three diffusion boundary-layer thicknesses. In Figure 11, the trench void fraction evolution is shown to be linear as a function of time for all three trenches, and the final void fraction is almost identical for all three trenches.

These simulations show that the 1-D model can predict the experimentally expected results of changing the diffusion boundary-layer thickness. That is, the trenches pinch off at the identical time and the steady-state potential is nearly identical for a given trench width, regardless of the diffusion boundary-layer thickness. The parameters used for the simulation shown in Figure 10 are identical to those used in the simulation shown in Figure 3. Comparison of Figures 3 and 10 show that the void in Figure 10 is much smaller than the void in Figure 3. The potentials at which the system is pushed to maintain the set point current density with the 2-D code are much more cathodic than with the 1-D code. Based on past work (Pricer et al., 2002a), it is natural to expect that since the system is run at a higher potential, the surface will be rougher, making it more facile to pinch-off the trench, with a higher void space that results from higher cupric-ion adsorption at the mouth of the trench. The trench simulated with the 2-D code pinches off slightly faster than the trench simulated with the 1-D code.

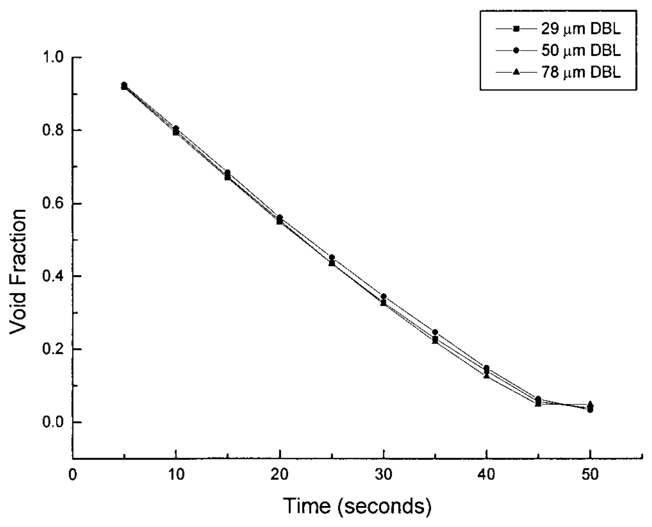


Figure 11. Trench void fraction as a function of time for a 0.50-μm trench at −15 mA/cm^2, with 29-μm, 50-μm, and 78-μm diffusion boundary layers (DBL); continuum code is the 1-D finite difference code.

Conclusions

Coupled Monte Carlo–continuum simulations of trench filling in a copper electrodeposition process are shown to be similar whether the continuum code is a 2-D externally linked code, or a 1-D internally linked code. While the simulations are similar, the two-dimensional code provides the ability to precisely control the diffusion boundary-layer thickness and, therefore, maintain the top boundary condition on the continuum domain precisely. Furthermore, the 2-D externally linked code is necessary to accurately simulate the effect of diffusion-limited additives in a trench. Numerical instability incurred between linked codes was corrected with the use of a first-order filter. The amount of concentration filtering is determined such that the filtering eliminates the numerical instability, while at the same time maintaining self-consistency. The filtering approach is applicable to other linked codes in which one code is updated more slowly than the others. Also, in cases of numerical instability induced by a mismatch of length scales, spatial filtering should be considered as an approach to suppressing the numerical instability. In any case, a self-consistency check should be performed to assess the effect of the filtering on the multiscale linked simulations.

One approach that could help to eliminate some of the numerical instability issues encountered here would be to couple the codes such that the domains of the Monte Carlo and continuum codes overlap (Broughton et al., 1999; Aktas and Aluru, 2002). A drawback of this approach is that it would be more difficult to keep the codes that simulate different time and length scales separate and modular as they are in this work. This modularity is important if multiscale simulation is going to become a dominant paradigm in chemical engineering applications.

Acknowledgments

Funding for this project was received from the National Science Foundation via the National Computational Science Alliance (NSF ACI 96-19019) as well as NSF grant CTS-0135621, and an IBM Faculty Partnership. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Computer time was provided through the National Science Foundation Partnership for Advanced Computing Infrastructure via the National Computational Science Alliance (NCSA, University of Illinois), and utilized the Condor system at the University of Wisconsin. The authors thank Professor Randall Bramley, Professor Dennis Gannon, and the Extreme! Computing Group at Indiana University for their assistance in the code linkage and for allowing us to perform many simulations on their workstations.

Notation

c =	vector of computed concentrations, M
c_{CL} =	vector of computed closed-loop concentrations, M
c_i =	concentration of species I, M
c_k =	vector of concentrations passed at time step k, M
$\hat{c}_k$ =	vector of filtered concentrations at time step k, M
C_0 =	bulk cupric ion concentration, M
D =	cupric ion diffusion coefficient in bulk electrolyte, cm^2/s
D_s =	cupric ion surface diffusion coefficient, [nm^2/s]
E =	energy (barrier), J/molecule
f =	vector of computed fluxes, mol/nm^2-s
f_{CL} =	vector of computed closed-loop fluxes, mol/nm^2-s
f_i =	frequency of action i, [s^{-1}]
f_k =	vector of fluxes passed at time step k, mol/nm^2-s
F =	Faraday's constant, C/equivalent
$i_{i,0}$ =	nominal plating current density, A/cm^2
k =	Boltzmann's constant, J/K
k_{rxn} =	reaction-rate constant, nm/s
K_{eg} =	equilibrium constant for homogeneous reaction, M or M^2
L =	trench depth, cm
L_b =	mesoparticle length, nm
n =	number of charge equivalents, equivalents
N_i =	Flux of species i, mol/(cm^2-s)
R =	ideal gas constant, J/(mol-K)
s_i^k =	stoichiometry of species I in reaction k
t =	deposition time, s
t_{MC} =	Monte Carlo time step, s
T =	temperature, K
w =	trench width, cm
z_i =	ionic charge of species i

Greek letters

α_{filter} =	filter constant
α_T =	charge-transfer coefficient
δ_D =	diffusion boundary-layer thickness, μm
Φ =	solution-phase potential, mV
η =	overpotential, V
ν =	kinematic viscosity, cm^2/s
$\bar{\rho}_j$ =	probability array
ω =	RDE rotation rate, rad/s
ξ_D =	dimensionless diffusion parameter

Literature Cited

Aktas, O., and N. R. Aluru, "A Combined Continuum/DSMC Technique for Multiscale Analysis of Microfluidic Filters," *J. Comput. Phys.,* **178,** 342 (2002).

Alkire, R., and M. Verhoff, "The Bridge from Nanoscale Phenomena to Macroscale Processes," *Electrochim. Acta,* **43,** 2733 (1998).

Andricacos, P. C., C. Uzoh, J. O. Dukovic, J. Horkans, and H. Deligianni, "Damascene Copper Electroplating for Chip Interconnections," *IBM J. Res. Dev.,* **42,** 567 (1998).

Andricacos, P. C., "Copper On-Chip Interconnections—A Breakthrough in Electrodeposition to Make Better Chips," *Electrochem. Soc. Interface,* **8,** 32 (1999).

Bird, G. A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows,* Clarendon Press, Oxford (1994).

Birdsall, C. K., and A. B. Langdon, *Plasma Physics via Computer Simulation,* McGraw-Hill, New York (1985).

Broughton, J. Q., F. F. Abraham, N. Bernstein, and E. Kaxiras, “Concurrent Coupling of Length Scales: Methodology and Application,” *Phys. Rev. B,* **60,** 2391 (1999).

Budevski, E., G. Staikov, and W. J. Lorenz, “Electrocrystallization Nucleation and Growth Phenomena,” *Electrochim. Acta,* **45,** 2559 (2000).

Coronell, D. G., D. E. Hansen, A. F. Voter, C. Liu, X. Liu, and J. D. Kress, “Molecular Dynamics-Based Ion-Surface Interaction Modes for Ionized Physical Vapor Deposition Feature Scale Simulations,” *Appl. Phys. Lett.,* **73,** 3860 (2000).

Datta, M., and D. Landolt, “Fundamental Aspects and Applications of Electrochemical Microfabrication,” *Electrochim. Acta,* **45,** 2535 (2000).

Drews, T. O., “Multi-Scale Simulations of Copper Electrodeposition from a Multiple Additive System,” M. S. Thesis, Univ. of Illinois at Urbana-Champaign (2001).

Drews, T. O., J. C. Ganley, and R. C. Alkire, “Evolution of Surface Roughness During Copper Electrodeposition in the Presence of Additives: Comparison of Experiments and Monte Carlo Simulations,” *J. Electrochem. Soc.,* **150,** C325 (2003).

Gau, W. C., T. C. Chang, Y. S. Lin, J. C. Hu, L. J. Chen, C. Y. Chang, and C. L. Cheng, “Copper Electroplating for Future Ultralarge Scale Integration Interconnection,” *J. Vac. Sci. Technol. A,* **18,** 656 (2000).

Georgiadou, M., D. Veyret, R. L. Sani, and R. C. Alkire, “Simulation of Shape Evolution During Electrodeposition of Copper in the Presence of Additive,” *J. Electrochem. Soc.,* **148,** C54 (2001).

Gill, W. N., D. J. Duquette, and D. Varadarajan, “Mass Transfer Models for the Electrodeposition of Copper with a Buffering Agent,” *J. Electrochem. Soc.,* **148,** C289 (2001).

Gobbert, M. K., T. P. Merchant, L. J. Borucki, and T. S. Cale, “A Multiscale Simulator for Low Pressure Chemical Vapor Deposition,” *J. Electrochem. Soc.,* **144,** 3945 (1997).

Hadji, D., Y. Marechal, and J. Zimmerman, “Finite Element and Monte Carlo Simulation of Submicrometer Silicon n-MOSFETs,” *IEEE Trans. Magnetics,* **35,** 1809 (1999).

Hansen, U., S. Rodgers, and K. F. Jensen, “Modeling of Metal Thin Film Growth: Linking Angstrom-Scale Molecular Dynamics Results to Micron-Scale Film Topographies,” *Phys. Rev. B,* **62,** 2869 (2000).

Harper, J. M. E., C. Cabral, Jr., P. C. Andricacos, L. Gignac, I. C. Noyan, K. P. Rodbell, and C. K. Hu, “Mechanisms for Microstructure Evolution in Electroplated Copper Thin Films Near Room Temperature,” *J. Appl. Phys.,* **86,** 2516 (1999).

Horkans, J., C. Cabral, Jr., K. P. Rodbell, C. Parks, M. A. Gribelyuk, S. Malhotra, and P. C. Andricacos, “Trends in Properties of Electroplated Cu with Plating Conditions and Chemistry,” *Electrochemistry Proc. ULSI Fabrication III,* Electrochemical Society, Pennington, NJ (2000).

Kolb, D. M., “The Initial Stages of Metal Deposition as Viewed by Scanning Tunneling Microscopy,” Advances in Electrochemical Science and Engineering, Vol. 7, Wiley, Weinheim (2002).

Lu, J., and M. J. Kushner, “Trench Filling by Ionized Metal Physical Vapor Deposition,” *J. Vac. Sci. Technol. A,* **19,** 2652 (2001).

Maroudas, D., “Multiscale Modeling of Hard Materials: Challenges and Opportunities for Chemical Engineering,” *AIChE J.,* **46,** 878 (2000).

Merchant, T. P., M. K. Gobbert, T. S. Cale, and L. J. Borucki, “Multiple Scale Integrated Modeling of Deposition Processes,” *Thin Solid Films,* **365,** 368 (2000).

Moffat, T. P., J. E. Bonevich, W. H. Huber, A. Stanishevsky, D. R. Kelly, G. R. Stafford, and D. Josell, “Superconformal Electrode-position of Copper in 500-90 nm Features,” *J. Electrochem. Soc.,* **147,** 4524 (2000).

Moffatt, T. P., D. Wheeler, W. H. Huber, and D. Josell, “Superconformal Electrodeposition of Copper,” *Electrochem. and Solid State Lett.,* **4,** C26 (2001).

Pricer, T. J., M. J. Kushner, and R. C. Alkire, “Monte Carlo Simulation of the Electrodeposition of Copper—Part I: Additive Free Acidic Sulfate Solution,” *J. Electrochem. Soc.,* **149,** C396 (2002a).

Pricer, T. J., M. J. Kushner, and R. C. Alkire, “Monte Carlo Simulation of the Electrodeposition of Copper—Part II: Acid Sulfate Solution with Blocking Additive,” *J. Electrochem. Soc.,* **149,** C406 (2002b).

Soukane, S., S. Sen, and T. S. Cale, “Feature Superfilling in Copper Electrochemical Deposition,” *J. Electrochem. Soc.,* **149,** C74 (2002).

Takahashi, K. M., and M. E. Gross, “Transport Phenomena that Control Electroplated Copper Filling of Submicron Vias and Trenches,” *J. Electrochem. Soc.,* **146,** 4499 (1999).

Webb, E. G., and R. C. Alkire, “Pit Initiation at Single Sulfide Inclusions in Stainless Steel—Part III: Mathematical Model,” *J. Electrochem. Soc.,* **149,** B286 (2002).

West, A. C., “Theory of Filling of High-Aspect Ratio Trenches and Vias in Presence of Additives,” *J. Electrochem. Soc.,* **147,** 227 (2000).

Vlachos, D. G., “Multiscale Integrations Hybrid Algorithms for Homogeneous-Heterogeneous Reactors,” *AIChE J.,* **43,** 3031 (1997).

Manuscript received Sept. 26, 2002, and revision received May 29, 2003.