IOPscience iopscience.iop.org

Home Search Collections Journals About Contact us My IOPscience

Parallel kinetic Monte Carlo simulations of Ag(111) island coarsening using a large database

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

2009 J. Phys.: Condens. Matter 21 084214

(http://iopscience.iop.org/0953-8984/21/8/084214)

View the table of contents for this issue, or go to the journal homepage for more

Download details:
IP Address: 129.252.86.83
The article was downloaded on 29/05/2010 at 17:58

Please note that terms and conditions apply.

J. Phys.: Condens. Matter **21** (2009) 084214 (12pp) doi:10.1088/0953-8984/21/8/084214

Parallel kinetic Monte Carlo simulations of Ag(111) island coarsening using a large database

Giridhar Nandipati[1], Yunsic Shim[1], Jacques G Amar[1], Altaf Karim[2], Abdelkader Kara[3], Talat S Rahman[3] and Oleg Trushin[4]

[1] Department of Physics and Astronomy, University of Toledo, Toledo, OH 43606, USA
[2] Brookhaven National Laboratory, Upton, NY 11973, USA
[3] Department of Physics and Astronomy, University of Central Florida, Orlando, FL 32816, USA
[4] Institute of Microelectronics and Informatics, Academy of Sciences of Russia, Yaroslavl 150007, Russia

E-mail: jamar@physics.utoledo.edu

Received 18 July 2008
Published 30 January 2009
Online at stacks.iop.org/JPhysCM/21/084214

Abstract

The results of parallel kinetic Monte Carlo (KMC) simulations of the room-temperature coarsening of Ag(111) islands carried out using a very large database obtained via self-learning KMC simulations are presented. Our results indicate that, while cluster diffusion and coalescence play an important role for small clusters and at very early times, at late time the coarsening proceeds via Ostwald ripening, i.e. large clusters grow while small clusters evaporate. In addition, an asymptotic analysis of our results for the average island size $S(t)$ as a function of time t leads to a coarsening exponent $n = 1/3$ (where $S(t) \sim t^{2n}$), in good agreement with theoretical predictions. However, by comparing with simulations without concerted (multi-atom) moves, we also find that the inclusion of such moves significantly increases the average island size. Somewhat surprisingly we also find that, while the average island size increases during coarsening, the scaled island-size distribution does not change significantly. Our simulations were carried out both as a test of, and as an application of, a variety of different algorithms for parallel kinetic Monte Carlo including the recently developed optimistic synchronous relaxation (OSR) algorithm as well as the semi-rigorous synchronous sublattice (SL) algorithm. A variation of the OSR algorithm corresponding to optimistic synchronous relaxation with pseudo-rollback (OSRPR) is also proposed along with a method for improving the parallel efficiency and reducing the number of boundary events via dynamic boundary allocation (DBA). A variety of other methods for enhancing the efficiency of our simulations are also discussed. We note that, because of the relatively high temperature of our simulations, as well as the large range of energy barriers (ranging from 0.05 to 0.8 eV), developing an efficient algorithm for parallel KMC and/or SLKMC simulations is particularly challenging. However, by using DBA to minimize the number of boundary events, we have achieved significantly improved parallel efficiencies for the OSRPR and SL algorithms. Finally, we note that, among the three parallel algorithms which we have tested here, the semi-rigorous SL algorithm with DBA led to the highest parallel efficiencies. As a result, we have obtained reasonable parallel efficiencies in our simulations of room-temperature Ag(111) island coarsening for a small number of processors (e.g. $N_p = 2$ and 4). Since the SL algorithm scales with system size for fixed processor size, we expect that comparable and/or even larger parallel efficiencies should be possible for parallel KMC and/or SLKMC simulations of larger systems with larger numbers of processors.

(Some figures in this article are in colour only in the electronic version)

 © 2009 IOP Publishing Ltd Printed in the UK

1. Introduction

Kinetic Monte Carlo (KMC) is an extremely efficient method [1–6] to carry out dynamical simulations of non-equilibrium processes when the relevant activated atomic-scale processes are known. Accordingly, KMC simulations have been successfully used to model a variety of dynamical processes ranging from catalysis to thin-film growth. However, in many cases it is difficult to calculate *a priori* all the possible atomic-scale events which may be important in a simulation, and so typically KMC simulations have relied on a relatively limited rate catalog.

To address this problem, Rahman and coworkers have recently developed a self-learning kinetic Monte Carlo (SLKMC) method [7, 8]. In an SLKMC simulation, rather than use a fixed catalog of processes and their corresponding activation barriers, the activation barriers corresponding to new configurations not already included in the database are obtained on the fly. The resulting processes are then added to the SLKMC database. By only determining events for configurations which actually occur during the simulation, and also using the symmetry of the lattice to reduce the number of possible distinct configurations, this method may be used to study a variety of different problems. As an example, in recent SLKMC simulations [8] of Cu(111) island diffusion it was found that, due in part to the inclusion of concerted moves which play an important role for small islands, the dependence of the diffusion constant on cluster size was quite different from that obtained in previous simulations with a limited rate catalog.

While KMC and SLKMC simulations can be effective tools to carry out atomistic simulations of non-equilibrium processes, often it is desirable to carry out simulations over extended timescales and length scales. As a result, a number of different algorithms for parallel KMC have recently been studied and applied to simulations of thin-film growth. These include the synchronous relaxation (SR) algorithm [9–11], the optimistic synchronous relaxation (OSR) algorithm [12] and the semi-rigorous synchronous sublattice (SL) algorithm [13]. In particular, it has recently been shown that the OSR algorithm can provide a reasonable parallel efficiency in low-temperature simulations of submonolayer nucleation and growth on metal (111) surfaces. The semi-rigorous SL algorithm [13] has also been shown to provide good parallel efficiency in simulations of a variety of simplified models of nucleation and thin-film growth. Due to the fact that it only requires local communications, the SL algorithm also scales with system size, e.g. for a fixed processor size the simulation time is independent of the number of processors.

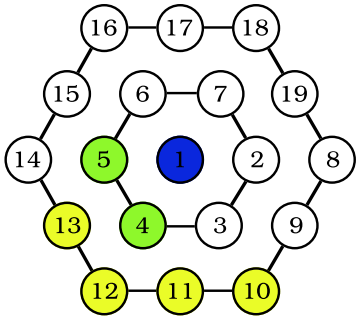
As a test of the applicability of such parallel algorithms to simulations over extended timescales and length scales at higher temperatures, here we present the results of kinetic Monte Carlo simulations of the room-temperature coarsening of Ag(111) clusters carried out using a large database obtained via SLKMC simulations. As discussed in more detail below, our results indicate that at late time the coarsening proceeds via Ostwald ripening. We also find that the inclusion of concerted (multi-atom) moves in our database has a significant effect on the asymptotic coarsening. Our results also indicate that the scaled island-size distribution does not change significantly during coarsening.

We note that this work is part of a larger project to develop parallel algorithms to carry out realistic SLKMC simulations over larger timescales and length scales. Therefore, in addition to presenting results for the coarsening of Ag(111) islands we also present results for the efficiency and accuracy of a number of different algorithms applied to Ag(111) island coarsening. In particular, we discuss the optimistic synchronous relaxation (OSR) algorithm, the optimistic synchronous relaxation with pseudo-rollback (OSRPR) algorithm, the synchronous sublattice (SL) algorithm, as well as a number of improvements such as dynamic boundary allocation (DBA). We note that, because of the very large range of event rates in these simulations (ranging from 4×10^3 s^{-1} for double-bond detachment from an island to 8×10^{10} s^{-1} for monomer hopping), developing an efficient algorithm for parallel KMC and/or SLKMC simulations is particularly challenging. However, our results indicate that, by using the SL algorithm along with dynamic boundary allocation to increase the cycle time and thus reduce communication overhead, a reasonable parallel efficiency can be achieved.

The organization of this paper is as follows. In section 2, we first describe the database used in our simulations as well as how it was obtained via SLKMC simulations. We also describe minor modifications that we have made to the closed database to improve the speed of our simulations. In section 3, we describe the binary tree algorithm which was used to maximize the efficiency of our KMC simulations. In section 4 we describe in detail the OSR, OSRPR and SL algorithms used in our parallel KMC simulations along with the dynamic boundary allocation (DBA) technique. Our results for the parallel efficiencies of these three algorithms—as obtained in simulations of room-temperature Ag(111) island coarsening—are presented in section 5.1. Detailed results for the evolution of the average island size and the scaled island-size distribution are then presented in section 5.2. Finally, in section 6, we summarize our results.

2. KMC database

Our database was obtained from self-learning kinetic Monte Carlo simulations of Ag(111) island and cluster motion carried out at 300 and 500 K and consists of two parts. The first part is a large cluster database which was obtained from SLKMC simulations of islands of 19 atoms and larger, i.e. clusters typically consisting of a central atom and at least two filled rings. For these large islands, we have found that, for homoepitaxy, cluster diffusion typically occurs via a series of single or multi-atom moves of edge atoms from fcc sites to fcc sites. Accordingly, in this case the drag method was used



Shell	Base 2	Base 10
1	001100	12
2	000000111100	60

Figure 1. Two-shell indexing around the central atom labeled 1.

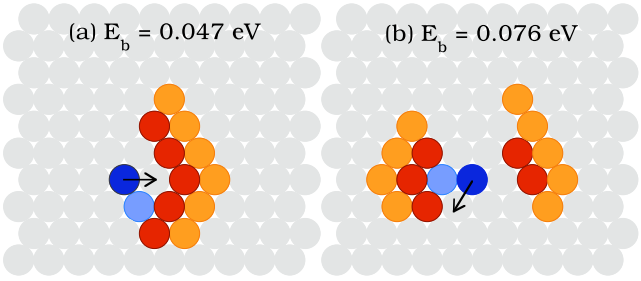


Figure 2. Two typical low-barrier moves in the large cluster database along with the corresponding activation barriers E_b. The dark blue atom corresponds to the central atom involved in the transition, while the light blue and red atoms correspond to occupied sites in the first and second rings, respectively. The third ring of atoms (orange) is also shown.

for the associated saddle-point searches while all moves were assumed to involve fcc sites[5].

We note that, for large islands, we have found that these approximations are adequate to describe island diffusion. In particular, the scaling of the island diffusion coefficient with the island size was found to be in excellent agreement with previous determinations [7]. Hence, the first part of the database was filled using studies of the diffusion of large islands (i.e. periphery diffusion using drag method and fcc occupation). All activation energies were determined using interaction potentials based on the embedded-atom method (EAM) as developed by Foiles *et al* [14].

While the third shell surrounding a central atom was included in the SLKMC simulations when building the large cluster database, in our KMC simulations only the first two shells were used, since the third shell surrounding a central atom has a relatively weak effect on the activation barrier. Accordingly, the large cluster database corresponds, as shown in figure 1, to the first two shells surrounding an occupied site and contains approximately 2302 configurations (not including symmetry) with approximately 4455 different 'moves' or transitions including concerted moves. Figure 2 shows some of the low-barrier moves included in this database.

The second part of the database was a small cluster database obtained from SLKMC simulations of islands of less than 19 atoms. In these small-island SLKMC simulations, it was found earlier [8] that the three assumptions used for large systems (periphery motion/drag/fcc) were unable to describe mechanisms revealed by molecular dynamics (MD) simulations of small islands on fcc(111). In particular, the most important mechanisms for diffusion of small islands were found (by MD) to be translation and rotation of the whole island. Since the drag method involving periphery atoms was unable to retrieve these two main mechanisms and others involving collective motion, we proceeded by calculating their barriers using the nudged-elastic band (NEB) method [15, 16] and incorporating them into the SLKMC database. The last hurdle was to overcome the hcp occupancy (seen in MD simulations) since our SLKMC database involves only fcc occupancy. Using the fact that, when a small cluster occupies fcc sites it moves as a whole to hcp sites and then to fcc sites, we described a sequence of fcc-to-hcp and hcp-to-fcc moves as a single fcc–fcc move with a modified prefactor determined using symmetry (degeneracy) factors [8]. Hence, the database for small clusters was built by hand using NEB for mechanisms revealed by MD simulations and involving collective motion of the islands, in addition to moves involving periphery atoms which were found using the drag method attached to SLKMC.

Since the small cluster database only involves clusters of less than 19 atoms, it only involves configurations in which there is a central atom surrounded by two partially filled rings and an empty third ring. We note that our small cluster database contains 50 configurations (not including symmetry) with 252 different 'moves' or transitions including concerted moves. Figure 3 shows some typical small cluster transitions which were included in this database.

In order to minimize the size of the database we have also used the symmetry of the fcc(111) surface. In particular, the following five symmetry operations were used: (1) 120° rotation, (2) 240° rotation, (3) mirror reflection, (4) mirror reflection followed by 120° rotation and (5) mirror reflection followed by 240° rotation. Thus, if a given configuration was not found in the database, then the symmetry operations mentioned above were performed and used to identify the configuration in the database. Since our database is a closed database, if the configuration is not found then no transition is identified. Figure 4 shows an example of a 120° rotation symmetry operation.

Thus, in order to find the possible transitions and their corresponding rates for a given configuration corresponding to an occupied central site and the surrounding rings, if the third ring was unoccupied, we first searched the small cluster database for a match for the first and second rings. If no match was found, or if the third ring was not empty, we then searched the large cluster database for a match. We note that, to save time, instead of doing a search through each

[5] In the drag method, the moving entity is dragged in very small steps toward the probable (aimed) final state. The dragged atom is constrained in the direction toward the aimed position while the other two degrees of freedom (perpendicular to this direction) and all degrees of freedom of the rest of the atoms in the system are allowed to relax. The other atoms are thus free to participate in the move, thereby activating many-particle processes in which neighbor adatoms start to follow the central leading atom. In connection with the SLKMC method, the central atom is always dragged toward one of its vacant fcc sites.

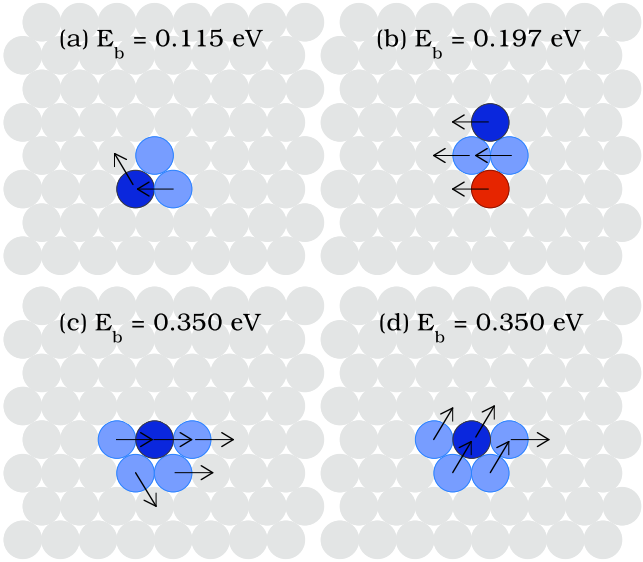


Figure 3. Some typical concerted moves in the small cluster database. Color code is the same as in figure 2.

database, we instead stored the data in a two-dimensional matrix with first (second) indices corresponding to the first (second) rings. However, if no match was found for either of the two indices, we then generated the (up to five) additional symmetry representations for each of our two configuration ring indices, in order to search for symmetry equivalents. Once a configuration was identified as belonging to the database, the possible moves and their respective barriers were added to our KMC lists. However, if after searching both databases, the configuration was not identified as part of the database, then nothing was done.

Figure 5 shows a histogram of the energy barriers corresponding to the 4712 processes (not including symmetry) in both databases. As can be seen, the distribution is very wide, covering activation energies as small as a few hundredths of an eV and as large as 0.8 eV. Assuming a prefactor of 10^{12} s^{-1}, as was assumed in our simulations, this corresponds at room temperature to rates ranging from 0.036 s^{-1} to approximately 8×10^{10} s^{-1}.

3. KMC algorithm

In order to maximize the efficiency of our simulations we have used a binary tree algorithm [6] combined with lists. In particular, at the beginning of a simulation we scan through the lattice and identify the configurations for every occupied site to determine the corresponding moves and activation barriers for each configuration. Based on this information we create an array of lists, where each list corresponds to one of the 4712 different possible mechanisms for activated events, and contains all the central atom locations corresponding to that type of event. The total rate for each type of move (corresponding to the rate for that move times the number of central atoms of that type) is calculated and placed at the 'base' of a binary tree. We then generate a random number (between 0 and the total rate for all events) and use the binary structure

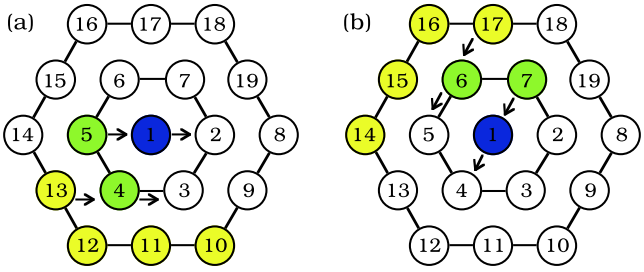


Figure 4. Example of 120° rotation symmetry operation showing two symmetry-related configurations (a) and (b).

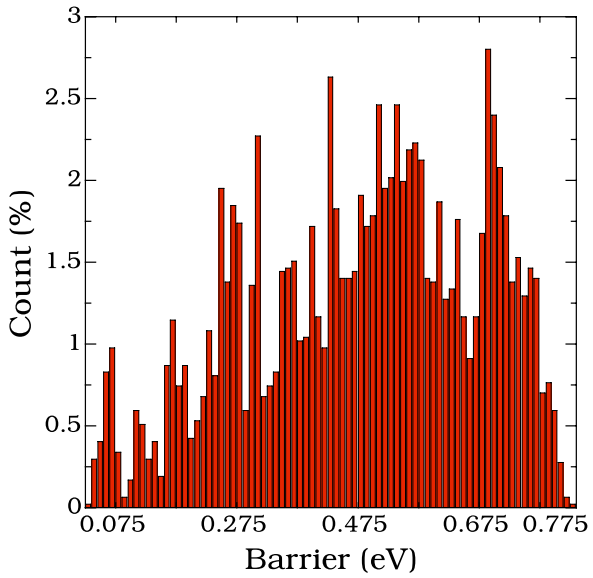


Figure 5. Histogram of energy barriers in the combined large and small cluster database (histogram width is 0.01 eV).

to efficiently select the list 'type' of the next move, while the particular move is randomly selected from the selected list. After each transition, the neighborhood of each changed site is updated along with the associated lists as well as the total rate and binary tree. This leads to a code which is efficient and scales with system size.

We note that all our simulations were carried out using the 1.4 GHz Itanium Cluster at the Ohio Supercomputer Center (OSC). By replacing a 'standard' KMC by one which uses a binary tree with lists as described above, as well as replacing a linear search through the database with a matrix as described in section 2, we were able to significantly reduce the average time per KMC step from 592.8 μs in our initial code to 24.6 μs. For comparison, we note that, in a typical KMC with a small number of different event types such as edge and/or corner diffusion, the corresponding time is typically 6 μs or less.

4. Algorithms for parallel kinetic Monte Carlo

Because of the large range of energy barriers (0.04–0.8 eV) in our database, as well as the relatively high temperature (room temperature) of our simulations, it is difficult to develop an efficient and accurate parallel algorithm. Therefore, before

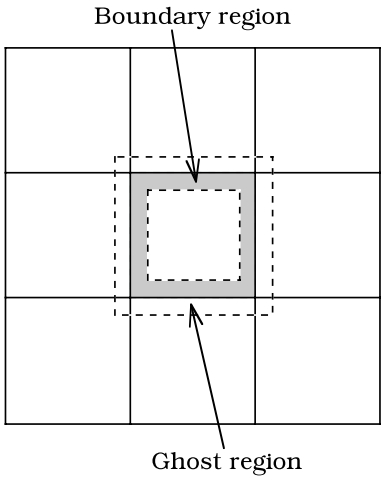


Figure 6. Schematic diagram of square decomposition for $N_p = 9$. Solid lines correspond to processor domains.

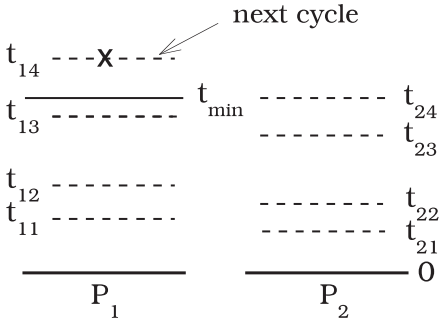


Figure 7. Time evolution of events for OSR and OSRPR algorithms with $G = 4$. Dashed lines correspond to selected events, while the dashed line with an **X** corresponds to an event exceeding t_{min} (see the text). In OSR this event is discarded while in OSRPR this event is added to the next cycle.

carrying out our parallel KMC simulations of coarsening we have tested and developed several different parallel algorithms in order to determine which is the most efficient. In particular, we have studied a recently developed rigorous algorithm, the 'optimistic synchronous relaxation' (OSR) algorithm as well as a modified version of this algorithm which we refer to as 'optimistic synchronous relaxation with pseudo-rollback' (OSRPR). In addition, we have tested the recently developed semi-rigorous synchronous sublattice (SL) algorithm. Finally, to reduce the number of events corresponding to boundary events between processors we have also developed a new method, which we refer to as 'dynamic boundary allocation' (DBA). Below we discuss each of these methods and some of the details of their implementation.

4.1. Optimistic synchronous relaxation (OSR) algorithm

One of the first rigorous algorithms for parallel discrete-event simulations was the synchronous relaxation algorithm developed by Lubachevsky [9]. We note that the application of this algorithm to KMC simulations as well as its scaling as a function of the number of processors N_p has been recently studied by Shim and Amar [11]. However, since this algorithm is relatively complex and requires multiple iterations for each cycle, Merrick and Fichthorn have recently developed a similar but simpler algorithm which they refer to as optimistic synchronous relaxation (OSR) [12].

Figure 6 shows a typical decomposition of a square system into N_p square regions, where N_p is the number of processors. Also indicated in figure 6 are the boundary and 'ghost' regions for the central processor, where the boundary region is defined as that portion of the processor's domain in which a change may affect neighboring processors. Similarly, the ghost region corresponds to that part of the neighboring processors' domains which can affect a given processor. Thus, in general the width of the boundary and ghost regions must be at least equal to the range of interaction.

As shown in figure 7, in the OSR algorithm in each cycle all processors start with the same initial time and then simultaneously and independently carry out KMC events in their domains until either the number of KMC events reaches a pre-determined fixed number G, or one of the events corresponds to a 'boundary event', i.e. an event which modifies the boundary region of the given processor and which can thus affect events in neighboring processors. Defining the time for the last event in each processor as t_{last}, a global communication is then carried out to determine the time t_{min} corresponding to the minimum of t_{last} over all processors. Each processor then 'rolls back' or undoes all KMC events which occur after t_{min}. If there are no boundary events then the processors all move on to the next cycle with the new starting time corresponding to t_{min}. However, if t_{min} corresponds to a boundary event, then an additional communication is needed to update the ghost and/or boundary regions of all processors affected by the boundary event.

We note that typically the OSR algorithm requires 2–3 global communications each cycle, one to determine t_{min}, another to determine if the event with t_{min} corresponded to a boundary event, and a third to update the boundary regions of the affected processors if there was a boundary event. To reduce the number of global communications we have encoded the processor identity as well as whether or not the last event was a boundary event, along with the least advanced time of each processor before doing a global communication to determine t_{min}. This was done by replacing t_{last} with a number whose most significant figures corresponded to t_{last} but whose least significant figures contained information about the processor ID and whether or not that processor had a boundary event[6]. Thus, in our implementation of the OSR algorithm only one global communication was needed if t_{last} corresponded to a non-boundary event, while two communications were needed if it was a boundary event.

[6] In this method, the time each processor advances from its previous cycle is multiplied by a very large number to form the integer part of the double precision packed number. The ratio of the processor ID to the total number of processors used N_p is then added to the decimal part if there is a boundary event in that processor. If there is no boundary event in that processor a decimal number is added such that it does not correspond to any processor identity. In our implementation the multiplying number was 10^{20}, which leads to good accuracy.

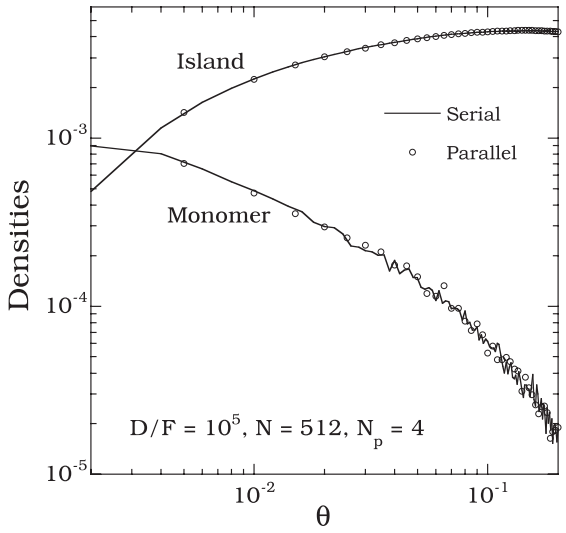


Figure 8. Comparison between parallel results using the OSRPR algorithm with square decomposition ($N_p = 4$) and serial results for a fractal model with $D/F = 10^5$ and $G = 7$.

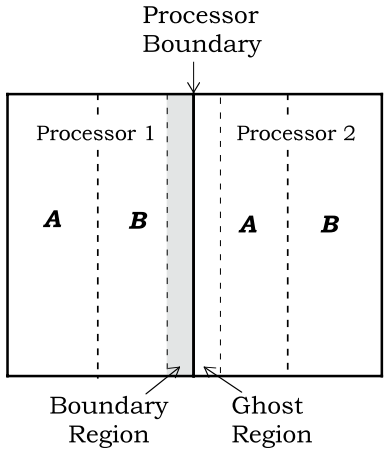


Figure 9. Schematic diagram of strip decomposition for $N_p = 2$. Each processor domain is subdivided into A and B sublattices. Boundary and ghost regions for B sublattice of processor 1 are also shown.

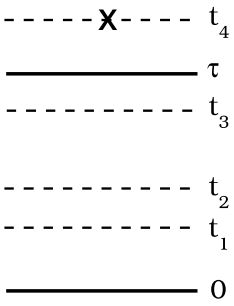


Figure 10. Time evolution in the SL algorithm. Dashed lines correspond to selected events, while the dashed line with an **X** corresponds to an event which is rejected since it exceeds the cycle time τ.

We note that, in the OSR algorithm for a given configuration, there is an optimal value of G which takes into account the tradeoffs between communication time (which is wasted if there are no boundary events) and rollbacks. While, in general, an adaptive method could be used to attempt to optimize the value of G from cycle to cycle, in practice we have found it more efficient to simply use trial and error to find the optimal fixed value of G for our simulation (see section 4.4).

4.2. Optimistic synchronous relaxation with pseudo-rollback (OSRPR) algorithm

In the OSR algorithm each processor discards all KMC events which occur after t_{min}. However, this is unnecessary if there are no boundary events in any of the processors. Therefore, we have considered a variation of the OSR algorithm (optimistic synchronous relaxation with pseudo-rollback) in which, when there are no boundary events in the system, those events that would have been discarded are added to the next cycle. This can reduce the loss of computational time due to undoing and then 'redoing' events and thus enhances the computational efficiency. As a test of the OSRPR algorithm, we have carried out parallel simulations using this algorithm for a 'fractal' model of irreversible submonolayer growth in which only monomer deposition and diffusion processes are included [11], with $N_p = 4$. As expected, there is excellent agreement between serial and parallel results for the island and monomer densities (see figure 8).

4.3. Synchronous sublattice (SL) algorithm

In order to maximize the parallel efficiency we have also carried out simulations using the semi-rigorous synchronous sublattice (SL) algorithm recently developed by Shim and Amar [13]. To avoid conflicts between processors, in the SL algorithm each processor domain is divided into subregions or sublattices (see figure 9). A complete synchronous cycle corresponding to a cycle time τ is then as follows. At the beginning of a cycle, each processor's local time is initialized to zero. One of the sublattices (A or B) is then randomly selected so that all processors operate on the same sublattice during the cycle. Each processor then simultaneously and independently carries out KMC events in the selected sublattice until the time of the next event exceeds the time interval τ (see figure 10). The processors then communicate any necessary changes (boundary events) with their neighboring processors, update their event rates and move on to the next cycle using a new randomly chosen sublattice. We note that, in order to ensure accuracy, the cycle time must typically be less than or equal to the inverse of the fastest possible single-event rate in the system [13].

Since it only requires local communication, the scaling behavior of the SL algorithm is significantly better than for the OSR and OSRPR algorithms. As a result, it has been shown to be relatively efficient in parallel KMC simulations of a variety of models of growth [13, 17] and island coarsening [18]. In addition, while it is not exact, in simulations of a variety of models [13, 17, 18] it was found that, unless the processor size is extremely small (smaller than a 'diffusion length') or the cycle time is too large, there is essentially perfect agreement

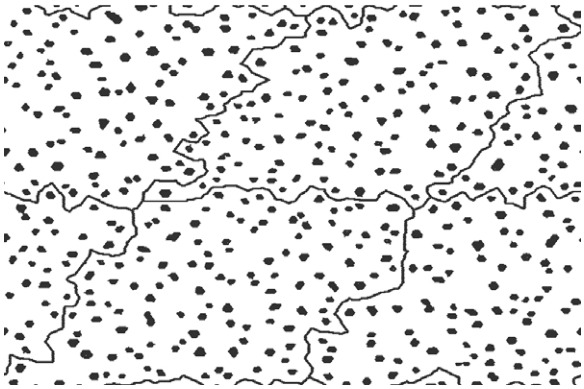


Figure 11. Example of square decomposition with DBA for system size $L = 512$ (triangular lattice with periodic boundary conditions) and $N_p = 4$.

between the results of parallel simulations using the SL algorithm and serial simulations. Furthermore, while the cycle time must typically be smaller than the inverse of the fastest possible single-event rate in the system [13], it has recently been shown [18] that in simulations of coarsening significantly longer cycle times can be used, thus decreasing the overhead due to communications and increasing the parallel efficiency.

4.4. Dynamic boundary allocation (DBA)

One of the main factors controlling the efficiency of a parallel KMC algorithm is the existence of boundary events, which can lead to 'rollback' in the OSR algorithm and can decrease the accuracy of semi-rigorous algorithms such as the SL algorithm. A decrease in the number of boundary events can also significantly increase the optimal value of G used in the OSR and OSRPR algorithms and thus reduce the communications overhead. In the case of the SL algorithm, such a decrease can also allow the cycle time τ to be increased without sacrificing accuracy, thus increasing the parallel efficiency.

As an example, edge diffusion near a processor boundary can lead to a large number of boundary events. Thus, if a processor or sublattice boundary passes near or cuts through an island this can significantly reduce the parallel efficiency. Accordingly, we have developed a method for dynamic boundary allocation (DBA) which keeps the processor and sublattice boundaries as far away as possible from islands. In our DBA method, we start with a spatial decomposition of the lattice using straight-line boundaries and use the island center-of-mass to assign islands to each processor or sublattice. We then use a 'burning-algorithm' starting from each island boundary to determine the processor and/or sublattice boundaries between islands.

Since atoms and islands can diffuse and/or grow during a simulation, atoms will eventually move into the boundary region and, as a result, the parallel efficiency will decrease. To overcome this, DBA is carried out regularly (e.g. several times per second of simulated time) to adjust the processor boundaries. Figure 11 shows a typical square decomposition with DBA for the case of island coarsening for a system size $L = 512$ with four processors. As can be seen, the processor boundaries are relatively convoluted but remain well away from the island edges. A similar DBA decomposition is shown in figure 12 for the case of sublattice strip decomposition with four processors. We note that, in this case, there are eight separate sublattices.

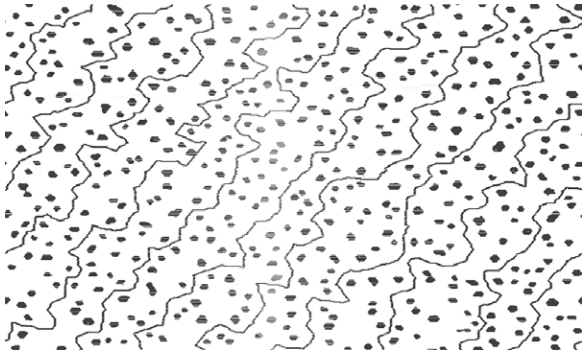


Figure 12. The same as figure 11 but for strip decomposition into sublattices with $N_p = 4$.

Table 1. Comparison of efficiencies of parallel algorithms with $N_p = 2$ and $L = 1024$. The cycle time τ used for the SL algorithm is given in parentheses. f_R is the fraction of rollback events per cycle in the OSR and OSRPR algorithms.

	No DBA			DBA		
Algorithm	PE	$G(\tau)$	f_R	PE	$G(\tau)$	f_R
OSR	0.33	9	0.27	0.43	161	0.21
OSRPR	0.47	11	0.04	0.58	161	0.02
SL	0.52	(10^{-7} s)	—	0.74	(10^{-6} s)	—

5. Results

5.1. Parallel efficiency results

Before presenting our results for island coarsening, we first present our results for the parallel efficiencies of the OSR, OSRPR and SL algorithms (with and without DBA) as obtained from simulations of room-temperature Ag/Ag(111) island coarsening (as described in more detail in section 5.2) using our SLKMC-derived database. We note that, in each case, the parallel efficiency (PE) was obtained using the expression $\mathrm{PE} = t_{ser}/(N_p\, t_p)$, where t_{ser} is the time for a serial simulation of the entire system (system size $L = 1024$), while t_p is the time for the corresponding parallel simulation where N_p is the number of processors.

Tables 1 and 2 summarize our results for the overall parallel efficiency of the OSR, OSRPR and SL algorithms obtained from coarsening simulations with and without DBA for $N_p = 2$ and 4. As can be seen, in contrast to previous results using the OSR algorithm to study Ag/Ag(111) island nucleation and growth at very low temperatures [12], in our room-temperature simulations of Ag(111) island coarsening, the PE of the OSR algorithm is generally low. In addition, due to the increased cost for global communications as well as the increased ratio of the boundary region to the processor 'core' region, the PE decreases significantly with

Table 2. Comparison of efficiencies of parallel algorithms with $N_p = 4$ and $L = 1024$.

	No DBA			DBA		
Algorithm	PE	$G(\tau)$	f_R	PE	$G(\tau)$	f_R
OSR	0.14	7	0.53	0.25	151	0.45
OSRPR	0.22	7	0.28	0.41	151	0.05
SL	0.39	(10^{-7} s)	—	0.60	(10^{-6} s)	—

increasing N_p. Our results also indicate that including pseudo-rollback (OSRPR algorithm) leads to a significant increase in the parallel efficiency although for the case $N_p = 4$ the PE still remains below 50%. However, in all cases we find that the SL algorithm yields the highest PE due to its significantly reduced communication overhead and relatively large number of KMC events carried out during a given cycle time τ.

We now discuss the effects of DBA on the PE as well as on the optimal value of G for the OSR and OSRPR algorithms. We note that without DBA the optimal value of G for the OSR and OSRPR algorithms is about 10 for $N_p = 2$ and decreases slightly with increasing N_p. On the other hand, with DBA the optimal value of G increases significantly due to the significantly reduced number of boundary events. However, despite the large increase in G, the improvement of the PE for the OSR algorithm is only moderate due to the significant fraction of rollback events f_R. On the other hand, the increase in the PE with DBA for the OSRPR algorithm is quite noticeable since the fraction of rollback events is significantly decreased.

We now consider the parallel efficiency of the SL algorithm both with and without DBA. As already noted, in parallel KMC simulations of island nucleation and growth using the SL algorithm, the cycle time must typically be smaller than the inverse of the fastest possible single-event rate in the system [13]. More recently it was found [18] that, in simulations of coarsening, significantly longer cycle times can be used, thus decreasing the overhead due to communications and increasing the parallel efficiency. As shown in figure 13, by reducing the number of boundary events with DBA, the cycle time can be made even longer without affecting the accuracy, thereby improving the parallel efficiency significantly. Thus, we find (see tables 1 and 2) that, due to the relatively low communications overhead as well as the relatively long cycle times, the PE for the SL algorithm is, in general, significantly higher than for the OSR and OSRPR algorithms. Accordingly, in our parallel KMC simulations of Ag(111) island coarsening over extended times, we have used the SL algorithm with $N_p = 4$ and a cycle time $\tau = 10^{-6}$ s. We note that, with these parameters, the average number of events carried out per cycle per processor ($n_e \simeq 76$) was somewhat smaller than the optimal value of G using the OSRPR algorithm.

5.2. Results for Ag/Ag(111) island coarsening

In order to test our parallel KMC algorithm and also to apply the large database to a realistic problem, we have carried out parallel KMC simulations of Ag(111) island coarsening

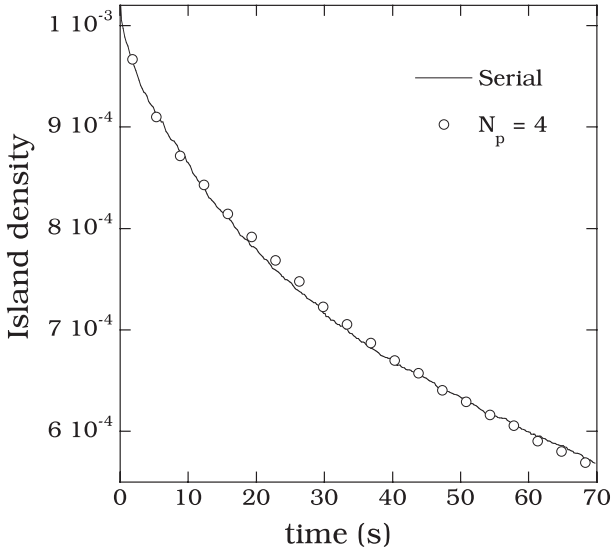


Figure 13. Comparison of results for the island density obtained from coarsening simulations carried out using serial KMC with the corresponding results obtained using the SL algorithm with DBA for $N_p = 4$ and $\tau = 10^{-6}$ s.

at room temperature. The initial configuration used in our coarsening simulations (see figure 14(a)) was generated by depositing $\theta = 0.1$ ML at a deposition rate of 1 ML s^{-1} at 145 K. We then carried out parallel KMC simulations of coarsening at room temperature for 400 s using the SL algorithm with DBA and a cycle time of 10^{-6} s with $N_p = 4$. In order to avoid finite-size effects our simulations were carried out using a relatively large lattice size $L = 1024$, while to have good statistics our results were averaged over 10 runs. We note that, if each run had been carried out using serial KMC, it would have taken almost 5 weeks. However, because of the use of parallel KMC the whole set of runs took only 2 weeks. In order to study the coarsening behavior, the island-size distribution $N_s(\theta, t)$ corresponding to the density of islands of size s (where s is the number of atoms in an island) at time t was measured along with the island density $N = \sum_{s \geqslant 2} N_s$, monomer density N_1 and average island size $S = \sum_{s \geqslant 2} s N_s / N$.

Before presenting our simulation results, we note that, for the case of 2D clusters on a surface, there are two particular limiting regimes—Ostwald ripening [19, 20] and cluster diffusion and coalescence [21–29]—in which the coarsening is dominated by diffusion. In the case of Ostwald ripening the islands are assumed to be immobile, while the coarsening is mediated by a background density of diffusing atoms such that islands bigger than a critical island size grow while smaller islands shrink or evaporate. This results in power-law growth of the average island size $S(t) \sim t^{2n}$ where $n = 1/3$ [30, 31]. However, in the case of cluster diffusion and coalescence, if the cluster diffusion coefficient $D(s)$ decays as a power law with island size s, i.e. $D(s) \sim s^{-x}$, then $n = 1/2(1 + x)$ [21]. In this case, three different limiting cases are of particular interest [22–28]—cluster diffusion due to periphery diffusion ($x = 3/2, n = 1/5$), cluster diffusion due to correlated evaporation/condensation ($x = 1, n = 1/4$) and finally cluster

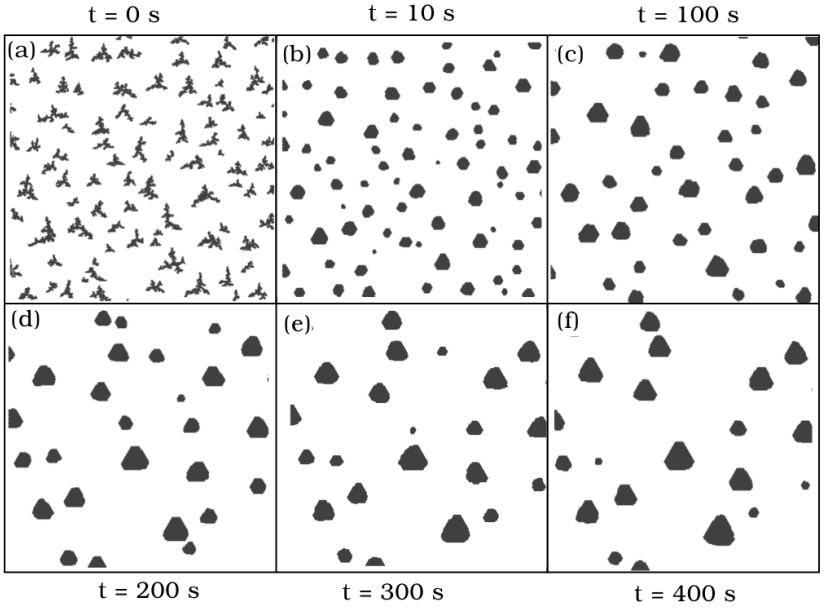


Figure 14. Evolution of island morphology during room-temperature coarsening. Pictures correspond to 256 × 256 portions of a 1024 × 1024 system.

diffusion due to uncorrelated evaporation–condensation ($x = 1/2, n = 1/3$). Although asymptotically, one might expect one of these processes to dominate, at intermediate times all of these processes may play a role.

Figure 14 shows a typical example of the evolution of a portion of the system starting from the initial configuration at $t = 0$ and ending with the final configuration at $t = 400$ s. As can be seen, during the annealing process there is a dramatic change in the island morphology while the average island size increases dramatically. In particular, the system evolves from the small dendritic islands at $t = 0$ shown in figure 14(a) to the much larger truncated-hexagonal islands at $t = 400$ s shown in figure 14(f). These results for the island morphology are consistent with previous work using a simplified KMC model based on the same EAM potential for Ag, which indicates that for this potential the A step edge is energetically favored over the B step edge [32].

Figure 15 shows a log–log plot of the average island size $S(t)$ as a function of time. As can be seen, the effective slope increases with time while a fit to the late-time region indicates a coarsening exponent $n \simeq 0.47$. However, if the initial island size $S(0)$ is subtracted, then as shown in the inset, after an initial transient period the slope appears to approach an asymptotic value (0.70) which is close to $2/3$ and thus corresponds to a coarsening exponent $n \simeq 1/3$. Also shown in figure 15 (dashed curve) are results for the average island size starting with the same initial configurations but without the inclusion of multi-atom or concerted events during coarsening. As can be seen, while the asymptotic coarsening behavior is very similar, the average island size is significantly smaller. Thus, the inclusion of complex concerted moves in our KMC database significantly increases the average island size.

In order to better understand the asymptotic coarsening behavior we have also calculated the effective exponent

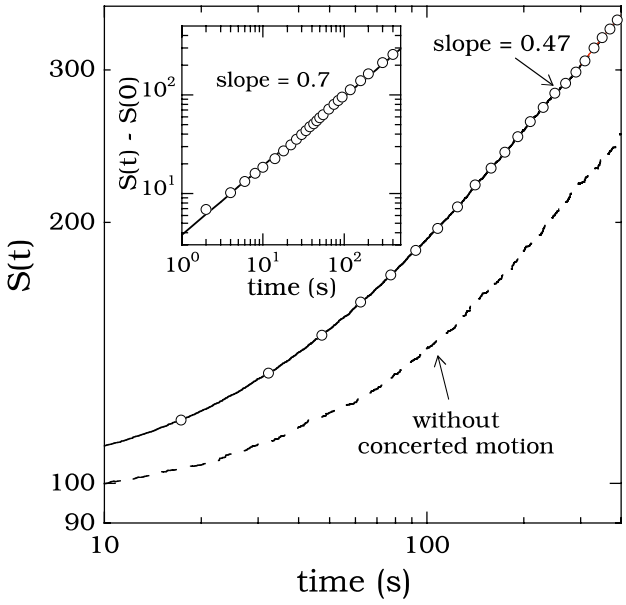


Figure 15. Average island size $S(t)$ as a function of time (open circles). The dashed line corresponds to simulations without concerted motion. The inset shows a log–log plot of $S(t) - S(0)$.

$n_{\rm eff}(t)$ [33] using the expression

$$n_{\rm eff}(t) = \frac{\ln[S(t_{\rm f})/S(t_{\rm i})]}{2\ \ln(t_{\rm f}/t_{\rm i})} \qquad (1)$$

where $t_{\rm f}/t_{\rm i} \simeq 2$. Figure 17 shows a plot of $n_{\rm eff}$ as a function of inverse average island size $1/S$. As can be seen, a linear fit to the data gives an asymptotic exponent $n_{\rm eff}(\infty) = 0.72$, while a fit to the last three points gives $n_{\rm eff}(\infty) = 0.66$. This suggests that the asymptotic exponent is indeed close to $1/3$. We note that such an exponent is consistent with both

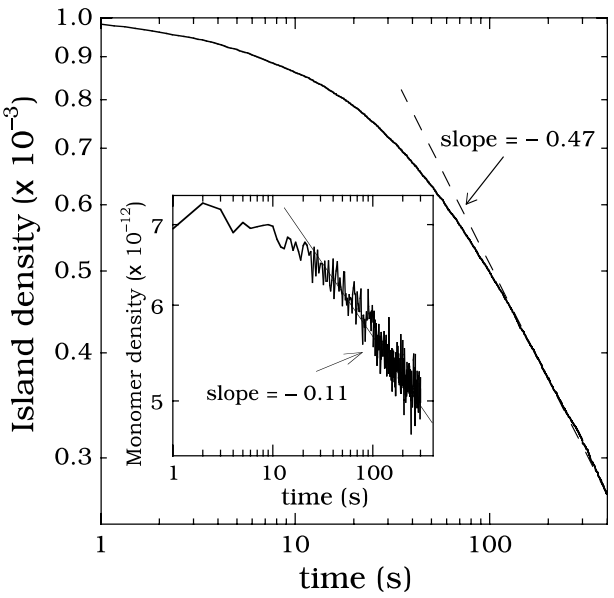


Figure 16. Island and monomer (inset) densities as a function of time.

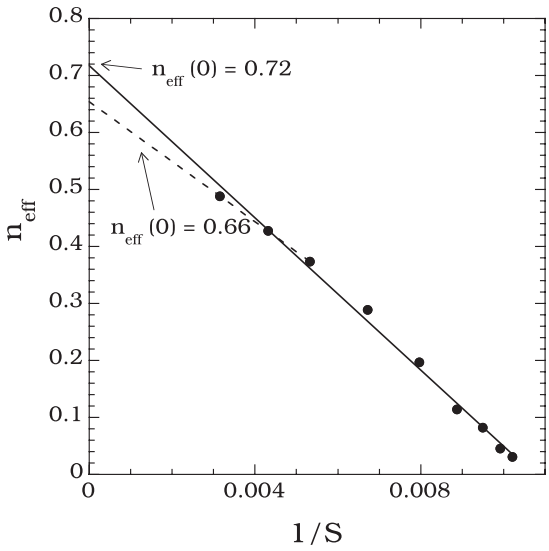


Figure 17. Effective coarsening exponent $n_{\rm eff}$ as a function of $1/S$.

cluster diffusion due to uncorrelated evaporation–condensation and Ostwald ripening. However, a detailed analysis of our simulations indicates that, while there is some cluster diffusion and coalescence at early times, at later times there is very little cluster diffusion. Instead, the coarsening appears to proceed via evaporation–condensation, i.e. small clusters shrink while larger clusters grow. Thus, in general, our results are consistent with Ostwald ripening.

We note that, in recent parallel KMC simulations of coarsening using a simplified bond-counting model [18], Ostwald ripening was also observed, although in this case an exponent of $1/3$ was obtained directly from the late-time slope on a log–log plot of average island size $S(t)$ as a function of annealing time t. In this case, it was also noted that the effective value of the coarsening exponent n on such a plot did not approach $1/3$ until the monomer density was larger than the island density, while for earlier times the effective exponent was close to $1/4$. We note that such a condition is reasonable, since only when the monomer density is significantly larger than the island density can one think of the islands as being in quasi-equilibrium with a gas of monomers.

In order to determine if such a late-time regime has been reached in our Ag/Ag(111) island coarsening simulations, we have measured both the island and monomer densities as a function of time as shown in figure 16. We note that the monomer hopping rate in our simulations ($D_{\rm m} = 8 \times 10^{10}$ s^{-1}) is sufficiently large, while the adatom-island detachment rate is significantly lower than $D_{\rm m}$ ($D_{\rm detach} \simeq 2.2 \times 10^{-4} D_{\rm m}$) and thus most of the time there are no monomers in the system, and instead the main processes are edge diffusion and island rearrangement. Accordingly, to accurately measure the monomer density, we instead measured the number of monomer hopping events over an extended time interval of 10^{-3} s, and then divided this number by the product of this time interval and the monomer hopping rate. As can be seen the monomer density is significantly lower than the island density, thus indicating that we have not yet reached the asymptotic regime. However, the slower decay of the monomer density indicates that, at large enough times, the monomer density will indeed be larger than the island density. These results also support our conclusion that, even though the asymptotic regime has not yet been reached, the coarsening is due to evaporation–condensation mediated by monomer diffusion.

We now consider the time evolution of the scaled island-size distribution (ISD) given by [34]

$$f(s/S) = N_s(t) S^2/\theta. \qquad (2)$$

Figure 18 shows our results for the scaled ISD at $t = 0$, along with results at later times. Somewhat surprisingly, we find that, even though the average island size increases significantly, the scaled ISD has very little dependence on time. This result is also in strong contrast to our previous study of a bond-counting model in which the asymptotic coarsening behavior was observed, while the scaled ISD was found to broaden and become less sharply peaked in the asymptotic regime. We note that this independence of the scaled ISD on time is consistent with recent experiments on room-temperature annealing of Cu/Cu(100) islands [35], in which the scaled ISD was also found not to change during the coarsening process. We believe that this independence of the ISD on annealing time is again due to the fact that we have not yet reached the asymptotic regime.

5.3. Analysis of energy barriers

In order to gain more insight into the dominant processes during coarsening we have also analyzed the frequency of events as a function of energy barrier. Figure 19 shows a histogram of the energy barriers for all events carried out during the first 100 s of coarsening. Somewhat surprisingly, we find that the energy barriers for the most frequently selected processes are spread over a relatively wide range of

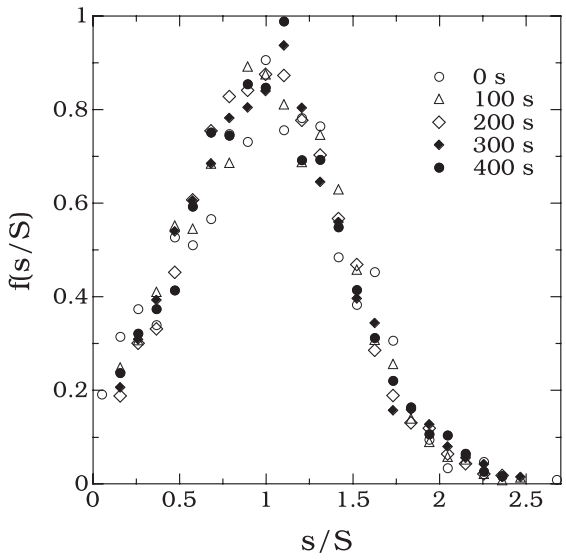


Figure 18. Scaled island-size distribution at four different times during coarsening.

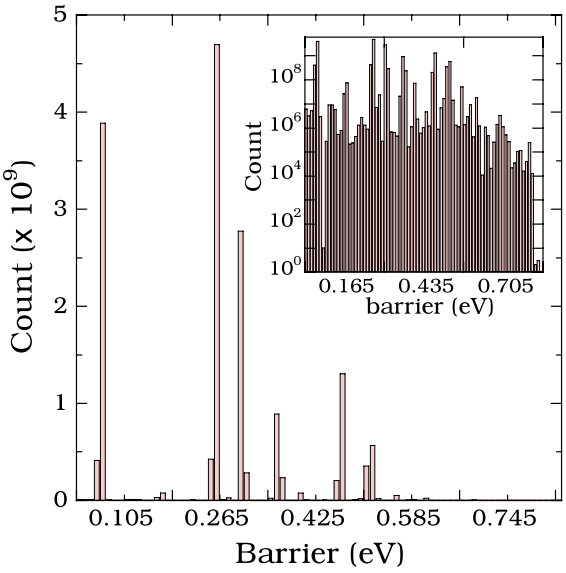


Figure 19. Histogram of energy barriers of selected events during the first 100 s of coarsening. Histogram width is 0.01 eV. The inset shows the same results on a semi-log scale.

values ranging from 0.06 to 0.5 eV, while the barrier for the most frequently selected event is approximately 0.26 eV. The inset of figure 19 shows the same data on a semi-log plot and indicates even more clearly that the energy barriers for selected events correspond to a broad distribution. Figure 20 shows the four most frequently selected moves other than monomer diffusion (which has a barrier of 0.066 eV) during the first 100 s of simulation. As can be seen the most frequently selected event corresponds to corner-rounding, with a barrier of approximately 0.07 eV. However, the next two most frequently selected events correspond to edge diffusion (see figures 20(b) and (c)) and have significantly higher energy barriers (0.26 eV). We also note that detachment from a

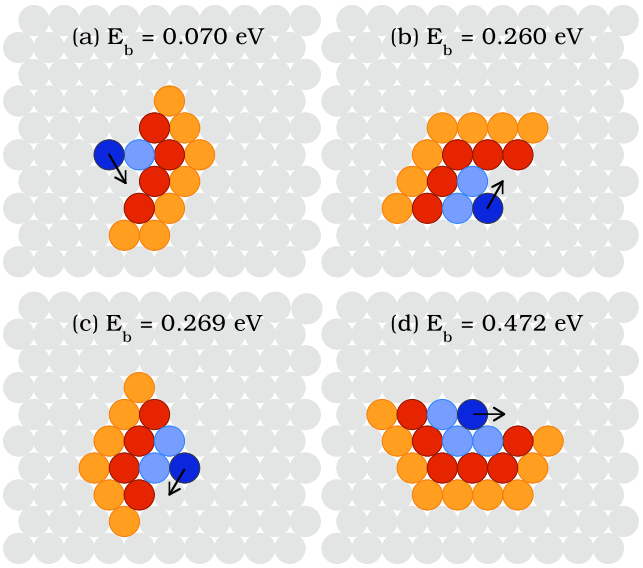


Figure 20. Configurations and energy barriers corresponding to the four most frequently selected events during the first 100 s of coarsening.

kink shown in figure 20(d) occurs quite frequently despite its relatively high energy barrier (0.472 eV). However, it also leads to rapid reattachment to the kink site (with a much lower energy barrier of 0.253 eV), as shown in figure 20(b). Thus, our results indicate that a variety of different processes with a wide range of energy barriers may play an important role in room-temperature island coarsening.

6. Conclusion

As a test of the applicability of parallel algorithms to realistic simulations over extended timescales and length scales, we have presented the results of kinetic Monte Carlo simulations of the room-temperature coarsening of Ag(111) islands carried out using a large database obtained via SLKMC simulations. Our results indicate that, while cluster diffusion and coalescence play a role for small clusters and at very early times, at late time the coarsening proceeds via Ostwald ripening, i.e. large clusters grow while small clusters evaporate. In addition, an asymptotic analysis of our results for the average island size as a function of time leads to a coarsening exponent $n = 1/3$ in good agreement with theoretical predictions for this case. By comparing with simulations without concerted (multi-atom) moves, we also find that the inclusion of such moves significantly increases the average island size. Somewhat surprisingly we also find that, while the average island size increases significantly during coarsening, the scaled island-size distribution does not change significantly.

In addition to presenting results for the coarsening of Ag(111) islands we have also presented results for the efficiency and accuracy of a number of different parallel algorithms. In particular, we have presented results for the optimistic synchronous relaxation (OSR), optimistic synchronous relaxation with pseudo-rollback (OSRPR) and

semi-rigorous synchronous sublattice (SL) algorithms, as well as a number of improvements such as dynamic boundary allocation (DBA). Because of the relatively high temperature of our simulations, as well as the large range of energy barriers present in the database ranging from 0.05 to 0.8 eV, developing an efficient algorithm for parallel KMC and/or SLKMC simulations was particularly challenging. However, by using dynamic boundary allocation (DBA) to minimize the number of boundary events, we have achieved significantly improved parallel efficiencies for the OSRPR and SL algorithms. In particular, the optimal value of G increased significantly for both the OSR and OSRPR algorithms due to the significantly reduced number of boundary events. However, there was a negligible improvement in the parallel efficiency for the OSR algorithm due to the significant fraction of rollback events. Finally, we note that, among the three parallel algorithms which we have tested, the semi-rigorous SL algorithm with DBA led to the highest parallel efficiencies. As a result, we have obtained reasonable parallel efficiencies in our simulations of room-temperature Ag(111) island coarsening using this algorithm for a moderate number of processors (e.g. $N_p = 2$ and 4). Since the SL algorithm scales with system size for fixed processor size, while the parallel efficiency increases with increasing processor size, we expect that comparable and/or even larger parallel efficiencies should be possible in parallel KMC and/or SLKMC simulations of larger systems with larger numbers of processors.

In conclusion, we have carried out realistic parallel KMC simulations of Ag(111) island coarsening using a large database obtained from SLKMC simulations and tested the parallel performance of the OSR, OSRPR and SL algorithms with and without DBA. We find that the SL algorithm with DBA yields the highest parallel efficiency due to the significantly increased cycle time, and also exhibits the best scaling behavior as a function of the system size and number of processors. However, the parallel efficiency for the OSRPR algorithm with DBA is also quite reasonable for a relatively small number of processors, suggesting that this algorithm may also be useful in a variety of parallel KMC simulations. Finally, our coarsening simulation results indicate that, while cluster diffusion and coalescence play a role at early and intermediate times, at late times the coarsening appears to proceed via Ostwald ripening.

Acknowledgments

This work was supported by the NSF through grants CCF-0428826 and DMR-0606307. We would also like to acknowledge grants of computer time from the Ohio Supercomputer Center (grant no. PJS0245).

References

[1] Bortz A B, Kalos M H and Lebowitz J L 1984 *J. Comput. Phys.* **17** 10
[2] Gilmer G H 1976 *J. Cryst. Growth* **35** 15
[3] Voter A F 1986 *Phys. Rev.* B **34** 6819
[4] Maksym P A 1988 *Semicond. Sci. Technol.* **3** 594
[5] Fichthorn K A and Weinberg W H 1991 *J. Chem. Phys.* **95** 1090
[6] Blue J L, Beichl I and Sullivan F 1995 *Phys. Rev.* E **51** R867
[7] Trushin O, Karim A, Kara A and Rahman T S 2005 *Phys. Rev.* B **72** 115401
[8] Karim A, Al-Rawi A N, Kara A, Rahman T S, Trushin O and Ala-Nissila T 2006 *Phys. Rev.* B **73** 165411
[9] Eick S G, Greenberg A G, Lubachevsky B D and Weiss A 1993 *ACM Trans. Model. Comput. Simul.* **3** 287
[10] Lubachevsky B D and Weiss A 2001 *PADS'01: Proc. 15th Workshop on Parallel and Distributed Simulation* (Piscataway, NJ: IEEE)
[11] Shim Y and Amar J G 2005 *Phys. Rev.* B **71** 115436
[12] Merrick M and Fichthorn K A 2007 *Phys. Rev.* E **75** 011606
[13] Shim Y and Amar J G 2005 *Phys. Rev.* B **71** 125432
[14] Foiles S M, Baskes M I and Daw M S 1986 *Phys. Rev.* B **33** 7983
[15] Jónsson H, Mills G and Jacobsen K W 1998 *Classical and Quantum Dynamics in Condensed Phase Simulations* ed B J Berne *et al* (Singapore: World Scientific)
[16] Henkelman G and Jonsson H 2001 *J. Chem. Phys.* **115** 9657
[17] Shim Y and Amar J G 2006 *Phys. Rev.* B **73** 035423
[18] Shi F, Shim Y and Amar J G 2007 *Phys. Rev.* E **76** 031607
[19] Ostwald W 1901 *Z. Phys. Chem.* **37** 385
[20] Voorhees P W 1985 *J. Stat. Phys.* **38** 231
[21] Meakin P 1990 *Physica* A **165** 1
[22] Wen J M, Chang S-L, Burnett J W, Evans J W and Thiel P A 1994 *Phys. Rev. Lett.* **73** 2591
[23] Van Siclen C D 1995 *Phys. Rev. Lett.* **75** 1574
[24] Khare S V, Bartelt N C and Einstein T L 1995 *Phys. Rev. Lett.* **75** 2148
[25] Sholl D S and Skodje R T 1995 *Phys. Rev. Lett.* **75** 3158
[26] Sholl D S and Skodje R T 1996 *Physica* A **231** 631
[27] Soler J M 1996 *Phys. Rev.* B **53** R10540
[28] Pai W W, Swan A K, Zhang Z and Wendelken J F 1997 *Phys. Rev. Lett.* **79** 3210
[29] Kandel D 1997 *Phys. Rev. Lett.* **79** 4238
[30] Lifshitz I M and Slyozov V V 1961 *J. Phys. Chem. Solids* **19** 35
[31] Wagner C 1961 *Z. Elektrochem.* **65** 581
[32] Cox E, Li M, Chung P, Ghosh C, Rahman T S, Jenks C J, Evans J W and Thiel P A 2005 *Phys. Rev.* B **71** 115414
[33] Amar J G, Sullivan F E and Mountain R D 1988 *Phys. Rev.* B **37** 196
[34] Bartelt M C and Evans J W 1992 *Phys. Rev.* B **46** 12675
[35] Pomeroy J M and Brock J D 2006 *Phys. Rev.* B **73** 245405