Dynamical space partitioning for acceleration of parallelized lattice kinetic Monte Carlo simulations

Takeshi Nishimatsu
Samsung R&D Institute Japan,
Yokohama 230-0027, Japan
t.nishimatsu@samsung.com

Anthony Payet
Semiconductor Research Center,
Samsung Electronics,
Hwaseong 18448, Korea

Byounghak Lee
Samsung Semiconductor Inc.,
3655 N 1st St.,
San Jose, CA 95134, USA

Yasuyuki Kayama
Samsung R&D Institute Japan,
Yokohama 230-0027, Japan

Kiyoshi Ishikawa
Samsung R&D Institute Japan,
Yokohama 230-0027, Japan

Alexander Schmidt
Semiconductor Research Center,
Samsung Electronics,
Hwaseong 18448, Korea

Inkook Jang
Semiconductor Research Center,
Samsung Electronics,
Hwaseong 18448, Korea

Dae Sin Kim
Semiconductor Research Center,
Samsung Electronics,
Hwaseong 18448, Korea

***Abstract*—A new dynamical space partitioning method is presented in a parallelized lattice kinetic Monte Carlo (kMC) simulator to overcome the loss of parallel efficiency found in other parallelized kMC simulators. The dynamical partitioning of the simulation cell allows better load balancing through all threads hence reducing time consuming events during the simulation. The new method is evaluated against both hypothetical and real cases. In both cases, minimal differences between serial and parallelized simulations are found. In real cases, other code optimizations may be needed to further improve the parallel efficiency.**

***Index Terms*—kMC, nano-scale, FinFET, OpenMP, shared memory, stochastic, parallelization efficiency**

I. Introduction

Lattice kinetic Monte Carlo (kMC) simulation is a powerful tool to simulate atomic level semiconductor technology processes of nano-scale fabrication. While downsizing of semiconductor devices is steadily pushed on, atomistic simulations, including kMC, have become more and more imperative. Moreover, among many atomistic simulation methods, kMC is almost the only method which can simulate stochastic phenomena such as growth and diffusion in realistic system sizes (larger than $10 \times 10 \times 10$ nm^3) and realistic time scales (minutes or hours).

To accelerate such kMC simulations of nano-scale semiconductor devices, several parallelization approaches for kMC have been introduced, tested, and practically used. Most parallelization approaches of kMC assume that possible events are distributed uniformly in simulation cells and parallelization efficiency become maximum in the cases of uniform distribution. However, especially in process simulations of large-scale devices such as array of FinFETs, sites having possible events (diffusing dopant atoms, surface sites of epitaxial growth, etc.) can be initially localized in simulation cells. In such localized cases, the simulation speed slows down and the accuracy deteriorates due to poor load balancing of the parallelization.

In this paper, we propose a new dynamical space partitioning method to overcome localized events issue. We apply our new dynamical space partitioning method to two earlier parallelization approaches of kMC, Ref. [1] and [2].

II. Methodology

The lattice kMC method is a Monte Carlo method computer simulation intended to simulate the time evolution of events occurring in a crystal lattice. Each event is assumed to have rate (or frequency) p_i of occurrence, where i is the index of each event. The general serial kMC algorithm using a binary tree as a rate table, as described in Sec. 4.4 of Ref. [1], is as following: (1) Generate two random numbers r_1 and $r_2 \in (0, 1]$, (2) Advance the clock $\Delta t = \frac{1}{p_s} \log \frac{1}{r_1}$, where p_s is the total cumulative rate calculated in previous iteration, (3) Search in the binary tree for the smallest j such that $r_2 p_s \leq \sum_{i=1}^{j} p_i$, (4) Perform the jth event, (5) Compute rate p_i of affected events, (6) Percolate changed rates up to the top of the binary tree, yielding $p_s = \sum_{i=1}^{N} p_i$, (7) Repeat this loop until the requested simulated time, t_{sim}. Therefore, The kMC time evolution has inherently serial nature and parallelization of kMC requires approximation.

In 2009, Slepoy *et al*. invented simple but efficient algorithm for parallelized kMC [1]. Using Message Passing Interface (MPI) and dividing a simulation cell into sectors, their kMC loop is: (1) Update states of boundary sites adjacent to the local sector, (2) Update rates of sites in local sector, (3) Run kMC on sectors until t_{stop}, (4) Tell updated states in boundary sites to neighboring sectors (5) Advance the clock t_{stop} and repeat this loop until the requested simulated time t_{sim}. Good efficiency of this parallelized algorithm is coming from step (3) in which ordinary kMC loops are repeated and multiple events are performed in a sector *locally*. However, localized events cause imbalance of the number of events executed in t_{stop}. Consequently, the parallelized computation speed slows down and results become less accurate. In this paper, we identify this algorithm as "TimeStop" approach.

In 2015, Martin-Bragado *et al.* reported a parallelization algorithm [2] of kMC for shared memory computers using OpenMP. They sliced a simulation cell into m domains, where m is the number of parallel threads, then, sliced each domain into three subdomains. With this idea, they successfully avoided conflicts during the update of event tables of neighboring subdomains as: (1) The total cumulative rate for each subdomain I, having N_I events to simulate, is built as $P_I = \sum_{i=1}^{N_I} p_i$, (2) A maximum of all the cumulative rates is computed: $P_{\max} \geq \max P_I$, (3) Null events rates are assigned to each subdomain: $p_I^{\text{null}} = P_{\max} - P_I$, (4) A random number is used to choose one subdomain from three subdomains, (5) For each chosen subdomain, one event is picked up proportionally to its rate and executed, (6) Advance the clock $\Delta t = \frac{1}{3P_{\max}} \log \frac{1}{r}$, where $r \in (0, 1]$ is a random number, (7) This loop is repeated until t_{sim} is reached. In this paper, we identify this algorithm as "OneStep" approach. This "OneStep" parallelization approach is more robust but slower than the "TimeStop" approach. Indeed, step (5) is the part done in parallel, but the creation and annihilation of the parallel threads have a equivalent or higher computational cost than the execution of an event in a subdomain and per kMC loop. It should be also noted that step (2) might not be done in parallel, because finding maximum among only $3m$ of P_I in parallel is expensive in the shared memory computer. Due to the step (3), imbalance of P_I causes large null events rates at some subdomains and execution of null events is the origin of slowness of this approach.

III. Dynamical space partitioning

In this paper, a simulation cell is partitioned into m sliced domains and each domain is sliced into three subdomains, where m is the number of parallel OpenMP threads, as proposed in Ref. [2] and as schematically illustrated in Fig. 1(a). This partitioning method can be used not only with the "OneStep" approach, but also with "TimeStop". Using binary trees as event tables for each subdomain, the total cumulative rates per subdomain is calculated as illustrated in Fig. 1(b). In the case of equi-space partitioning, as illustrated in Fig. 1(a) and (b), if the active sites are localized in the simulation cell, the total rates of each subdomain becomes imbalance. As mentioned in Sec. II, such imbalance may cause degradations in computational speed and accuracy in both "TimeStop" and "OneStep" approaches. To avoid such deceleration and inaccuracy, we introduce a dynamical space partitioning as depicted in Fig. 1(c) and (d). According to the previously calculated total rates, subdomain boundaries are dynamically adjusted to equalize total rates. Such adjustment

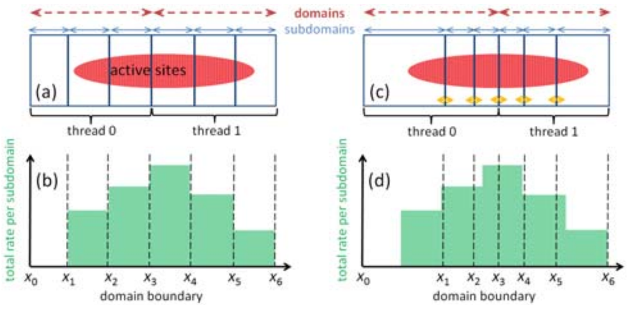

Fig. 1. (a) A lattice kMC simulation cell is schematically illustrated. Equi-space partitioning for $m = 2$ threads computation is drawn. One OpenMP thread has one domain. Each domain is divided into three subdomains. The active sites are localized in the cell as indicated with red elliptic region. (b) Due to the localization, total rates of subdomains are different. (c) By adjusting domain boundaries, (d) subdomains have equal total rates as indicated with green areas divided with dashed vertical lines.

or repartitioning can be easily realized and implemented by using ordered indexes of sites from one end to another end of the simulation cell. Another advantage of the use of such ordered index is separation of time evolution and partitioning algorithms from an object structure of kMC simulation on which they operate.

IV. Technical Details

Recently non-uniform memory access (NUMA) architecture has been widely used for shared-memory multiprocessor computers. As shown in Fig. 2, in a shared-memory computer, there are several slots—generally two or four slots—of CPUs and each CPU contains many cores. In the two parallel algorithms described above, one core handles one thread and one thread has three subdomains. the rate tables of each subdomain is placed on a "local" memory of NUMA, but occasionally a thread has to update the adjacent rate table placed on a "remote" memory, if an event occurs at the boundary between two subdomains. Generally, access to the "remote" memory required longer time than access to the "local" one, because the "remote" access has to go through an inter connection between CPU s. Therefore, with these parallel algorithms, it is difficult to get good parallelization efficiency beyond a single CPU.

The Array of Structures (AoS) approach is often implemented in simulation codes due to its handiness and flexibility [3]. For a class of problems including kMC, however, the performance of Structure of Arrays (SoA) is much better than that of AoS because SoA allows efficient use of vector units and cache of modern CPUs. Moreover, in lattice kMC simulations, it is found that placing attributes of neighboring sites close in memory gives higher performance than random placement. For this purpose, if the material has a diamond lattice (e.g. in the case of silicon) and the direction of slice can be determined arbitrary (e.g. in the case without periodic boundary condition), slice parallel to one of $\{111\}$ surfaces and indexing along the silicon zigzag network could bring good performance.

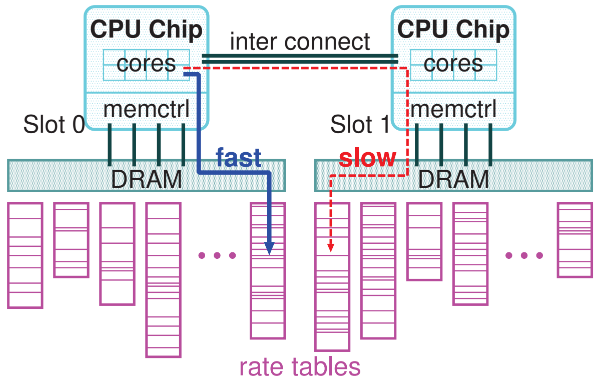

Fig. 2. Schematic illustration of Non-uniform memory access (NUMA) architecture. In a shared-memory computer, there are several slots (here, 2 slots) of CPUs and each CPU contains many cores (8 cores). CPUs are connected with inter connection(s) with each other. A CPU is connected to DRAM through a memory controller and memory channels. Read and write access to a rate table on the "local" DRAM is faster (blue thick solid arrow) than that of "remote" (thin red dashed arrow).

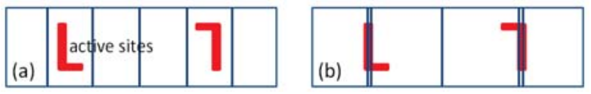

Fig. 3. Schematic illustration of an example case of active sites (red L-shaped regions) are localized on planes parallel to the slice planes. (a) Before partitioning. (b) After partitioning, subdomain(s) might become thinner than 5 bond lengths.

Finally, a limitation of automatic repartitioning is introduced because each slice has to be thicker than the diffusion jump and interaction distance (typically about 5 bond lengths). In very limited number of cases when active sites are localized on planes parallel to the simulation cell slice planes, this limitation shows up, as shown in Fig. 3. Typically, these situations can be avoided by proper choice of slicing direction.

V. Evaluation of parallelization efficiency

To evaluate parallelization efficiency of "TimeStop" and "OneStep" approaches and the repartitioning, simple hypothetical dopant diffusion simulations are performed with one 18-core Intel Xeon Gold 6154 CPU. A simulation cell of $512 \times 64 \times 63$ simple cubic lattice with periodic boundary condition is assumed and a bulked $384 \times 64 \times 3 = 73{,}728$ dopant atoms are initially placed at the center of the cell. Therefore, initially, there are regions without dopant atoms at the two ends of the simulation cell in $\pm x$-directions. The dopant atom can be diffused to 6 directions with diffusion rate of 1 T^{-1}, where T is arbitrary unit of time. Consequently, along a kMC simulation, dopant atoms gradually blot from the initial center region into $\pm x$- and $\pm z$-directions of the simulation cell. Each lattice kMC simulation is performed until $t_{\mathrm{sim}} = 500$ T is reached. For the two parallelization approaches, equi-space, initial-only and dynamical partitioning are tested. Initial-only means that repartitioning is performed only once in the beginning of a simulation. For dynamical

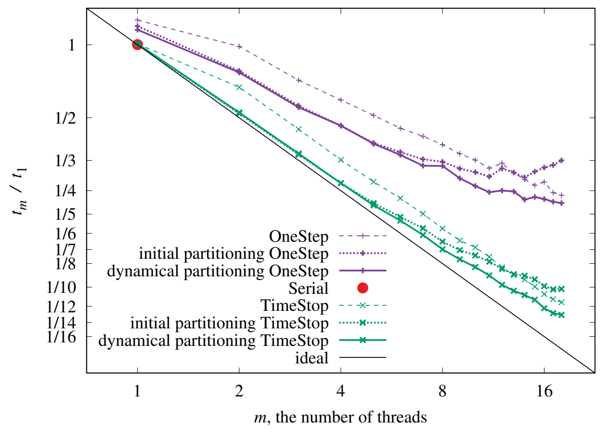

Fig. 4. m-thread parallelized computational times t_m are compared to serial computational time t_1 (red circle). For two parallel algorithms "OneStep" (three purple lines with $+$ marks) and "TimeStop" (three green lines with $\times$ marks), equi-space (thin dashed lines), initial-only (thick dotted lines) and dynamical (thick solid lines) partitioning are tested. A thin black straight line shows ideal parallelized efficiency.

partitioning, repartitionings to equalize total rates of each subdomain are performed 5 times at 0, 101, 202, 303, 404 T according to the total rates of previous iteration. $t_{\mathrm{stop}} = 10^{-2}$ T is used for every "TimeStop" algorithm simulations. To bind OpenMP threads to cores within one CPU, `numactl` Linux command is used. The numbers of executed diffusion events are around 208 million within difference less than 1%. One-core serial calculation (red circle in Fig. 4) takes $t_1 = 345$ s. As shown in Fig. 4, the "TimeStop" parallelization approach has better efficiency compared to that of "OneStep" for all equi-space, initial-only and dynamical partitioning. Under this simulation condition and for both parallelization approaches, initial-only and dynamical partitioning improves parallelization efficiency up to $m = 7$–10 threads. However, for m larger than 10–12 threads, efficiency of initial-only partitioning get worse, while dynamical partitioning keeps better efficiency than equi-space partitioning. This is because, in the case of initial-only partitioning with large m, two subdomains at two ends, i.e. $I = 1$ and $I = 3m$ subdomains, are partitioned much larger than other subdomains in the beginning of a kMC simulation, because there are no dopant atoms in the ends initially. Along the simulation, dopant atoms are diffused into the two large end subdomains, but end subdomains are never repartitioned, consequently imbalance in total rates, i.e. load imbalance, get bigger and bigger. Finally, it should be noted that determination and optimization of the period of repartitionings and t_{stop} are remaining future tasks.

VI. Practical example of semiconductor process simulation

To confirm applicability of the partitioning method in parallelized lattice kMC simulations on practical semiconductor processes, we tested "OneStep" approach with epitaxial silicon growth at the source/drain (S/D) of three array of FinFETs

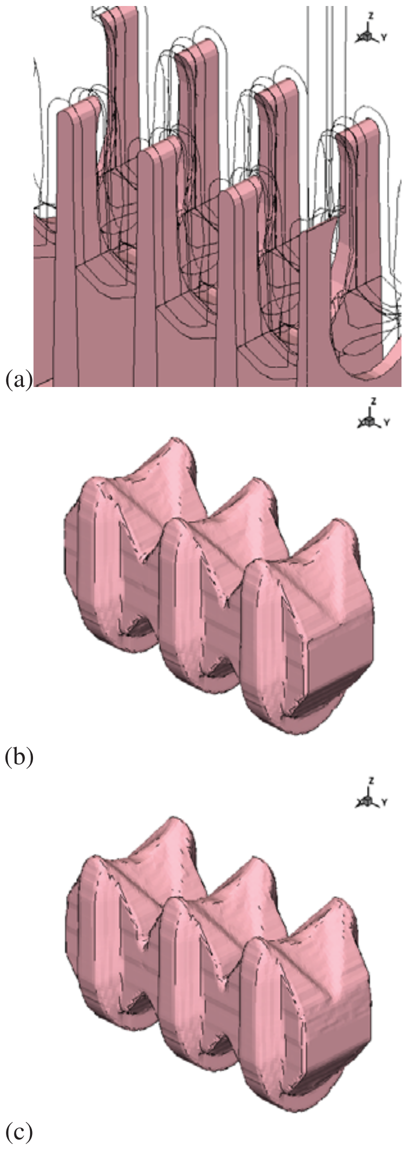

Fig. 5. (a) Initial configuration of recessed fins of FinFETs before epitaxial silicon growth to make sources/drains (S/D). Three fins are cut out from the center of this configuration. Growth starts from inside U-shaped structures. (b) Epitaxially grown connected three S/D simulated with a serial simulation. (c) That with a 4-times repartitioning parallelized simulation with 6 threads. In (a)-(c), surfaces surrounding silicon atoms are drawn.

in a $54 \times 90\ \mathrm{nm}^3$ simulation cell, as an example, as shown in Fig. 5. Other conditions of this epitaxial growth simulation are: at 700°C, during 600 s, up to the second nearest neighbors are counted for the determination of event rates, stuck atoms are assumed not to migrate on the growing surface, etc. Regrettably, The "TimeStop" parallelization approach and data structure optimization described in Sec. IV have not yet been implemented in our in-house simulator.

Timing results are plotted in Fig. 6. One-core serial calculation (red circle in Fig. 4) takes $t_1 = 45$ s. The numbers of executed epitaxial growth events are around 4.59 million within difference less than 1%. Unfortunately, the repartitioning results do not have better parallelization efficiency compared to the equi-space partition one. This is because reconstruction of rate table in repartitioning is time consuming. We expect that data structure optimization described in Sec. IV is also required to improve parallel efficiency. However, as shown in Figs. 5 (b) and (c), we can get identical results in two with/without repartitioning epitaxial growth lattice kMC simulations. Therefore, we believe that, with future optimization, the repartitioning method is effective to speed-up parallelized lattice kMC simulations of practical semiconductor processes.

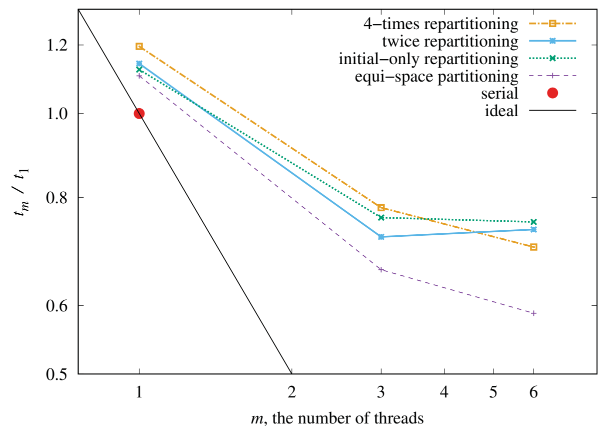

Fig. 6. m-thread parallelized computational times t_m are compared to serial computational time t_1 (red circle). "OneStep" parallel algorithm is used. equi-space (purple thin dashed line), initial-only (green thick dotted line) and twice (sky-blue thick solid line) 4-times (yellow chain line) partitioning are tested. A thin black straight line shows ideal parallelized efficiency.

VII. Summary

In this work, we have proposed a new dynamical space partitioning method which can speed up parallelized lattice kMC simulations. Capability of the new partitioning method is confirmed with two earlier parallelization approaches using a simple hypothetical dopant diffusion. Technical details on NUMA and data structures for efficient parallelization are discussed. It has been also shown that, in certain cases, not only initial partitioning but also occasional dynamical partitionings is effective. Applicability of the partitioning method is also confirmed with an example of epitaxial growth at S/D of FinFETs, though optimization of the method is left as a remaining issue.

References

[1] S. Plimpton, C. Battaile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, X. Zhou, C. G. Cardona, and A. Slepoy, "Crossing the mesoscale no-man's land via parallel kinetic Monte Carlo," *Sandia Report*, vol. SAND2009-6226, 2009.

[2] I. Martin-Bragado, J. Abujas, P. L. Galindo, and J. Pizarro, "Synchronous parallel Kinetic Monte Carlo: Implementation and results for object and lattice approaches," *Nuclear Instruments and Methods in Physics Research Section B*, vol. 352, pp. 27–30, 2015.

[3] H. Homann and F. Laenen, "SoAx: A generic C++ structure of arrays for handling particles in HPC codes," *Computer Physics Communications*, vol. 224, pp. 325 – 332, 2018.