Solid-State Electronics 199 (2023) 108520

Contents lists available at ScienceDirect

Solid State Electronics

journal homepage: www.elsevier.com/locate/sse

GPGPU MCII for high-energy implantation

Fumie Machida [a,*], Hiroo Koshimoto [a], Yasuyuki Kayama [a], Alexander Schmidt [b], Inkook Jang [b], Satoru Yamada [a], Dae Sin Kim [b]

[a] DS2 Lab, DS Center, Samsung R&D Institute Japan (SRJ), Tsurumi-ku, Yokohama 230-0027, Japan
[b] CSE Team, Innovation Center, Samsung Electronics Co., Ltd., Hwasung-si, Gyeonggi-do 18448, Korea

ARTICLE INFO

Keywords:
GPGPU acceleration
Ion Implantation
Monte Carlo Simulation

ABSTRACT

In this paper, we develop a GPGPU acceleration methodology for the Binary-Collision-Approximation based Monte Carlo ion implantation simulation (MCII). Our proposed method avoids the branch-divergence issue which comes from the difference of material crystallinity for the structure with multiple materials. We also introduce an efficient scheme to mitigate the side effect for damage accumulation due to massive parallelization of simulation. Our demonstration of high energy implantation into CIS structure shows almost 40x speed-up compared to CPU implementation of MCII. We conclude that GPU-MCII is effective for acceleration of Monte Carlo simulations with high energy implantation e.g. deep photodiode or well isolation formation.

1. Introduction

Ion implantation is one of the main methods to introduce dopants in semiconductor technology and high energy implantation was used for over 40 years for many industrial applications [1], including formation of buried dopant layers for bipolar transistors, deep photodiodes for CMOS image sensors and deep well isolation implantations. The Binary-Collision-Approximation (BCA) based Monte Carlo (MC) simulation is widely used for prediction of ion implantation profiles and optimization of semiconductor processing conditions. Various physical models have been proposed and improved by many researchers so far [2–6], and many efforts have been made to reduce the calculation time of the BCA-based MC simulators [7–9]. Nevertheless, the calculation time is still an issue for high energy implantations because of many collision events that have to be considered within BCA MC simulation.

To resolve this problem, we have developed a GPU implementation of the BCA-based MC simulator (GPU-MCII) with NVIDIA's Compute Unified Device Architecture (CUDA). GPUs are capable of massively parallel processing with thousands of cores. However, they have some restrictions due to the architecture. In this paper, the GPU performance, the drawbacks in GPU implementation of MCII, and the improvement techniques are shown.

2. Methodology

2.1. MCII

BCA MC simulation flow considers that energetic particles travel through material losing energy until they come to rest due to successive collisions with the target material atoms nuclei and inelastic electronic energy losses. Thus, a MC simulation cycle includes the following calculation steps:

(1) search target atoms,
(2) calculate nuclear/electronic energy loss and direction of a projectile ion after a collision based on BCA,
(3) update next particle position assuming that it moves in a straight line,
(4) calculate the amount of energy transferred during a collision and generated crystal damage (in crystalline materials only),
(5) check criterion of trajectory splitting [9],
(6) update dopant profile.

This MC cycle is repeated until the energy of a particle reaches the cut-off energy (5 eV). Due to a large number of real ions that are typically implanted in semiconductor technology (dose is in a range of $10^{11}\text{cm}^{-2} \sim 10^{15}\text{cm}^{-2}$), in simulation, each MC particle corresponds to many real ions and so implanted dose and damage are scaled

* Corresponding author.
E-mail address: f.machida@samsung.com (F. Machida).

https://doi.org/10.1016/j.sse.2022.108520

Available online 6 November 2022

accordingly. It is critically important to track dynamic changes of lattice damage since they are affecting subsequent MC particle collision probabilities at step (1) of the loop [6]. It is considered that ions move in straight lines between collisions with target atoms and free path length is calculated for amorphous and crystalline materials following method described in [6]. We are using ray tracing method [10,11] to update projectile position at step (3). Trajectory splitting at step (5) is one of the statistical enhancement methods [8,9], which splits a tracked MC particle into two when the criterion related to energy and dopant concentration at the particle position are satisfied. In this manner, profile fluctuations can be efficiently reduced especially at low concentration part (at the tail of implantation profile and near the structure surface in case of high energy implantation). The method is inevitable because suppression of dopant profile fluctuations without trajectory splitting needs an increase of statistics by more than an order of magnitude, leading to excessive simulation time.

Fig. 1 shows the simulation flow for an MC particle in the CPU implementation of BCA algorithm (CPU-MCII). In CPU-MCII, a loop of the MC cycles is executed sequentially for every MC particle. Even though the multicore CPU computes multiple MC particles in parallel, the impact on damage accumulation is limited because the number of parallel executions is small. However, it would be problematic with GPU due to massively parallelized execution (Section 3). Also, GPU implementation would need some way to handle the added MC particles by trajectory splitting because the buffer size cannot be flexibly changed during GPU execution. Furthermore, when there are many branch codes like "if/else" as shown in the MCII flow (Fig. 1), it is known that they could be a cause of degradation of performance for GPU.

2.2. Implementation

Implementing the MC cycle onto GPU without ingenuity introduces the branch divergence [12,13] which is the performance degradation caused by branch instructions. Because multiple GPU cores running at the same execution unit cannot execute different operations at once, the threads entering the other side at the branch are blocked. Consequently, the portion of effective GPU threads gets decreased. This is called the branch divergence. However, it is inevitable to have branching where a particle is moving from one type of material to another or where trajectory splitting happens. To avoid the branch divergence, we implemented a mechanism that we call the Dynamic Batch Charger (DBC). The DBC consists of the following treatments:

(a) decomposing the MC cycle into multiple GPU kernels removed the branch instructions (Fig. 2(a)),
(b) defining a GPU batch which is a tuple of the GPU kernels and the MC particles for a material crystallinity,
(c) binding a GPU batch with a chunk of GPU cores (a part of a GPU board),
(d) monitoring GPU batches, flushing out results of GPU batches and loading next GPU batches (Fig. 2(b)), and
(e) recharging the MC particles in the GPU batches flushed out (Fig. 3).

The treatment (c) is the key concept of the DBC, because this enables flexibility on GPU in exchange for a small performance loss. Later we discuss this performance loss with the benchmark. Additionally, we could change the number of MC particles dynamically by treatment (e). The MC particles still in flight are held for a next execution, and the MC particles splitting are copied to create more MC particles. In this way, the GPU-MCII carries out individual and variable calculations for a mass of MC particles at once.

2.3. Slowcoach scheme

In order to reduce the impact on damage accumulation as discussed in Section 2, we introduced the slowcoach scheme with multiple slowcoach approaches. Slowcoach approach is a method to use not full size of a GPU batch but the specified smaller size. It reduces GPU utilization, but generates damage little by little, avoiding unphysical effects of sudden damage change. The keys to obtain accurate profile are the optimal task size and the end condition of the slowcoach approach. To define them, we firstly take sample data with a small number of particles N_s using the slowcoach approach (Fig. 4). Using maximum damage density $C_{d,max}$ from the sample data and the damage threshold ($C_{d,max}$) for amorphization, we could estimate that the number of particles required for the damage density to reach to the threshold is $N_s \frac{C_{d,th}}{C_{d,max}}$,. And

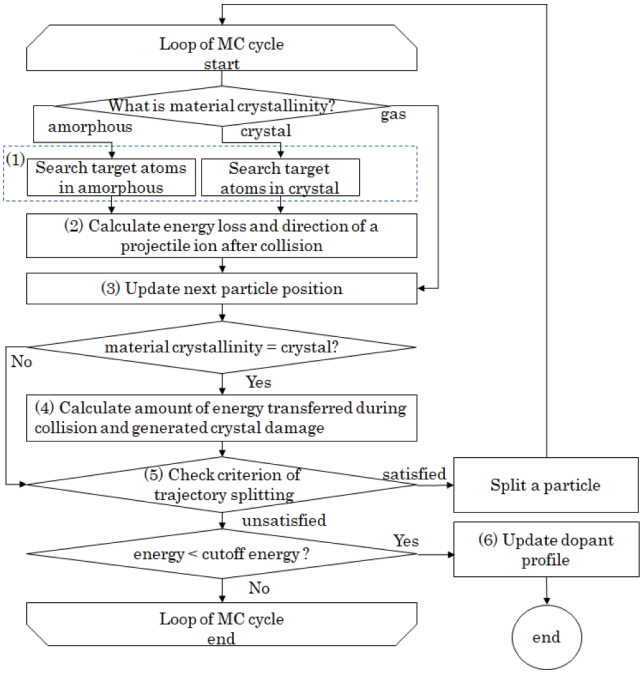

Fig. 1. Simulation flow of the CPU-MCII for a particle.

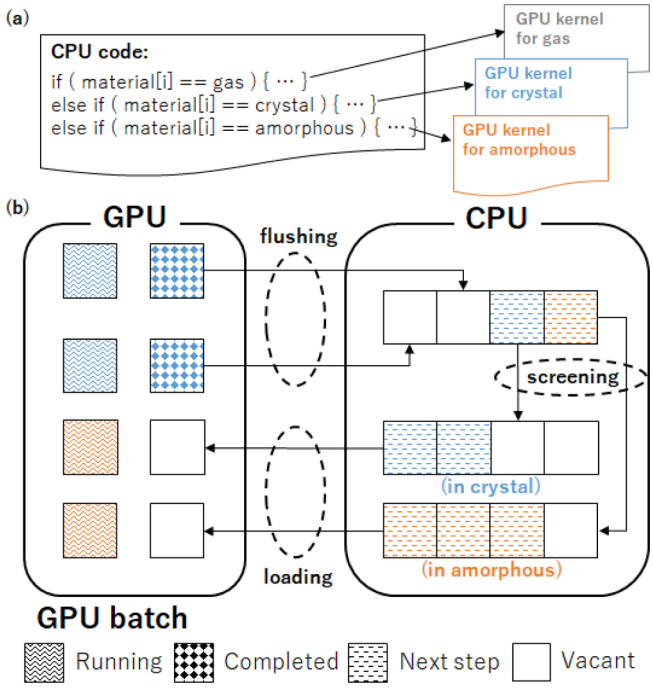

Fig. 2. Dynamically controlled GPU batches.

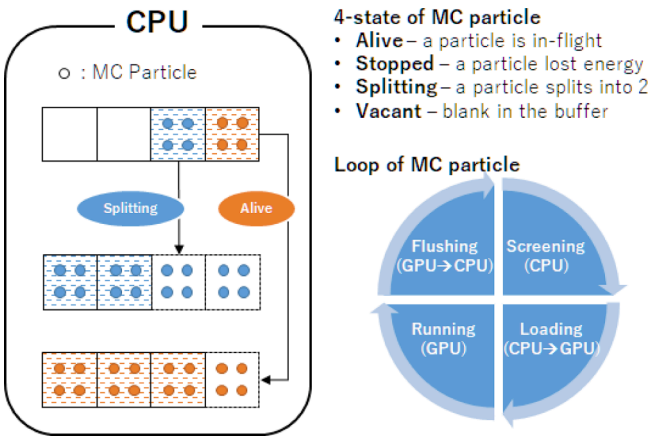

Fig. 3. Dynamically controlled MC particles.

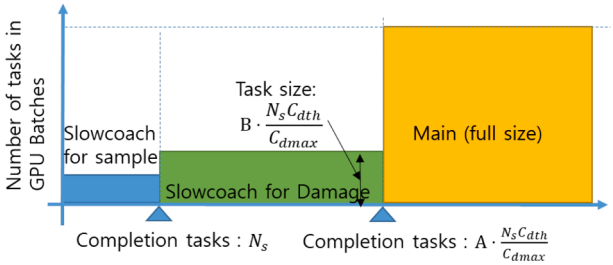

Fig. 4. Slowcoach approach schematic.

then, the number of completed tasks to end the slowcoach approach is defined by $A{\cdot}N_s\frac{C_{d,th}}{C_{d,max}}$. In the same way, the task size to be used in the slowcoach approach is defined by $B{\cdot}N_s\frac{C_{d,th}}{C_{d,max}}$. Where, A and B are user parameters, and $A \geq 1$ and $B < 1$ are desirable to generate damage slowly.

Fig. 5 shows Arsenic (As) profiles obtained by CPU-MCII (calibrated to internal SIMS data) and GPU-MCII. Without a slowcoach approach, the tail profile is suppressed compared to the CPU-MCII because the peak area is amorphized in the early stage due to the concentration of damage generation. In reality, at the early stage of implantation surface damage is small and some portions of ions have high chance of channeling, forming deep tail profile. When a slowcoach approach is used this effect is properly captured and the profile becomes similar to the result of CPU-MCII.

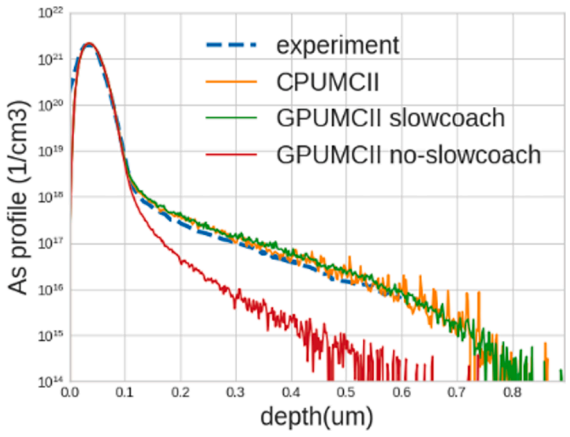

Fig. 5. Implanted profile with and without slowcoach, and experimental data [6] (dose = 8.0 × 10^{15}cm^{-2}, energy = 50 keV, A = 4, B = 0.1, tilt = 0°, (1 0 0)Si).

2.4. Benchmark

To evaluate the acceleration effect of DBC, we compared it with the ingenious GPUMCII which uses "if/else" in the kernel to switch material models. Tests were performed with Nvidia Tesla V100. For a single material structure as shown in Fig. 6(a), branch-divergence induced by differences of the material-dependent model doesn't occur. Therefore, for 50 keV implantation, a no-batch method takes almost the same calculation time as the DBC method (Fig. 7). On the other hand, for a structure with 2 materials as Fig. 6(b), DBC is faster than a no-batch method. Code profiling shows that for a "search target atom" kernel the warp execution efficiency increased from 14.7 % to 69.8 % when DBC is used. The difference becomes larger for high energy implantation, since for the simulations with large number of MC collision cycles DBC can effectively use vacant threads that have already been computed.

For more practical comparison with CPU-MCII, we've tested a deep implantation into a 3D CIS structure. Simulations were performed with Nvidia Tesla V100 and 8 cores on Intel Xeon Gold 6146 3.2 GHz. Fig. 8 shows the results of simulation of low dose (5 × 10^{11} cm^{-2}) Arsenic implantation with 3 MeV energy in 3D CIS structure. Simulation time of GPU-MCII is merely 4.0 % of CPU-MCII for 1 × 10^6 particles statistics. For 1 × 10^8 particles, it's reduced further to 2.4 % due to reduced relative overhead for simulation initialization (Table 1).

3. Summary

We developed the GPU-MCII method that utilizes the decomposition strategies to avoid the branch divergence performance degradation on GPU. We demonstrated that it can work not only for the branch divergence but also for the effective use of empty threads. Slowcoach approach algorithm was implemented in order to mitigate the all-at-once damage accumulation overestimation due to massively parallel GPU execution. Notwithstanding the fact that parallel performance of GPGPU-MCII can degrade for low energies and high doses, when frequent CPU-GPU data sharing is needed, our methodology is showing excellent results for high energy and low dose implantation steps simulation, which are important for CIS technology optimization. For typical process conditions and modeling setup about 40× simulation time acceleration can be achieved.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

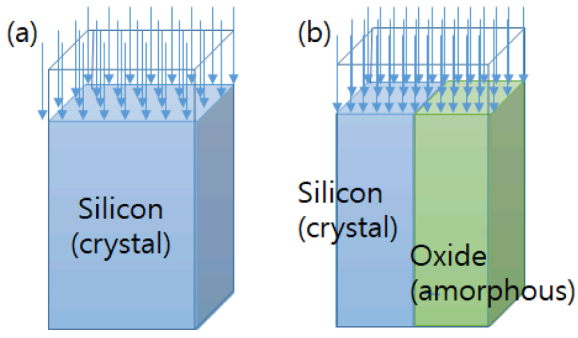

Fig. 6. (a) single material structure (b) 2 material structure.

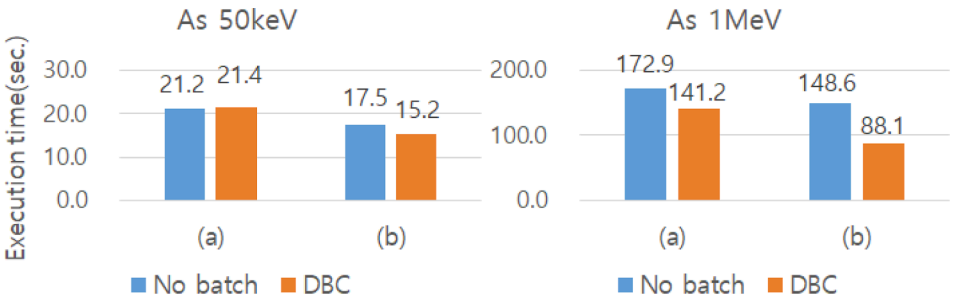

Fig. 7. Execution time for 50 keV/1MeV implantation for no-batch and DBC with structure (a), (b) in Fig. 6.

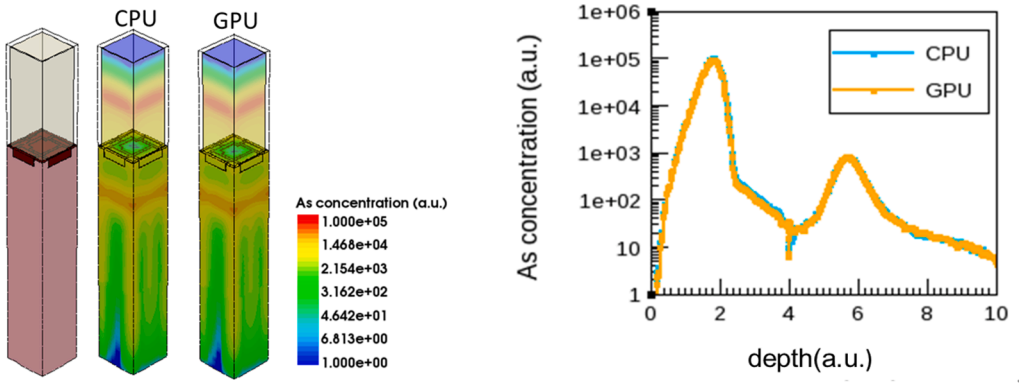

Fig. 8. Deep implantation of As into 3D CIS with 3 MeV energy, $5 \times 10^{11}\text{cm}^{-2}$, 284,638 mesh elements (right: 3D profile, left: extracted 1D profile).

Table 1
TAT comparison between CPU and GPU.

Number of particles	Calculation time[hour]		CPU × 8/GPU
	CPU × 8	GPU	
1×10^6	0.63	0.025	25.2
1×10^8	58.76	2.43	41.1

Data availability

No data was used for the research described in the article.

References

[1] Ziegler JF. High energy ion implantation. Nucl Instrum Methods Phys Res B 1985;6 (1-2):270–82.
[2] Ziegler JF, Biersack JP, Ziegler MD. SRIM – The stopping and range of ions in matter. James Ziegler 2008.
[3] Robinson MT, Torrens IM. Computer simulation of atomic displacement cascades in solids in the binary-collision. PRB 1974;9(12):5008–24.
[4] Norgett MJ, Robinson MT, Torrens IM. A proposed method of calculating displacement dose rates. Nucl Eng Des 1975;33(1):50–4.
[5] Hobler G, Pötzl H, Gong L, Ryssel H. Two-dimensional Monte Carlo Simulation of Boron Implantation in Crystalline Silicon. Simulat Semiconduct Dev Process 1991; 4:389–98.
[6] Tian S. Predictive Monte Carlo ion implantation simulator from subkeV to above 10 MeV. JAP 2003;93:5893–904.
[7] Wang G, et al. A computationally efficient target search algorithm for a Monte Carlo ion implantation simulator. Journal of Technology Computer Aided Design TCAD 1996:1–19. https://doi.org/10.1109/TCAD.1996.6449179.
[8] Bohmayr W, Burenkov A, Lorenz J, Ryssel H, Selberherr S. Statistical Accuracy and CPU Time Characteristic of Three Trajectory Split Methods for Monte Carlo Simulation of Ion Implantation. Simulat Semiconduct Proceedes 1995;6.
[9] Kubotera H, et al. Efficient Monte Carlo simulation of ion implantation into 3D FinFET structure. 20th International Conference on Ion Implantation Technology (IIT) 2014;1–4. https://doi.org/10.1109/IIT.2014.6939999.
[10] Fang Q, Yan S. GPU-accelerated mesh-based Monte Carlo photon transport simulations. J Biomed Opt 2019;24(11).
[11] A. S. M. Shevtsov, A. Kapustin, Ray-Triangle intersection algorithm for modern CPU architectures, in: GraphiCon'2007, 2007, pp. 33–39.
[12] W. W. Fung, I. Sham, G. Yuan, T. M. Aamodt, Dynamic Warp Formation and Scheduling for Efficient GPU Control Flow, in: 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007), 2007, pp. 407–420. doi:10.1109/MICRO.2007.30.
[13] H. Wong, M.-M. Papadopoulou, M. Sadooghi-Alvandi, A. Moshovos, Demystifying GPU microarchitecture through microbenchmarking, in: 2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), IEEE, 2010, pp. 235–246.