# Memory-efficient Particle Annihilation Algorithm for Wigner Monte Carlo Simulations

P. Ellinghaus, M. Nedjalkov, and S. Selberherr
Institute for Microelectronics, TU Wien, Gußhausstraße 27–29/E360, 1040 Wien, Austria
E-mail: {ellinghaus | nedjalkov | selberherr}@iue.tuwien.ac.at

***Abstract*—The Wigner Monte Carlo solver, using the signed-particle method, is based on the generation and annihilation of numerical particles. The memory demands of the annihilation algorithm can become exorbitant, if a high spatial resolution is used, because the entire discretized phase space is represented in memory. Two alternative algorithms, which greatly reduce the memory requirements, are presented here.**

## I. Introduction

The roughness of the interface between materials significantly influences the electrical performance of modern device architectures, like gate-all-around nanowires, due to increased scattering. Therefore, it is of interest to model rough interfaces to investigate their effect in quantum transport simulations.

Interface roughness can be modeled by adding random perturbations to a smooth interface. The perturbations are statistically characterized by an autocorrelation function with parameters signifying the mean offset and the correlation length. An example is shown in Fig. 1. The mean offset can range between $0.1\,\mathrm{nm}$ and $0.3\,\mathrm{nm}$ for a Si/SiO$_2$ interface [1], [2]. To appropriately resolve such perturbations requires a very fine spatial resolution for the potential profile. The latter leads to very high memory demands for the annihilation algorithm, which is fundamental to the signed-particle Wigner Monte Carlo (SPWMC) method.

The following section will introduce the SPWMC method, used to solve the Wigner transport equation. Thereafter, two alternative annihilation algorithms will be presented, which significantly reduce the memory demands associated with the current version of the algorithm.

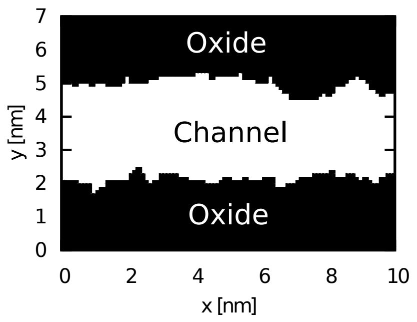

Figure 1: The geometry represents a $3\,\mathrm{nm}$ wide silicon channel between two oxide layers at a resolution of $0.1\,\mathrm{nm}$. The roughness of the Si/SiO$_2$ interface is characterized by an exponential auto-correlation function with a mean displacement, obtained from experiments in [1].

## II. Signed-particle Method

The SPWMC method has enabled the investigation of transient and stationary processes in multi-dimensional semiconductor structures, ranging from quantum-coherent to scattering-dominated transport. The simulations are based on the semi-discrete Wigner transport equation, which assumes a finite coherence length $(L)$ leading to a discretized $k$-space with a resolution of $\Delta k = \frac{\pi}{L}$, and for the one-dimensional case is given by

$$\frac{\partial f_w}{\partial t} + \frac{\hbar q \Delta k}{m^*}\frac{\partial f_w}{\partial x} = \sum_{q=-K}^{K} V_w\left(x, q-q', t\right) f_w\left(x, q', t\right). \quad (1)$$

Here, $q$ is an index which refers to the quantized momentum, i.e. $p = \hbar\left(q\Delta k\right)$.

Equation (1) can be reformulated as Fredholm integral equation of the second kind and solved using the Monte Carlo technique with the signed-particle method [3]. Each particle carries a $+$ or $-$ sign which conveys the quantum information. The particles evolve freely according to their momentum and generate additional particles in pairs with $+$ and $-$ signs with different momenta. This generation process captures the effects of the potential profile and the statistics (generation rate and momentum offsets) are dictated by the Wigner potential, defined as

$$\begin{aligned} V_w\left(x, q\right) &\equiv \frac{1}{i\hbar L}\int_{-L/2}^{L/2} ds\, e^{-i2q\Delta k\cdot s}\delta V\left(s; x\right); \quad (2) \\ \delta V\left(s; x\right) &\equiv V\left(x+s\right) - V\left(x-s\right). \end{aligned}$$

This is akin to a Fourier transform of the potential differences within the coherence length $L$ around a point $x$. The Wigner potential encapsulates the higher-order derivatives of the potential and not only its first derivative, i.e. the electric field, as in the classical Boltzmann equation.

For potential differences in the order of $100\,\mathrm{meV}$, the particle generation rate typically lies in the order of $10^{15}\,\mathrm{s}^{-1}$. This fast rate makes the exponential increase in the number of particles become numerically debilitating after evolving the system for only a few femtoseconds. The concept of particle annihilation is used to counteract the dramatic increase in the number of particles and is discussed in the next section.

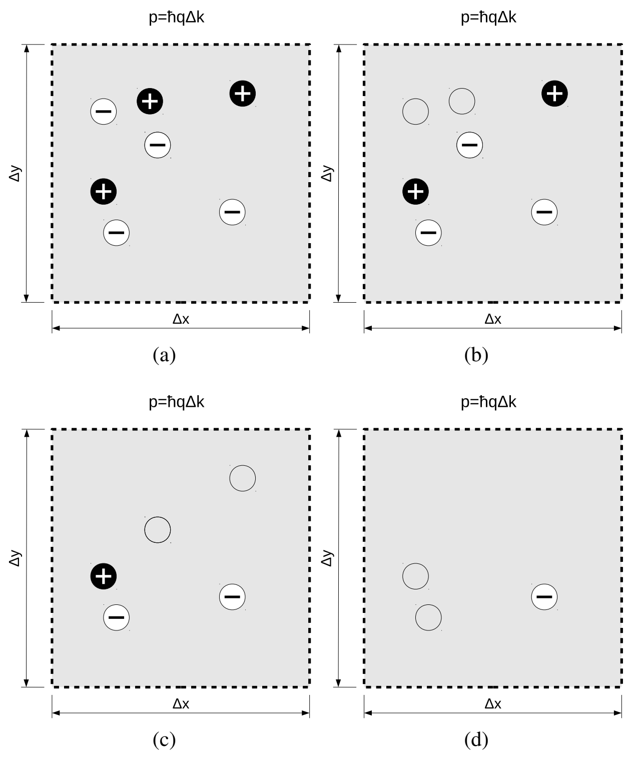

Figure 2: Illustration of the pair-wise annihilation of particles in a cell of the phase space (a) - (d), representing a two-dimensional area $\Delta x \Delta y$ and a fixed momentum $\mathbf{p} = \hbar \mathbf{q} \Delta k$. All particles within a cell are considered indistinguishable.

## III. Annihilation Algorithms

The annihilation concept entails dividing the phase space into many cells; in the case of a simulation with two spatial dimensions and a three-dimensional $k$-space, each cell represents a volume of $\Delta x \Delta y \Delta k^3$. Within each cell particles with opposite signs annihilate each other and cease to exist. This is justified as every particle pair has the same probabilistic future, but their contributions to the calculation of any physical quantity will cancel out. Consider a cell with $A$ particles with a positive sign and $B$ particles with a negative sign, which are summed up to yield a remainder of particles, $R = A - B$; $|R|$ particles, each carrying the sign of $R$, are regenerated within the cell and the evolution continues. The annihilation procedure is illustrated in Fig. 2.

The $|R|$ particles which survive the annihilation procedure should, ideally, recover the information represented by the$(A + B)$ particles before the annihilation took place. Since the momenta are quantized and a single value is shared amongst all particles within a cell, the distribution in the $k$-space is recovered after annihilation. The positions of the particles, however, are real-valued and require additional consideration. To avoid a numerical diffusion taking place in the regeneration process, the mean position of all particles in a cell should be recorded before the annihilation [4]. The particles are uniformly regenerated around the mean of each cell over an area $\Delta x \Delta y$.

In the following, three different annihilation algorithms are presented, starting with the conventional implementation, which serves as a reference to compare the two newly introduced algorithms against.

### A. Reference implementation

The conventional annihilation algorithm represents the entire phase space using an array of integers in order to record the sum of signs in each cell. Additionally, arrays are needed to record the absolute number of particles and the sum of position values in each cell in order to calculate the mean position before the annihilation. The associated memory requirements quickly become exorbitant in multi-dimensional simulations. This especially holds true, if a high spatial resolution is required, since the number of cells in the phase space increases with the power of the dimensionality of the phase space (the spatial resolution also affects the number of $k$-values which must be retained to ensure a unitary Fourier transform in (2)). As an example, a $10\,\mathrm{nm} \times 10\,\mathrm{nm}$ domain and a three-dimensional $k$-space with a coherence length of $10\,\mathrm{nm}$ results in an array size exceeding $80\,\mathrm{GB}$ at a resolution of $0.1\,\mathrm{nm}$. This clearly exceeds the memory available on common workstations and even single computation nodes in high-performance computing cluster. A distributed-memory (MPI) approach addresses these large memory requirements through a spatial domain decomposition [5], which allows several nodes to share the computational load. However, the computational demands of the SPWMC simulator allow it to be run on a typical workstation, therefore, its memory demands should also follow suit.

### B. Distribution fitting

A possibility to reduce the memory requirements of the annihilation algorithm is to reduce the spatial resolution of the grid on which the particles are recorded for annihilation; the resolution of the $k$-values and the potential mesh remain unaltered. The concept is depicted in Fig. 3. To counteract the loss in resolution, the spatial distribution of the particles in the enlarged cell is fitted to a statistical distribution before annihilation ensues. The obtained distribution is then used to regenerate the particles which remain after the annihilation.

Fig. 4 compares the regeneration of particles, annihilated on a coarsened grid, using a uniform distribution or a Gaussian distribution around the pre-annihilation mean position of particles. The former follows the true solution much better than the Gaussian distribution which provides a poor approximation of the distribution in each cell; this is consistent with the observations made in [4]. The use of a Gaussian distribution artificially re-introduces information which conflicts with the assumption made to perform annihilation, namely that all particles in a cell are considered to be indistinguishable regardless of their position. The uniform distribution best reflects this state of information. Irrespective of the distribution used, both approaches reduce the memory consumption in the example roughly by a factor of 16; the Gaussian distribution requires some extra memory and computation to calculate the additional moment of the distribution (the standard deviation).

### C. Ensemble sorting

The representation of the phase-space as an array to record the signs of particles is a direct reflection of the physical con-

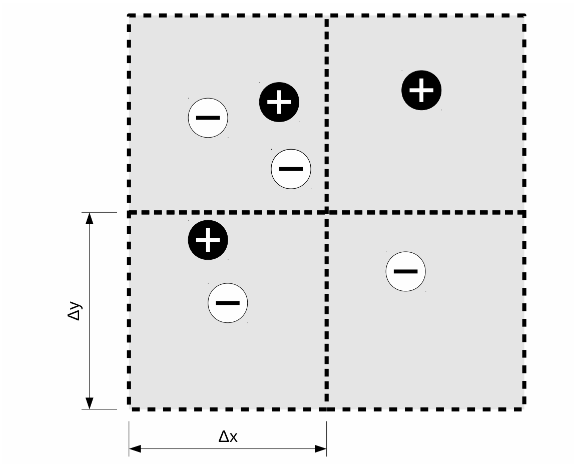

Figure 3: A distribution of $+$ and $-$ particles in four spatial cells (dashed lines). The gray cell, encapsulating all four cells, represents a cell of the coarsened phase-space grid used to perform annihilation. It is enlarged by a factor 2 in both directions, reducing the array size by a factor of four.

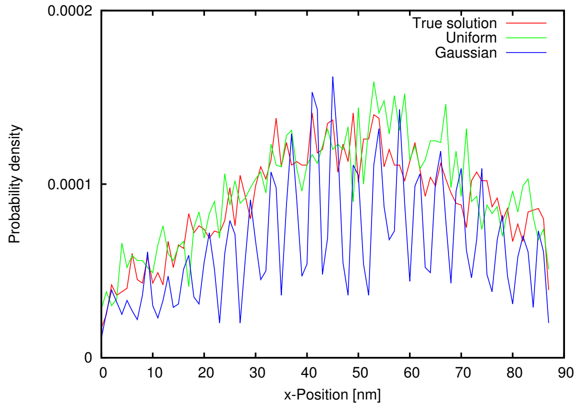

Figure 4: A slice of the two-dimensional probability density of a wave packet, evolving freely in a domain with a spatial resolution of $0.25\,\mathrm{nm}$, after 80 forced annihilation steps. The annihilation is performed on a coarsened grid with a $1\,\mathrm{nm}$ resolution and the particles are regenerated using uniform and Gaussian distributions. The 'true solution' – the evolution, when the annihilation step is omitted – is followed the best, when particles are regenerated by a uniform distribution around the mean particle position in each cell.

cept underlying particle annihilation. However, the annihilation concept can be also realized with an algorithm which avoids representing the phase space by an array, thereby completely avoiding the huge memory demands associated with it.

An integer index can be associated to the position and momentum attributed to each particle. These indices are mapped to a single integer $H$, uniquely identifying the cell of the phase space in which the particle resides:

$$(i_x, j_y, q_x, q_y, q_z) \rightarrow H. \tag{3}$$

All the particles with the same value of $H$ are in the same cell of the phase space and their signs must be accumulated. To

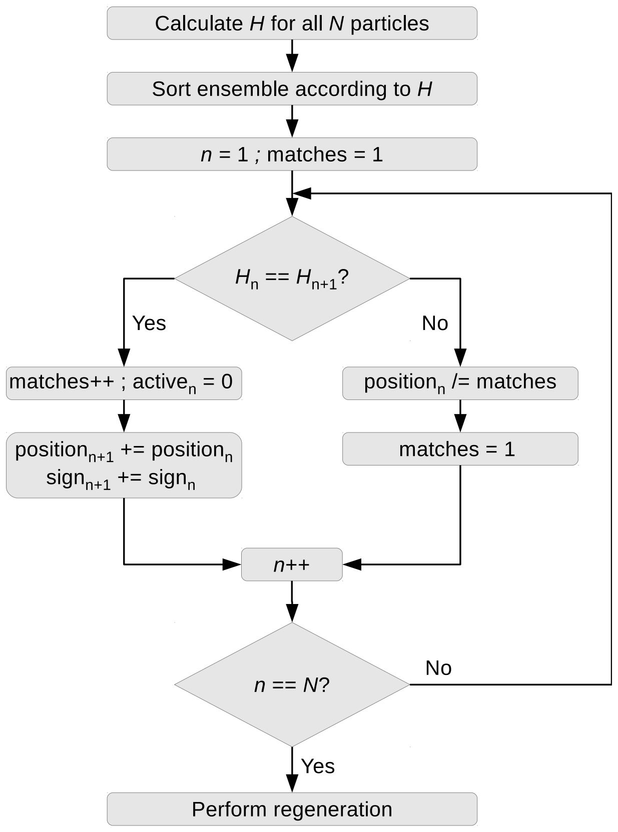

Figure 5: Flow-chart of annihilation algorithm based on ensemble sorting.

search an ensemble consisting of $N$ particles, to find particles with matching values of $H$, requires an algorithm with an $\mathcal{O}\left(N^2\right)$ time complexity. As $N$ can be several millions, the additional computation time is not tolerable.

The situation is greatly improved by first sorting the array, representing the particle ensemble, according to the values $H$. This can be efficiently performed by a quicksort algorithm which has a $\mathcal{O}\left(N \log_2 N\right)$ time complexity [6]. In the sorted particle ensemble, all particles in the same cell of the phase space (i.e. the same value of $H$) now appear consecutively in the array – this is also beneficial for the memory access speed. The sorted array allows the sum of signs and mean positions of the particles to be calculated in-place without the need of any additional memory. A flow-chart of the algorithm is shown in Fig. 5. The sorted array is iterated, if the value of $H$ for particle $n$ and $n+1$ are identical, a 'matches' counter is incremented, particle $n$ is deactivated (by a flag), and its sign and position is added to the values of particle $n+1$. This process continues until $H_n \neq H_{n+1}$, then the mean position of all the particles in the cell corresponding to $H_n$ is calculated by dividing the sum of the positions by the counter which is then reset.

Once the entire ensemble has been covered, the regeneration process commences. The information required for regeneration is stored in the fields for position and sign for

Table I: Simulation parameters for benchmark examples

| | A.1 | A.2 | B.1 | B.2 |
|---|---|---|---|---|
| Max. ensemble size [$\times 10^6$] | 20 | 10 | 5 | 5 |
| Annihilations performed | 23 | 210 | 24 | 33 |
| Simulation time [fs] | 100 | 100 | 100 | 100 |
| Domain [nm] | $100 \times 150$ | $100 \times 150$ | 100 | 100 |
| $\Delta x$ $(= \Delta y)$ [nm] | 1 | 1 | 1 | 1 |
| Coherence length $L$ [nm] | 30 | 30 | 30 | 60 |

Table II: Simulation times for different annihilation algorithms

| | Reference [s] | Sort-based [s] | Change |
|---|---|---|---|
| A.1 | 2041 | 2425 | +19% |
| A.2 | 1717 | 2028 | +18% |
| B.1 | 427 | 455 | +7% |
| B.2 | 508 | 548 | +8% |

particles which are still marked active (active$_n = 1$; *cf.* Fig. 5). The new particles are regenerated and stored in the original array, overwriting the particles in the array, which have been deactivated during the annihilation process. This allows the entire annihilation and regeneration process to be completed with an insignificant amount of additional memory.

The trade-off for the small memory footprint of this annihilation algorithm is the additional computation time required to sort the array of the particle ensemble. The calculation of the indices and the regeneration of particles is essentially identical between all the annihilation algorithms presented here. The impact of the sorting on the overall computation time of a simulation depends on i) the regularity of the annihilation (the generation rate) and ii) the threshold value for the number of particles in the ensemble ($N$) at which annihilation (sorting) occurs. To characterize the performance impact of the algorithm, the simulation time of two simulation examples are compared with different parameters, as listed in Table I. The results in Table II reveal the sorting-based annihilation algorithm increases the computation time between $7\%$ and $19\%$. If sufficient memory is available, there are no advantages to using this algorithm. However, it should be noted that in certain cases the memory demands of the conventional annihilation algorithm exceed the capacities of a workstation. Therefore, this algorithm makes two-dimensional Wigner Monte Carlo simulations accessible to users without extensive computational resources.

## IV. Conclusion

The debilitating increase in the memory requirements to perform the annihilation step in SPWMC simulations with a fine spatial resolution can be effectively remedied by alternative annihilation algorithms.

A reduction of the spatial resolution of the phase space grid used for annihilation reduces the memory requirements. By fitting the pre-annihilation spatial distribution of particles in a cell to a statistical distribution, the loss in resolution can be mitigated. Under the assumptions made for annihilation the uniform distribution is the best-suited; other common statistical distributions, like Gaussians, are ill-suited for the fitting.

The annihilation algorithm based on ensemble sorting eradicates the memory demands almost entirely. The trade-off is a slight increase in computation time, which depends on the parameters of the particular simulation problem.

The presented advancements in the annihilation algorithm allows two-dimensional Wigner Monte Carlo simulations to be performed on conventional workstations, also when using high-resolution meshes, which greatly improves the accessibility of multi-dimensional time-dependent quantum transport simulations.

## References

[1] S. M. Goodnick, D. K. Ferry, C. W. Wilmsen, Z. Liliental, D. Fathy, and O. L. Krivanek, "Surface Roughness at the Si(100)-SiO$_2$ Interface," *Phys. Rev. B*, vol. 32, pp. 8171–8186, 1985.

[2] D. Vasileska and D. K. Ferry, "Scaled Silicon MOSFETs: Universal Mobility Behavior," *Electron Devices, IEEE Transactions on*, vol. 44, no. 4, pp. 577–583, 1997.

[3] M. Nedjalkov, P. Schwaha, S. Selberherr, J. M. Sellier, and D. Vasileska, "Wigner Quasi-Particle Attributes - An Asymptotic Perspective," *Applied Physics Letters*, vol. 102, no. 16, pp. 163113–1–163113–4, 2013.

[4] P. Ellinghaus, M. Nedjalkov, and S. Selberherr, "Optimized Particle Regeneration Scheme for the Wigner Monte Carlo Method," in *Lecture Notes in Computer Science, Vol. 8962*, pp. 27–33, Springer International Publishing, 2015.

[5] P. Ellinghaus, J. Weinbub, M. Nedjalkov, S. Selberherr, and I. Dimov, "Distributed-Memory Parallelization of the Wigner Monte Carlo Method Using Spatial Domain Decomposition," *Journal of Computational Electronics*, vol. 14, no. 1, pp. 151–162, 2015.

[6] C. A. R. Hoare, "Algorithm 64: Quicksort," *Commun. ACM*, vol. 4, no. 7, pp. 321–322, 1961.