Efficient deterministic parallel simulation of 2D semiconductor devices based on WENO-Boltzmann schemes

José Miguel Mantas [a] María José Cáceres [b]

[a] *Depto. de Lenguajes y Sistemas Informáticos.*

E.T.S. Ing. Informática y Telecomunicaciones, Univ. Granada, 18071 Granada.

jmmantas@ugr.es

[b] *Depto. de Matemática Aplicada.*

Facultad de Ciencias, Univ. Granada, 18002 Granada.

caceresg@ugr.es

Abstract

A flexible parallel deterministic solver of the Boltzmann-Poisson system for 2D semiconductor device simulation on computer clusters is presented. The simulator is obtained by parallelizing a previously proposed numerical scheme based on high order finite difference Weighted Essentially Non-Oscillatory (WENO) schemes. Although the underlying numerical scheme presents important advantages over Direct Simulation Monte Carlo methods, this scheme imposes very high demands of computing power. Due to this, the parallelization of the different calculation phases in the numerical scheme has been tackled. The data subdomain which demands most

of the computational workload has been suitably distributed among the processors and several parallel design decisions has been taken in order to achieve good performance. Moreover, the resultant parallel application can be easily adjusted to simulate a wide range of devices and could be easily used by engineers without mathematical background about the underlying numerical scheme. The parallel algorithm has been implemented in C++ augmented with calls to MPI functions and functions of optimized linear algebra libraries. Several experiments have been performed by simulating particular MOSFET and DG-MOSFET devices on a SMP cluster in order to show its efficiency.

Key words: semiconductor simulation, parallel numerical algorithms, finite difference weighted essentially non-oscillatory schemes, high performance cluster computing

---

1 Introduction

The computer simulation of semiconductor devices is an important area in the computer aided design of electronic circuits. The electron transport in these devices, at a mesoscopic level, can be modeled by the Boltzmann Transport Equation (BTE) for semiconductors in the semi-classical approximation:

$$\frac{\partial f}{\partial t} + \frac{1}{\hbar}\nabla_{\mathbf{k}}\varepsilon \cdot \nabla_{\mathbf{x}} f - \frac{\mathfrak{e}}{\hbar}\mathbf{E} \cdot \nabla_{\mathbf{k}} f = Q(f), \tag{1}$$

where f represents the electron probability density function (**pdf**) in phase space $\mathbf{k}$ at the physical location $\mathbf{x}$ and time t. Physical constants $\hbar$ and $\mathfrak{e}$ denote the Planck constant divided by 2π and the positive electric charge, respectively. The energy-band function ε is given by the Kane non-parabolic

band model, which is a non-negative continuous function of the form:

$$\varepsilon(\mathbf{k}) = \frac{1}{1+\sqrt{1+2\,\dfrac{\tilde{\alpha}}{m^*}\hbar^2\,|\mathbf{k}|^2}}\frac{\hbar^2}{m^*}\,|\mathbf{k}|^2\,, \tag{2}$$

where m^* is the effective mass and $\tilde{\alpha}$ is the non parabolicity factor.

The right-hand side of Equation (1) models the interaction of electrons with lattice vibrations of the crystal and is written as

$$Q(f)(t,\mathbf{x},\mathbf{k}) = \int_{\mathbb{R}^3}\left[S(\mathbf{k}',\mathbf{k})f(t,\mathbf{x},\mathbf{k}') - S(\mathbf{k},\mathbf{k}')f(t,\mathbf{x},\mathbf{k})\right]\,d\mathbf{k}'\,, \tag{3}$$

where $S(\mathbf{k},\mathbf{k}')$ is the transition probability from state $\mathbf{k}$ to $\mathbf{k}'$ per unit of time for each scattering mechanism. Therefore, the collision term $Q(f)$ depends on the semiconductor material by means of the different scattering mechanisms which are considered. Interactions between electrons are not directly included in the right hand side since we consider low probability densities and the quadratic terms can be neglected. However, we take into account the electrostatics produced by the electrons and the dopants in the semiconductor, since the electric field is selfconsistently computed by the Poisson equation:

$$\nabla_{\mathbf{x}}\left[\epsilon_r(\mathbf{x})\,\nabla_{\mathbf{x}}V\right] = \frac{\mathfrak{e}}{\epsilon_0}\left[\rho(t,\mathbf{x}) - N_D(\mathbf{x})\right], \qquad \mathbf{E} = -\nabla_{\mathbf{x}}V \tag{4}$$

where ϵ_0 is used to denote the dielectric constant in a vacuum, $\epsilon_r(\mathbf{x})$ represents the relative dielectric function dependent on the material (which could be different depending on the position $\mathbf{x}$), $N_D(\mathbf{x})$ is the doping profile, V is the electric potential and $\rho(t,\mathbf{x}) = \displaystyle\int_{\mathbb{R}^3} f(t,\mathbf{x},\mathbf{k})\,d\mathbf{k}$ is the electron density.

Equations (1)-(4) are called the Boltzmann-Poisson system for electron transport in semiconductors.

The main numerical tools which are used in practice by electronic engineers to simulate this Boltzmann-Poisson system have been based on Direct Simulation Monte Carlo (DSMC) approach. However, this approach presents some drawbacks: a) presence of noise in the numerical solution, b) it does not show the evolution of the distribution density until the stationary state and therefore it does not capture all the moments which someone could need, and, c) it does not approximate almost empty regions in the device since there are not enough particles to obtain acceptable statistics. These disadvantages do not appear in deterministic simulations for the full Boltzmann-Poisson system. However, the main problem to simulate directly the Boltzmann-Poisson system is the fact that the system has 7 dimensions (6 space-momentum variables and time), which produces numerical simulations with high demands for computational power. To overcome this problem, several approximated systems and corresponding deterministic numerical methods have been proposed in literature (see the review about deterministic kinetic solvers for semiconductors devices in [3]). However, currently there are powerful and low cost parallel machines like the clusters of Symmetric Multiprocessors (SMPs), which makes it possible, by deriving the suitable parallel software, to obtain simulation results in reasonable response times. There exist proposals of parallel deterministic solvers for semiconductor device simulation, both for distributed memory machines [9,2] and shared memory multiprocessors [6]. However, these proposals do not solve the full Boltzmann Equation for the transport but they are based on macroscopic models like hydrodynamic and drift-diffusion models. The aim of this paper is to show a flexible parallel WENO-Boltzmann solver for a wide family of 2D semiconductor devices. This work completes the preliminary one in [10], where a parallel solver for a MESFET device was presented, using the deterministic solver developed in [4,5]. The parallel solver in [10] was devel-

oped only for a MESFET device. The program structure was designed using the particularity of the MESFET geometry and the authors were focused on the experimental results, without a description of the main design aspects. Therefore, the program had to be modified to simulate every different device. Here, we overcome this particularity by presenting a general improved parallel design and implementation of the numerical scheme proposed in [5], which is a general parallel solver for a wide range of devices. The main improvements with respect to [10] also concern the Poisson solver, which is based on a direct linear system solver (replicated on every processor), and a more efficient implementation of the interprocessor communication. Moreover, the software presented in [10] is restricted to users who are familiar with it. However, this new application makes it possible that any interested person, without previous knowledge about the underlying numerical scheme and the parallelization, can take advantage of this solver by exploiting efficiently a parallel architecture to simulate a particular semiconductor device.

In the next section, we describe the underlying numerical scheme, details of that section are included in an Appendix. Section 3 introduces the semiconductor device representation, which enables an efficient and flexible implementation of the numerical scheme. Section 4 describes the design of our parallel solver. In Section 5, we show and analyze the results obtained when MOSFET and Double Gate MOSFET devices are simulated on a cluster of SMPs. Finally, in Section 6 we collect the main conclusions of the work and present the lines of further work.

2 WENO-Solver for the Boltzmann-Poisson system

The WENO-solver for the Boltzmann-Poisson system (1)-(4) in 2D physical space simulates another system, which is obtained from the previous one after a dimensionless process and a pseudo-spherical change of variables in the wave vector $\mathbf{k}$, such as was done in [4,5]. Of course, the results can be translated to the original Boltzmann-Poisson system by going backwards in the process. The new unknown is

$$\Phi(t, x, y, \omega, \mu, \phi) = s(\omega) f(t, x, y, \omega, \mu, \phi),$$

where x and y are the spatial coordinates, $\omega \geq 0$ is a dimensionless energy, $\mu \in [-1, 1]$ is the cosine of the angle with respect to x-axis and $\phi \in [0, 2\pi]$ is the azimuthal angle and $s(\omega) = \sqrt{\omega(1 + \alpha_\kappa \omega)}(1 + 2\alpha_\kappa \omega)$ (where α_κ is introduced in [4]). Then, a dimensionless Boltzmann equation in the 2D physical space is written in conservative form as:

$$\frac{\partial \Phi}{\partial t} + \frac{\partial}{\partial x}(a_1 \Phi) + \frac{\partial}{\partial y}(a_2 \Phi) + \frac{\partial}{\partial \omega}(a_3 \Phi) + \frac{\partial}{\partial \mu}(a_4 \Phi) + \frac{\partial}{\partial \phi}(a_5 \Phi) = s(\omega) C(\Phi), \tag{5}$$

where $C(\Phi)$ represents the dimensionless collision operator, in the new variables (see Appendix for more details).

The strategy to obtain the numerical solution is as follows: we consider a pseudo-spherical change of variables and dimensionless process for the Boltzmann equation and we simulate the dimensionless equation (5). Thus, we obtain the dimensional electron density (see Appendix, Equation (10)). Then, we solve numerically the Poisson equation and obtain the electric field, which will be used in the transport equation (see expression of flux terms in Appendix).

Numerically, we simulate the Equation (5) using a conservative fifth order finite difference WENO scheme [7] for the transport variables. This is the most demanding computation in the numerical algorithm and will be explained in Section 4 (see also Appendix for details about WENO-scheme). In order to guarantee the precision, the 5D computational domain (directions x, y, ω, μ, ϕ) is discretized by taking an uniform mesh in each direction. We will focus on the simulation of charge transport in a rectangular area of a 2D semiconductor device with dimensions $L_x \times L_y$ and we will use a 5D mesh with dimensions $(N_x+1)\times(N_y+1)\times N_\omega \times N_\mu \times N_\phi$ in the numerical resolutions. The coordinates of the mesh points are calculated as follows:

$$\begin{array}{lll} x_i = i\Delta x, & i = 0, 1, \ldots, N_x, & \Delta x = L_x/N_x \\ y_j = j\Delta y, & j = 0, 1, \ldots, N_y, & \Delta y = L_y/N_y \\ \omega_k = (k-1/2)\Delta\omega, & k = 1, \ldots, N_\omega, & \Delta\omega = \omega_{max}/N_\omega \\ \mu_m = (m-1/2)\Delta\mu - 1, & m = 1, \ldots, N_\mu, & \Delta\phi = \pi/N_\phi \\ \phi_n = (n-1/2)\Delta\phi, & n = 1, \ldots, N_\phi, & \Delta\mu = 2/N_\mu \end{array} \tag{6}$$

where w_{max} denotes the maximum value of the energy, which is adjusted in the numerical experiments to fulfill that $\Phi(t, x, y, \omega, \mu, \phi) \simeq 0$ for every t, x, y, μ, ϕ and $\omega > \omega_{max}$. Moreover, we consider $\omega_{max} = N\alpha$ (an integer multiple of α, see [4]).

The midpoint quadrature formula is used to evaluate the dimensionless collision operator $C(\Phi)(t, x_i, y_j, \omega_k, \mu_m, \phi_n)$ (see Appendix, Equation (8)) at each point of the 5D domain, and the electron density $\rho(t, x_i, y_j)$ (see Appendix, Equation (10)). As a consequence, we obtain a charge-conservative method.

To solve the Poisson Equation (4) and compute the electric potential V, we have to solve a linear system whose coefficient matrix is a banded matrix

which does not change through time integration. Therefore, we can use a direct method to solve the linear system where the factorization of the matrix is made only once and one forward elimination and back substitution is made for each Runge-Kutta stage. When the electric potential V has been computed, we can approximate the electric field at each 2D mesh point by using central finite differences.

The computation of the potential is performed in the whole device, but Equation (5) is only simulated in the transport zone. This fact constitutes an important difference with [10]. Moreover, in the present work a different solver has been considered to simulate the Poisson Equation (see Section 4). The evolution in time of the distribution function Φ is developed by mean of a third order TVD Runge-Kutta scheme (see Appendix for a detailed description).

3 Device model

The Boltzmann-Poisson system depends on the characteristics of the semiconductor device we wish to simulate, due to the influence of the geometry and the scatterings in the materials. For instance, in MOSFET devices (with one or two gates) there are regions without electron transport. These regions are oxide bands (see Figure 1 for a MOSFET device with a single gate and for a DG-MOSFET device). Consequently, the BTE Equation (5) is only taken into account in the silicon band (transport region), while the Poisson Equation (4) has to be solved in the whole device. This means an important difference with the analysis developed in [10] for a MESFET device, where only transport region had to be considered.

In order to avoid redundant computations, two different regions in the 2D device are considered:

- *Non transport region*: It is an area without charge transport, where only the electric potential and the electric field are computed.
- *Transport region*: It is an area with electron transport. Since the function Φ is defined only in this region, the computational cost of the algorithm is concentrated in this region and a good balance of the computational workload associated to this area is essential to design a parallel version.

The whole physical 2D device (including the non transport region) is discretized using $Nxall + 1$ mesh points in the x direction and $Nyall + 1$ mesh points in the y direction. In the transport region, the distance between mesh points in x and y is considered uniform due to the use of the WENO scheme. Although this can be considered as a limitation of the numerical scheme, the scheme makes it possible to define different regions in the physical space of the device, each one using different uniform spatial step sizes (in x and y directions), by using interpolation between regions (see [15]). To discretize the physical space in the transport region, we consider $N_x + 1$ equidistant points in x and $N_y + 1$ equidistant points in y (see Figure 2). For the transport region, we have considered an uniform mesh in each direction of the $5D$ space $(x, y, \omega, \mu, \phi)$ with mesh sizes $\Delta x, \Delta y, \Delta\omega, \Delta\mu$ and $\Delta\phi$ (see (6)).

With this discretization, we have considered the following data structures:

- *Structures defined in the whole device*: we have used a 2D array, $\mathbf{V}$, with $(Nxall + 1)(Nyall + 1)$ cells to store the values of the electric potential V. Therefore: $\mathbf{V}(i, j) = V(t, x_i, y_j)$, $i = 0, \ldots, Nxall$, $j = 0, \ldots, Nyall$.

  Additionally, two arrays with this structure have been used to store the

values for the components of the electric field, E_x and E_y, in order to save them into a data file.

- *Structures defined only in the transport region*:
  - To store the values of the adimensional distribution function Φ at each point of the 5D domain and each time instant t_n, we have used a 5D array $\mathbf{\Phi}$ with $(N_x+1)\times(N_y+1)\times N_\omega \times N_\mu \times N_\phi$ cells. Therefore, in the s-th Runge-Kutta stage ($s=0,1,2$) of the n-th time step of simulation, $\mathbf{\Phi}$ fullfills that $\mathbf{\Phi}(i,j,k,m,n) = \Phi^{(n,s)}(t^n, x_i, y_j, \omega_k, \mu_m, \phi_n)$, where i,j,k,m,n vary between the limits given in (6).

    We use an additional array with the same structure of $\mathbf{\Phi}$ to model $L(\Phi^{(n,s)})$ at each time step as defined in (11), where L is an approximation of the spatial-momentum derivatives and the collision operator (see Appendix). This array will be denoted by $\boldsymbol{L}(\Phi)$.
  - The electron density ρ and the electric field components (which only need to be computed in the transport region) are modeled with 2D arrays, $\boldsymbol{\rho}$, E_x and E_y, with $(N_x+1)\times(N_y+1)$ cells, as follows:

$$\begin{aligned} \rho(t,x_i,y_j) &= \boldsymbol{\rho}(i,j) \\ \mathbf{E}(t,x_i,y_j) &= (E_x(i,j), E_y(i,j)), \qquad i=0,\ldots,N_x, \quad j=0,\ldots,N_y. \end{aligned}$$

    Additionally, other arrays with the same structure and type are used to store several intermediate results (for instance, to store the right hand side of the Poisson Equation (4)).

3.1 Device description

The user of this application describes the information about the 2D device to be simulated in a file, using a previously fixed notation (see Figure 3 for a

MOSFET device and Figure 4 for a DG-MOSFET device). In this way, the sofware can be applied to simulate devices with any rectangular geometry (or more general geometries, which can be divided in rectangular regions with and without charge transport). A description file is structured in sections (`GEOMETRY`, `POTENTIAL`, `SIMULATION`, etc.) and has to include the following information:

- **2D device geometry**: the section `GEOMETRY` allows the user to define the sizes of the whole device, the position and size of the transport region (where the exclusion of oxide areas can be described with the clause `EXCLUDING` as it is shown in the MOSFET description of Figure 3) and the contact zones (source, drain and gates). Additionally, in this section, we must include the information concerning to the number of mesh points ($Nxall$, $Nyall$, N_x, N_y, N_ω, N_μ, N_ϕ).
- **Initial conditions**: there are specific sections to assign the initial values at each mesh point of the physical space for variables associated to the doping concentration (`DONORS`, `ACCEPTORS`), dielectric constant (`DIELECTRIC`) and the electric potential applied to the contacts (`POTENTIAL`). These values can be defined by assigning values to rectangular regions of the 2D spatial device or providing the name of a file containing the values assigned to each spatial mesh point.
- **Simulation parameters**: the user provides several parameters which result of interest for the simulation such as the final simulation time and the maximum number of time steps which are permitted.

4 Parallel WENO BTE solver

4.1 Decomposition strategy

In this numerical algorithm, each component of the array $\mathbf{\Phi}$ (which is the most important data structure) can be updated simultaneously by performing the same type of calculations at each time step. Therefore, the solution methods can be parallelized in a natural way using domain decomposition techniques [8]. The data arrays described in Section 3 have been suitably partitioned among the processors of the parallel machine and each processor will update its local data subdomain simultaneously. To minimize the total execution time, the mapping of data to processors must balance the computational workload while minimizing the volume and frequency of interprocessor communication. We have considered that a good strategy consists of partitioning the 2D physical space of the device among the processors (geometrical decomposition) by using a 2D block distribution scheme [8]. This type of data distribution is applied to 5D arrays (like $\mathbf{\Phi}$) by splitting the x and y dimensions.

A bidimensional grid of logical processors is defined automatically from the mesh size parameters for the transport area (N_x and N_y) and the number of existing physical processors, although the user can define explicitly the dimensions of the logical processor grid. The data arrays are assigned to the processors of this grid by partitioning the dimensions of the 2D physical space by rectangular blocks. However, the load balancing of this block decomposition is guided by the dimensions of the transport area.

Given a 2D logical grid with $P_x \times P_y$ processors, the array $\mathbf{\Phi}$ is assigned to

this grid by using rectangular blocks (if we only consider the partitioning of the 2D physical space) with $b_x \times b_y \times N_\omega \times N_\mu \times N_\phi$ contiguous elements, with $b_x = \lceil (N_x + 1)/P_x \rceil$ and $b_y = \lceil (N_y + 1)/P_y \rceil$, where for all $a \in \mathbb{R}$, $\lceil a \rceil$ denotes the smallest integer not smaller than a. The same type of data distribution, but considering 2D blocks with $b_x \times b_y$ elements, is followed for the 2D arrays defined on the transport area ($\boldsymbol{\rho}$, E_x, E_y, etc.).

Since most of the computational cost of the numerical scheme is concentrated on the transport area, the 2D array $\mathbf{V}$, which maintains values for points out of the transport area, will be replicated in all the processors. Moreover, the computation of this data array is not decomposed among the processors but all the components of $\mathbf{V}$ are computed on all the processors. Figure 5 illustrates the mapping of a DG-MOSFET device which includes a transport region, discretized with 27×7 points ($[N_x + 1] \times [N_y + 1]$), on a logical grid with 3×2 processors. In this figure, we can see how each processor manages a rectangular block with a similar number of points in the transport area and all the processors must manage the array $\mathbf{V}$ which includes points that do not belong to the transport area. This strategy exhibits the following advantages:

- The computational load is well balanced among the processors since simulation of the transport area consumes almost all the execution time and each point in the transport area requires approximately the same computation.
- We will show that the communication costs are relatively low if compared with the computation because the most significant communication operations are derived from the computation of the derivatives with respect to the physical space dimensions (x and y).
- Since many computation phases do not require remote communication, a great deal of the existing sequential code can be reused without changes to

derive the parallel version. As a result, the software engineering costs have been considerably reduced.

4.2 Parallel algorithm

A sketch of the parallel algorithm which simulates the evolution of Φ is shown in Figure 6, where the main computing phases are represented by labelled boxes. In this figure, several boxes have a shadow to indicate that the corresponding calculations are distributed among the processors of the parallel machine. The boxes without shadow indicate that the corresponding calculations are replicated in all the processors of the parallel machine.

Initially, the information of the device is read from a description file (see example in Figures 3 and 4) and these data (which include the values of the doping concentration and the dielectric constant at each spatial mesh point together with the potential in the contacts) are copied to all the processors.

The Poisson Equation (4) involves the solution of a linear system with the following form:

$$A \cdot V = \frac{\mathfrak{e}}{\epsilon_0} \left[\rho(i,j) - N_D(i,j)\right] \tag{7}$$

where A can be obtained by discretizing the spatial derivatives of Equation (4) by using a combination of finite differences schemes [11] and introducing the appropriate boundary conditions (see [5]). As a result, A is a $n \times n$ matrix with $n = (Nxall + 1)(Nyall + 1)$. Since A is a constant banded matrix (it does not change through time integration) with bandwidth $2Nxall + 3$ and n is not too large, we build and obtain the LU decomposition of A before

the main loop of the application. This LU decomposition will be used to perform one forward elimination and back substitution for each time step and Runge-Kutta stage (see Figure 7) in order to solve the linear system (7). The construction and factorization of A is performed in all processors only once at the start of the computation (see Figure 6). A different scheme was considered in [10], where an iterative parallel red-black SOR (Successive Over Relaxation) method was used. A direct solver becomes more appropriate in this case, since LU decomposition is computed only once at the beginning of the process. Therefore, it is not worth considering a parallel scheme to solve the systems after the LU decomposition, since it is not expensive at all.

Next, the local block of the array $\mathbf{\Phi}$ which each processor maintains is initialized. The local block of the processor $P_{l,m}$ in a $P_x \times P_y$ processor grid includes the elements of the form $\mathbf{\Phi}(i,j,k,m,n)$ with $i = istart, \ldots, iend$, $j = jstart, \ldots, jend$, $k = 1, \ldots, N_\omega$, $m = 1, \ldots, N_\mu$, $n = 1, \ldots, N_\phi$, where:

$$\begin{array}{ll} istart = l * b_x, & iend = \min(N_x + 1, (l+1) * b_x - 1), \\ jstart = m * b_y & jend = \min(N_y + 1, (m+1) * b_y - 1), \end{array}$$

and the block sizes b_x, b_y has been defined in the previous section. The local blocks of the 2D distributed arrays (like $\boldsymbol{\rho}$) has the same structure as the local blocks of $\mathbf{\Phi}$ but omits the indexes k, m and n.

When $\mathbf{\Phi}$ is initialized, the following computational phases have to be performed for each time step and Runge-Kutta stage:

(1) The local block of the electron density array, $\boldsymbol{\rho}$, is computed from $\mathbf{\Phi}$. This phase requires the parallel evaluation of the composite midpoint quadrature formula to approximate the integral (see Appendix, Equation (10)). The mapping of elements of $\boldsymbol{\rho}$ and $\mathbf{\Phi}$ onto the processors makes

it possible that all the data necessary to compute the local block of $\boldsymbol{\rho}$ is available in the subdomain of each processor.

(2) The Poisson solver must be used to obtain the local blocks of the arrays E_x and E_y (see Figure 7). Taking into account that the solver is applied to relatively small 2D domains ($N_x < 300$ and $N_y < 300$), we have finally decided to replicate the computation of the electric potential $\mathbf{V}$ onto all the processors. The right hand side of Equation (4) is computed in parallel, following the block distribution of the array $\boldsymbol{\rho}$, but then it is replicated onto all processors. When a copy of the array V has been computed in all the processors, each processor computes its main block of the arrays E_x and E_y. The replication of the code to solve the linear system (7) has a beneficial effect on the performance of the parallel solver. In fact, the small dimension of the system and the type of distribution followed by the right hand side vector make the overhead of a distributed solution for (4) greater than the parallelization benefits. This fact represents an improvement with respect to the code developed in [10]. We have checked experimentally that the time needed to compute the potential is reduced considerably with this choice. Moreover, the execution time required by the Poisson solver is negligibe in comparison with the time required by the rest of calculations (see Figure 16).

(3) The local block of the array $\boldsymbol{L}(\Phi)$, which stores the addition of the derivatives obtained by WENO5 and the evaluation of the collision term (see Appendix, Equation (12)), is computed at each processor (see the following subsection).

(4) The local block of $\boldsymbol{\Phi}$ is updated from the block of $\boldsymbol{L}(\Phi)$ during the s-th Runge-Kutta stage by using a data parallel version of the method introduced in Equation (11) of Appendix. With the distribution followed for

the distributed arrays $\mathbf{\Phi}$ and $\boldsymbol{L}(\Phi)$, each processor can update its main subdomain of $\mathbf{\Phi}$ during this phase without interprocessor communication.

(5) After a time step, the new step size Δt is computed and the current time t is updated.

(6) When the new current time t fullfills a certain condition defined by the user (according to a given update frequency), the distributed array $\mathbf{\Phi}$ is gathered to one particular processor in order to be processed by a visualization system. The visualization system allows the user to view graphical representations of several macroscopic quantities (electron density, electric potential, energy, x and y components of the electric field, x and y components of the moment, etc.) on the screen (see Figure 10) at the current simulation time, and must also periodically generate several output files containing data about the probability density function f (obtained from the $\mathbf{\Phi}$ array) at intermediate time steps.

4.2.1 Computation of the spatial derivatives and the collision term

In Figure 8, the different phases which are necessary to update the distributed array $\boldsymbol{L}(\Phi)$ are illustrated.

The local block of the distributed array $\boldsymbol{L}(\Phi)$ is initialized with the approximation of the collision operator (see Appendix, Equation (8)) by using the composite midpoint quadrature formula. Taking into account the distribution of $\mathbf{\Phi}$ among the processors, this calculation can also be performed in parallel without interprocessor communication.

To parallelize the computation of the flux terms in each dimension by using the WENO5 scheme, it is necessary to consider some dependences since the

derivatives of Φ at each mesh point are calculated following an asymmetric pattern with 6 points (2 points on the left and 3 on the right or vice versa), which depends on the sign of the corresponding coefficient (see Appendix for more details). These dependences are less important in directions of ω, μ y ϕ, since the distribution of the array $\mathbf{\Phi}$ among the processors guarantees that the computation of these derivatives can be completed in each processor without interprocessor communication. However, to compute the derivatives with respect to x and y, each processor must exchange the data points located in the boundary of its $\mathbf{\Phi}$ subdomain with its neighbours in the processor mesh.

In order to avoid the dependence on the coefficient sign and to reduce the data reordering cost, we have assumed a symmetric pattern with 7 points to compute the derivatives in x and y in the parallel version. Precisely, we assume that the Φ derivative with respect to x direction at a point value with position (i, j, k, m, n) in array $\mathbf{\Phi}$ depends on values of points with positions in the set $\{(l, j, k, m, n) : l = i-3, i-2, i-1, i, i+1, i+2, i+3\}$. We reason in an analogous way for the derivative in the y direction by changing i by j. Therefore in these derivatives, each processor must transfer 3 rows or 3 columns in its subdomain boundaries of the array $\mathbf{\Phi}$ with each neighbour in the processor mesh (see Figure 9). Moreover, the local block of the $\mathbf{\Phi}$ array at each processor must include some additional ghost cells in their boundaries in order to store and access easily to the values received from neighbour processors.

The time taken to perform the remote communication in the parallel computation of $\boldsymbol{L}(\Phi)$ is reduced by using nonblocking communication primitives [8] in order to overlap the computation of the derivatives in directions ω, μ and ϕ (and the collision operator) with the communication necessary to exchange the subdomain boundaries in the x and y directions. To implement this stra-

tegy, the transference must be initiated before starting the computation of $\boldsymbol{L}(\Phi)$ and it must be completed before the computation of the derivatives in x and y directions as it is illustrated in Figure 8.

4.3 Implementation issues

The solver has been implemented following a SPMD (Single Program Multiple Data) programming style [8] as a C++ SPMD program augmented with calls to MPI-1 functions [18]. In order to obtain the best performance, we have used the C++ gnu compiler with the options which AMD recommends to optimize numerical codes onto AMD Opteron processsors (-O3 -m64 -march=opteron -ffast-math -funroll-all-loops -ftree-vectorize).

Several C++ classes have been defined to manage 2D and 5D arrays distributed following the above mentioned 2D block mapping. The class which encapsulates 5D arrays (which models the function Φ) includes methods to perform the following: to compute the derivatives in each direction, to start and finalize the exchange of the boundaries with neighbour processors (by using the ghost cells), to update the array by performing a Runge-Kutta stage, to evaluate the collision term, to compute de electron density array, etc. The encapsulation of the distributed data structures facilitates the maintenance of the existing parallel code and makes it possible to reuse this code in applications with similar requirements.

MPI nonblocking communication functions [18] have been used to enable the overlapping of computation and communication in the parallel evaluation of $\boldsymbol{L}(\Phi)$. These functions involve the use of opaque request objects which identify

the particular communication operation in order to be able to check its completion. Since the data exchange between processors is repeatedly performed with the same arguments for each Runge-Kutta stage, we have used MPI persistent communication requests to implement efficiently this data exchange and reduce the overhead for remote communication in each Runge-Kutta stage.

To solve the Poisson equation (4), we used routines of the LAPACK library for banded linear systems [1] which have been optimized for AMD Opteron processors.

The visualization system consists in a graphical user interface where the values of several macroscopic quantities at each 2D spatial grid point of the device can be drawn on the screen by using 3D graphics, in order to study interactively its evolution in time. The user can interact with the visualization system by: choosing the particular magnitudes which are to be drawn on the screen, stablishing what is the update frequency for the graphics and for the output data files, performing geometric transformations on the 3D graphics (rotation, translation and scaling) and modifying the parameters of the 3D graphical representation. A full screenshot which shows the 3D graphical representations of the electron density, the potential and the energy are presented in Figure 10. To implement the visualization system, it has been necessary to integrate several libraries to develop graphical interfaces and 3D graphics with a Posix Thread library and MPI. In particular, one particular MPI process must manage two different Posix threads: the main thread is part of the MPI parallel simulation and the other thread executes the visualization system. Both threads are synchronized by using semaphores to share simulation data but their interaction does not affect to the consistency of the simulation.

5 Experimental results

The solver has been applied to two types of MOSFET devices: a MOSFET with a single gate and a DG-MOSFET, whose geometries and characteristics are described in Figure 3 and Figure 4, respectively. Several numerical experiments have been done on a heterogeneous cluster of 7 SMP machines with 4 GBs. of RAM (per machine) running Open SuSE 10.3 Linux, connected with a Gigabit Ethernet switch. 4 of these machines are biprocessor AMD Opteron (2.4 GHz) and 3 are biprocessor AMD Opteron dual core (2.0 GHz). The cluster has 20 processing elements and several experimental tests have shown that the difference of performance between their processing elements is very small. Therefore, we can consider this cluster as an homogeneous parallel system with 20 processors, taking into account the synchronous nature of this application.

One particular simulation of the DG-MOSFET device has been performed by considering a mesh with $N_x = 30$, $N_y = 48$, $N_\omega = 120$, $N_\mu = 12$ and $N_\phi = 12$ for the domain $(x, y, \omega, \mu, \phi)$ in the transport region. The parameters to define the mesh for the whole 2D device are $Nxall = 30, Nyall = 56$. Figure 10 shows how the electric potential, density and energy vary on the 2D physical device when $t = 0.164036$ ps.

We have performed experiments to study the runtime performance of the solver on the parallel machine when devices with different geometry are simulated.

Figures 11 and 13 represents the evolution of the execution time when the number of processors is increased in the parallel simulation of the MOSFET

and DG-MOSFET device respectively. In these figures, 4 different meshes are used for the transport region and the execution times corresponds with the simulation of the integration time interval [0 ps, 0.01 ps]. Figures 12 and 14 show the speedups obtained for different mesh sizes and number of processors for the same integration time interval. The speedup results have been obtained by comparing the execution times of the parallel solver with the runtimes obtained with a highly optimized monoprocessor code based on the same numerical scheme. . The monoprocessor application have been executed on one of the multicore AMD processors of the target machine but only exploit one core of the physical node. In fact, the runtime results for one processor (see Figures 11 and 13) are obtained with this sequential solver.

The results show a very good scalability and a high parallel efficiency for meshes of practical interest when both devices considered are simulated.

For the particular MOSFET geometry considered (see Figure 3), the non transport area (SiO_2 area) occupies approximately the 11% of the full 2D device area. The computation related to this area is not parallelized and it is an overhead source in the parallel simulation. Since the proportion of oxide for this MOSFET device is small, the influence on the performance is small and an efficiency higher than 80% is achieved for medium size meshes and higher than 90% for transport meshes with a high number of points. Figures 11 and 12 show a very good scalability of the parallel solver in the simulation of this device for the range of number of processors we have considered.

For the particular DG-MOSFET geometry (see Figure 4), the non transport area occupies approximately the 14% of the full 2D device area and consequently the negative influence on the performance is something more no-

ticeable. However, an efficiency higher than 75% is achieved for medium size meshes and higher than 80% for transport meshes with a high number of points. Figures 13 and 14 also show a very good scalability in this situation for the range of number of processors we have considered.

The beneficial effect of the communication-computation overlapping on the performance is only obtained when the number of processors is sufficiently large. In particular, with number of processors less than 16, the benefits of the overlapping are not evident. However, with $P = 16$ and $P = 20$ the time reduction in the evaluation of $\boldsymbol{L}(\Phi)$ is considerable, as shown in Figure 15. In this figure, the execution times taken to compute $\boldsymbol{L}(\Phi)$ at one integration step of the DG-MOSFET device simulation with different meshes and communication strategy are compared for $P = 16$ and $P = 20$.

The relative impact on the execution time of the most important computing phases are shown in Figure 16 when the DG-MOSFET is simulated using several transport mesh sizes and $P = 20$ processors. It is important to note that the contribution of the Poisson solver is very small and consequently the replication of this computing phase does not degrade the scalability of the parallel solver.

6 Conclusions and Further work

A flexible and efficient parallel solver of the Boltzmann-Poisson system for semiconductor devices has been presented. This system models the behavior of realistic 2D semiconductor devices. A suitable geometrical decomposition of the data domain among the processors, the replication of the Poisson solution

by means of a direct linear system solver and an efficient design of the remote communications have been considered to reduce the overheads. As a consequence, the parallel solver exhibits a good balance of the computational load, and the communication costs for the most important simulation phases are relatively small in comparison with the computation costs. The experimental results obtained by simulating devices with different geometries on a SMP cluster show a good speedup and scalability in the range of processor numbers which has been considered. These results also suggest that the efficiency can be limited if the non transport area occupies a relatively high proportion in the full 2D spatial device. This solver allows us to simulate a huge range of realistic devices in a simple way without need of knowing details about the numerical scheme. This is possible because the solver is designed to deal with a great variety of shapes, materials, boundary conditions ... Therefore, it can be used by electronic engineers who could reach the results by only describing the geometry of the device with an input file.

Although the solver is not competitive with Monte Carlo schemes in run-time performance (because of the deterministic simulation of full Boltzmann-Poisson system is very costly), the response times of our parallel solver are quite acceptable and the solver overcomes several drawbacks of Monte Carlo schemes: noise free resolution, it enables evolution on time of the distribution function and therefore of all the moments, it makes it possible to simulate almost empty regions, etc. On the other hand, the Boltzmann-Poisson system represents the most realistic model for this kind of devices where a semiclassical description is enough. Therefore a deterministic simulation of this system is interesting since could be considered as a reference to tune other simpler models (drift-diffusion, hydrodynamic, ...).

As a continuation of this paper, we are working on extending the parallelization strategy to enable efficient simulations on parallel heterogenous systems and developing a version of the solver for Graphics Processing Unit (GPUs) [16], in order to improve the performance drastically.

Appendix

In this Appendix we collect the missing details in the paper. Although they were developed in previous publications (included in the References), we find it convenient to write them here in order to facilitate the reading to the interested reader.

Dimensionless Boltzman Equation

In the 2D physical space a dimensionless Boltzmann equation, using the change of variables and the dimensionless process given in [4,5], is written in conservative form as:

$$\frac{\partial \Phi}{\partial t} + \frac{\partial}{\partial x}(a_1\Phi) + \frac{\partial}{\partial y}(a_2\Phi) + \frac{\partial}{\partial \omega}(a_3\Phi) + \frac{\partial}{\partial \mu}(a_4\Phi) + \frac{\partial}{\partial \phi}(a_5\Phi) = s(\omega)C(\Phi),$$

where $C(\Phi)$ represents the dimensionless collision operator, in the new variables, which is given by:

$$C\,(\Phi)(t,x,y,\omega,\mu,\phi) = \frac{1}{2\pi\, t_*}\int_0^{\pi}\int_{-1}^{1}[\beta\Phi(t,x,y,\omega,\mu',\phi') + a\Phi(t,x,y,\omega+\alpha,\mu',\phi') + \Phi(t,x,y,\omega-\alpha,\mu',\phi')]d\phi' d\mu'$$

$$- \frac{1}{s(\omega)\, t_*}[\beta s(\omega) + a s(\omega - \alpha) + s(\omega + \alpha)]\Phi(t, x, y, \omega, \mu, \phi)\,. \tag{8}$$

where the constant parameters t_*, α, a and β depend on the material (see [4] and [3] for details).

The coefficients for the flux terms $(a_1, \ldots, a_5)$ are given by:

$$\begin{aligned}
a_1(\omega) &= \frac{1}{t_*}\frac{\mu s(\omega)}{p(\omega)^2}, \qquad\qquad a_2(\omega, \mu, \phi) = \frac{1}{t_*}\frac{q(\mu) s(\omega)\cos\phi}{p(\omega)^2} \\
a_3(t, x, y, \omega, \mu, \phi) &= -\frac{1}{t_*}\frac{2 s(\omega)}{p(\omega)^2}\left[E_x(t, x, y)\mu + E_y(t, x, y) q(\mu)\cos\phi\right] \\
a_4(t, x, y, \omega, \mu, \phi) &= -\frac{1}{t_*}\frac{p(\omega)}{s(\omega)} q(\mu)\left[E_x(t, x, y) q(\mu) - E_y(t, x, y)\mu\cos\phi\right] \\
a_5(t, x, y, \omega, \mu, \phi) &= \frac{E_y(t, x, y)}{t_*}\frac{\sin\phi}{q(\mu)}\frac{p(\omega)}{s(\omega)}\,,
\end{aligned}$$

where $q(\mu) = \sqrt{1 - \mu^2}$ and $p(\omega) = (1 + 2\alpha_\kappa \omega)$. E_x and E_y are the components of the electric field vector ($\mathbf{E} = [E_x, E_y] = [\frac{\partial V}{\partial x}, \frac{\partial V}{\partial y}]$), which is computed self-consistently by solving the Poisson equation (4) in the 2D spatial domain:

$$\frac{\partial\left[\epsilon_r(x, y)\frac{\partial V}{\partial x}\right]}{\partial x} + \frac{\partial\left[\epsilon_r(x, y)\frac{\partial V}{\partial y}\right]}{\partial y} = \frac{\mathfrak{e}}{\epsilon_0}\left[\rho(t, x, y) - N_D(x, y)\right]. \tag{9}$$

where $\rho(t, x, y)$ is obtained from Φ by:

$$\rho(t, x, y) = C_D \int_0^{\pi}\int_0^{\infty}\int_{-1}^{1} \Phi(t, x, y, \omega, \mu, \phi)\, d\mu\, d\omega\, d\phi, \tag{10}$$

where C_D is a dimensional constant (see [4] and [3] for details).

Fifth order finite difference WENO scheme

Finite differences WENO schemes are useful to approximate the derivatives of functions with singularities. Due to the usually discontinuous doping profile,

and consequently with high gradient regions, it becomes appropriate to use a fifth order finite difference WENO scheme to approximate the derivatives on space and momentum. This method is used for non linear conservation laws and has been considered in a huge number of different applications (see [7,12,13]). The WENO scheme is fifth order for smooth solutions and is non oscillatory in the high gradient regions. Due to this fact we can use a third order TVD (*Total Variation Diminishing*) Runge-Kutta explicit method [14] for the evolution on time.

We obtain the approximations to the point values of the solution by means of dimension by dimension approximation to the spatial derivatives by using the above-mentioned fifth order WENO scheme. To be more precise, the approximation of $\frac{\partial}{\partial x}(a_1\Phi)$ at the mesh point $(x_i, y_j, \omega_k, \mu_m, \phi_n)$, is obtained with the following procedure:

- The variables $y = y_j$, $\omega = \omega_k$, $\mu = \mu_m$ and $\phi = \phi_n$ are fixed.
- We call $g_i = a_1\Phi_i$, $i = -3, -2, -1, \ldots, N_x+3$, where the values of $\Phi_{n,i,j,k,m,n}$ with $i < 0$ and $i > N_x$ are given by the boundary conditions considered in [3], $\Phi_i = \Phi(t^n, x_i, y_j, \omega_k, \mu_m, \phi_n)$ and $a_1 = a_1(\omega_k, \mu_m)$ where j, k, m, n are fixed.
- Finally, for $i = 0, \ldots, N_x$, we approximate $\frac{\partial}{\partial x}(a_1\Phi)(x_i, y_j, \omega_k, \mu_m, \phi_n)$ by:

$$\frac{\partial}{\partial x}a_1\Phi_i = \begin{cases} W(g_{i-3}, \ldots, g_{i+2}), \text{ if } a_1 > 0 \\ \\ W(g_{i+3}, \ldots, g_{i-2}), \text{ if } a_1 \leq 0 \end{cases}$$

  where $W : \mathbb{R}^6 \to \mathbb{R}$ is a non linear function which is described in [4].

Therefore, if $a_1(\omega_k, \mu_m) > 0$, the approximation in (i, j, k, m, n) depends on values of Φ in three left neighbour mesh points (in i) and two right neighbours,

while if $a_1(\omega_k, \mu_m) \leq 0$ the approximation depends on two neighbours on the right side and three on the left side in i.

Time discretization

A third order TVD Runge-Kutta method has been used for the evolution on time of the distribution function Φ. For a given value of the function on time t^n, Φ^n, we obtain an approximation of the distribution function on time $t^{n+1} = t^n + \Delta t$, Φ^{n+1}, after performing three steps:

$$\begin{aligned} \Phi^{(n,0)} &= \Phi^n + \Delta t L(\Phi^n); \quad \Phi^{(n,1)} = \frac{3}{4}\Phi^n + \frac{1}{4}\Phi^{(n,0)} + \frac{1}{4}\Delta t L(\Phi^{(n,0)}) \\ \Phi^{n+1} = \Phi^{(n,2)} &= \frac{1}{3}\Phi^n + \frac{2}{3}\Phi^{(n,1)} + \frac{2}{3}\Delta t L(\Phi^{(n,1)}) \end{aligned} \tag{11}$$

where L is an approximation of the spatial-momentum derivatives and the collision operator, which has the following form:

$$\begin{aligned} L(\Phi) = s(\omega)C(\Phi) - \Bigg(& \frac{\partial}{\partial \omega}(a_3\Phi) + \frac{\partial}{\partial \mu}(a_4\Phi) + \frac{\partial}{\partial \phi}(a_5\Phi) \\ & + \frac{\partial}{\partial x}(a_1\Phi) + \frac{\partial}{\partial y}(a_2\Phi) \Bigg) \end{aligned} \tag{12}$$

The time step size, Δt, is dynamically fixed using a CFL condition which depends on the evaluations of the coefficients of the flux terms $(a_1, \ldots, a_5)$ and the 5D mesh parameters (see [3]).

Acknowledgements

The authors are grateful to the authors of [5] for providing them the sequential code developed in their paper. It was very useful to check the experimental results of the parallel solver for a single gate MOSFET device. The authors also acknowledge partial support from DGI-MEC project MTM2008-06349-C03-03/MTM. J. M. Mantas also acknowledges partial support from DGI-MEC project TIN2004-07672-c03-02.

References

[1] Anderson E. , Bai Z., Bischof C., Demmel J., Dongarra J. J., Du Croz J., Grenbaum A., Hammarling S., McKenney A, Ostrouchov S., Sorensen D. LAPACK User's Guide. Society for Industrial and Applied Mathematics. 1999.

[2] Bova, S.W. and Carey, G.F. A distributed memory parallel element-by-element scheme for semiconductor device simulation. Comput. Methods Appl. Mech. Eng.. **181**,. 403–423 (2000).

[3] Cáceres, M. J., Carrillo,J. A., Gamba, I., Majorana, A., Shu, C.-W. Deterministic kinetic solvers for charged particle transport in semiconductors devices. in Cercignani, C., Gabetta, E. (eds.) Transport Phenomena and Kinetic Theory, Birkhuser, 151-171.

[4] Carrillo,J. A., Gamba, I., Majorana, A., Shu, C.-W. A WENO-solver for the transients of Boltzmann-Poisson system for semiconductor devices: performance and comparisons with Monte Carlo methods. J. of Comp. Physics, **184**, 498–525 (2003).

[5] Carrillo, J.A.; Gamba, I.M.; Majorana, A.; Shu, C.-W.: 2D semiconductor device

simulations by WENO-Boltzmann schemes: efficiency, boundary conditions and comparison to Monte Carlo methods, J. Comput. Phys., **214**, (2006), 55–80.

[6] Gazzaniga G., Lanucara P., Pietra P., Rovida S. and Sacchi G., Rapid parallelization of the drift diffusion model for semiconductor devices, In Proceedings of EWOMP 2002 (2002).

[7] Jiang, G. and Shu, C.-W., Efficient implementation of Weighted ENO schemes, J. Comp. Physics, **126**, 202–228 (1996).

[8] Grama, A., Gupta A., Karypis G., Kumar. V., Introduction to Parallel Computing, 2Ed. Addison-Wesley, 2003.

[9] Li Y., Sze S.M, Chao T.-S. A Practical Implementation of Parallel Dynamic Load Balancing for Adaptive Computing in VLSI Device Simulation, Engineering with Computers, Springer London, **18**, 124-137 (2002).

[10] Mantas, J.M.; Carrillo J.A.; Majorana, A.: Parallelization of WENO-Boltzmann schemes for kinetic descriptions of 2D semiconductor devices. Sci. Comp. in Electrical Eng. Mathematics in Industry Springer Series. **9**, 361-366 (2006).

[11] Selberherr, S., Analysis and Simulation of Semiconductor Devices. Springer Verlag, Wien, 1984.

[12] Shi, J., Zhang, Y.-T., Shu, C.-W., Resolution of high order WENO schemes for complicated flow structures, J. of Comp. Physics, **186**, 690–696 (2003).

[13] Shu, C.-W. Essentially non-oscillatory and Weighted Essentially Non-Oscillatory schemes for hyperbolic conservation laws. L. N. in Mathematics **1697** , 325-432 (1998)

[14] Gottlieb, Sigal., Shu, C.-W. Total variation diminishing Runge-Kutta schemes. Mathematics of Computation, **67** , 73-85 (1998)

[15] Sebastian, K., Shu, C.-W., Multidomain WENO finite difference method with interpolation at subdomain interfaces, J. Sci. Computing, **19**, 405-438 (2003).

[16] Rumpf M., Strzodka R., Graphics Processor Units: New Prospects for Parallel Computing, L. N. in Computational Science and Engineering, **51**, 89-121 (2006).

[17] Van de Velde, E. F. Concurrent Scientific Computing, Springer Verlag, (1994).

[18] Message Passing Interface Forum, MPI: A Message Passing Interface Standard, Univ. of Tennessee, Knoxville, Tennessee, (1995).

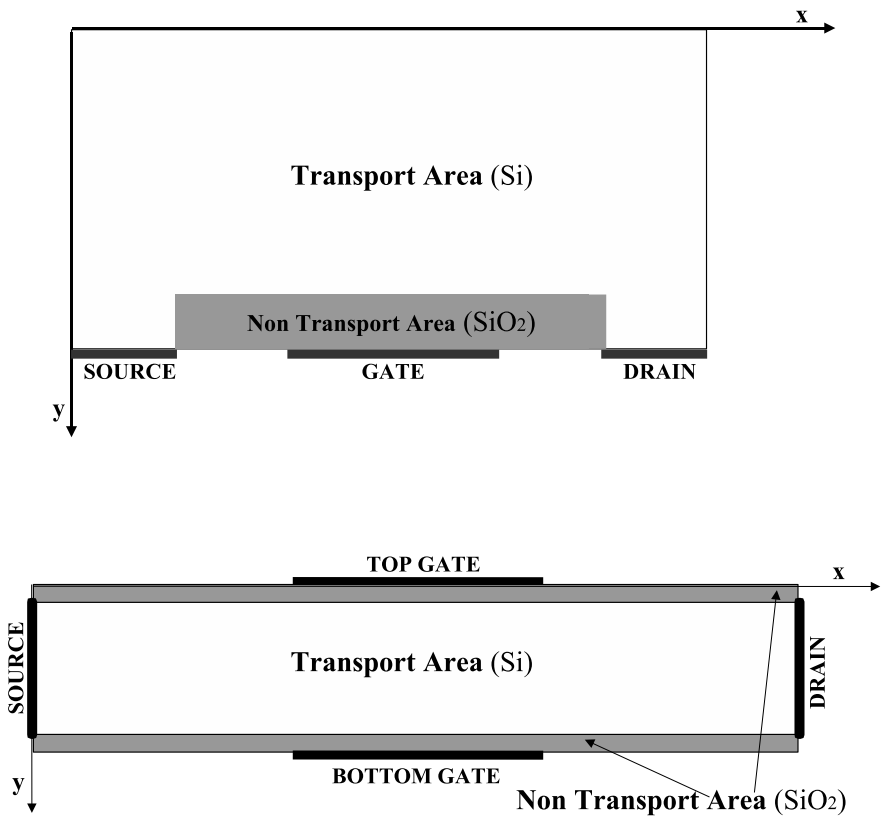


Fig. 1. Top: Silicon MOSFET scheme. Bottom: Silicon Double Gate MOSFET scheme.

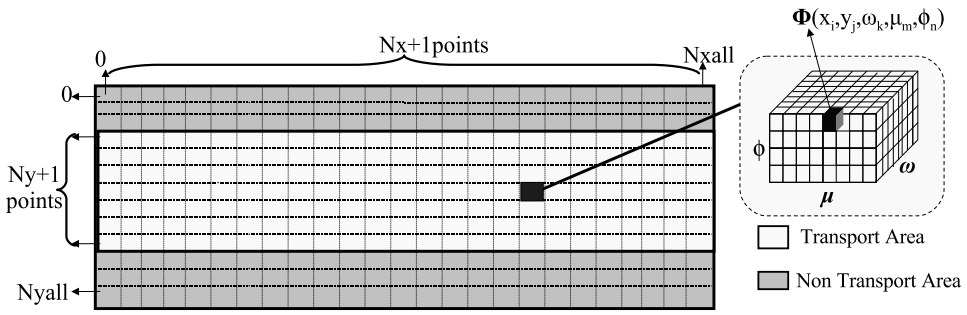
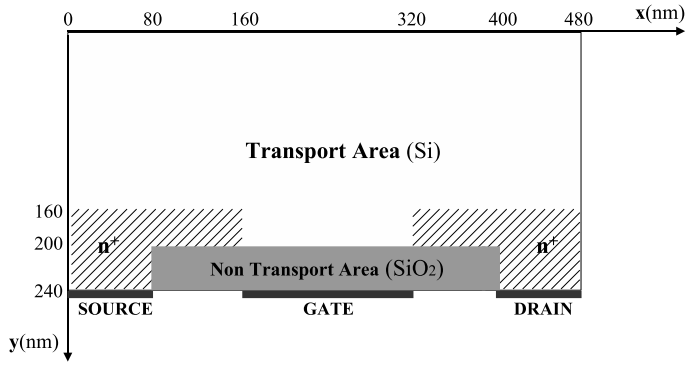


Fig. 2. Discretization for a 2D DG-MOSFET device.

Scheme of a particular MOSFET device



Description File for a particular MOSFET device

```
SET SCALE (centimeter,volt,sec);

GEOMETRY
 XLength = 480.0e-7; YLength = 240.0e-7;
 TRANSPORT[(0.0, 0.0)->(480.0e-7, 240.0e-7)]
      /* Non transport area embedded*/
      EXCLUDING [)80.0e-7,200.0e-7(->)400.0e-7,240.0e-7)];
 SOURCE [(0.0,240.0e-7)->(80.0e-7,240.0e-7)];
 DRAIN [(400.0e-7 ,240.0e-7)->(480.0e-7,240.0e-7)];
 GATE [(160.0e-7  ,240.0e-7)->(320.0e-7,240.0e-7)];
 Nxall=48; Nyall=36;
 Nx   =48; Ny   =36; Nw=66; Nmu=12; Nphi=12; NMULT   =6;
END

DONORS
/* n+ area*/
 REGION  [ (0.0, 160.0e-7)->(160.0e-7, 200.0e-7) ]          =3.0e17;
 REGION  [ (0.0, 200.0e-7)->(80.0e-7, 240.0e-7) ]           =3.0e17;
 REGION  [ (320.0e-7, 160.0e-7)->(480.0e-7, 200.0e-7) ] =3.0e17;
 REGION  [ (400.0e-7, 200.0e-7)->(480.0e-7, 240.0e-7) ] =3.0e17;
 /* n area*/
 REGION  [ (0.0, 0.0)          ->(480.0e-7, 160.0e-7( ]         =10.0;
 REGION  [ )160.0e-7, 160.0e-7)->)320.0e-7, 200.0e-7) ]  =10.0;
END
```

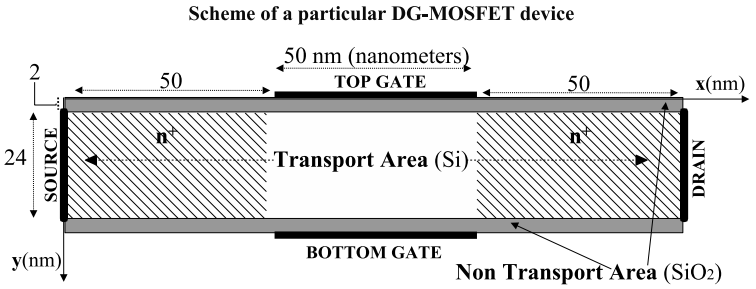
```
ACCEPTORS
 REGION [ (0.0, 0.0)     ->(480.0e-7, 200.0e-7)      ]       =0.0;
 REGION [ (0.0, 200.0e-7)->(80.0e-7, 240.0e-7)       ]    =0.0;
 REGION [ (400.0e-7, 200.0e-7)->(480.0e-7, 240.0e-7) ]=0.0;
END

POTENTIAL
  SOURCE=0.0;
  DRAIN=1.0;
  GATE=[0.4];
END

DIELECTRIC
   REGION [)80.0e-7, 200.0e-7(->)400.0e-7, 240.0e-7)]    =3.9;
   REGION [ (0.0, 0.0)     ->(480.0e-7, 200.0e-7)      ]       =11.7;
   REGION  [ (0.0      , 200.0e-7)->(80.0e-7 , 240.0e-7)]     =11.7;
   REGION  [ (400.0e-7, 200.0e-7)->(480.0e-7, 240.0e-7)]=11.7;
END

SIMULATION
  TEND=5.0;
  NTOT    =99999999;
  CFL     =0.89;
END
```

Fig. 3. Description file for a particular MOSFET device.



Description File for a particular DG-MOSFET device

```
SET SCALE (centimeter,volt,sec);

GEOMETRY
 /* Dimensions of the spatial device*/
 XLength = 150.0e-7;  YLength = 28.0e-7;
 /* Transport area */
 TRANSPORT[(0.0, 2.0e-7)->(150.0e-7, 26.0e-7)];
  /* Contacts*/
 SOURCE [(0.0     , 2.0e-7)->(0.0     , 26.0e-7)];
 DRAIN  [(150.0e-7, 2.0e-7)->(150.0e-7, 26.0e-7)];
 GATE   [(50.0e-7, 0.0)->(100.0e-7, 0.0)];
 GATE   [(50.0e-7, 28.0e-7)->(100.0e-7, 28.0e-7)];
 /* Grid parameters*/
 Nxall=30; Nyall=56;  Nx=30; Ny=48;
 Nw=120; Nmu=12; Nphi=12;
 NMULT   =2;
END

DONORS
 /* SOURCE */
 REGION  [ (0.0, 2.0e-7)->(50.0e-7, 26.0e-7) ]          =1.0e20;
 /* DRAIN */
REGION  [ (100.0e-7, 2.0e-7)->(150.0e-7, 26.0e-7) ]  =1.0e20;
 /* N CHANNEL*/
   REGION  [ )50.0e-7, 2.0e-7)->)100.0e-7, 26.0e-7) ]  =1.0;
END
```

```
ACCEPTORS
 /* SOURCE */
   REGION  [ (0.0, 2.0e-7)->(50.0e-7, 26.0e-7) ]         =1.0;
 /* DRAIN */
   REGION  [ (100.0e-7, 2.0e-7)->(150.0e-7, 26.0e-7) ]=1.0;
 /* N CHANNEL*/
   REGION  [ )50.0e-7, 2.0e-7)->)100.0e-7, 26.0e-7) ]  =1.0e12;

END
POTENTIAL
  SOURCE=0.0;
  DRAIN=0.5;
  GATE=[1.0,1.0];
END

DIELECTRIC
   REGION [ (0.0, 0.0)->(150.0e-7, 2.0) ]          =  3.9;
   REGION [ (0.0, 26.0e-7(->(150.0e-7, 28.0e-7) ]= 3.9;
   REGION  [ (0.0, 2.0e-7(->(150.0e-7, 26.0e-7) ] =11.9;
END

SIMULATION
  TEND=3.2;
  NTOT    =99999999;
  CFL     =0.9;
END
```

Fig. 4. Description file for a particular DG MOSFET device.

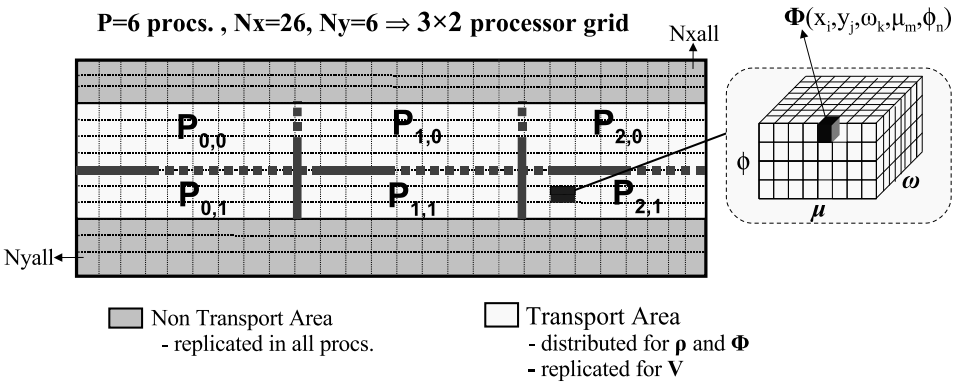


Fig. 5. Partitioning of the domain among the logical processors for a DG-MOSFET device.

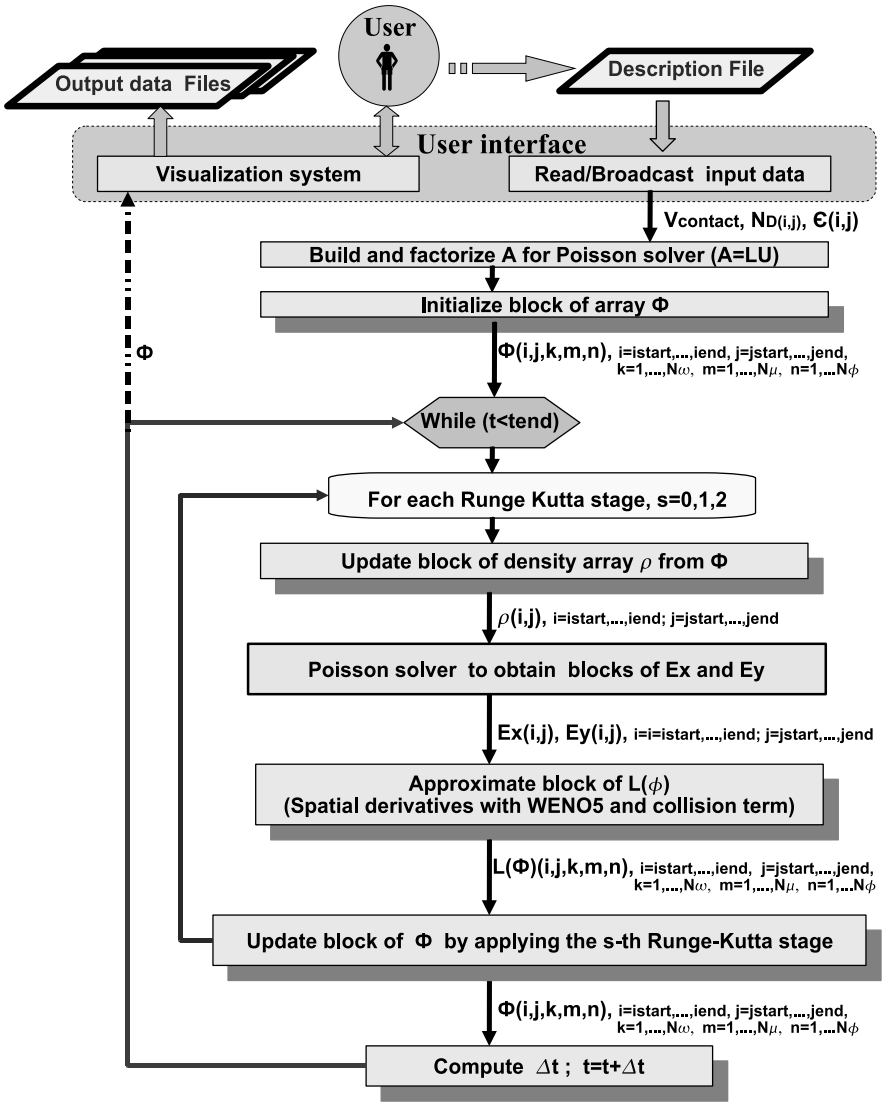


Fig. 6. General view of the algorithm.

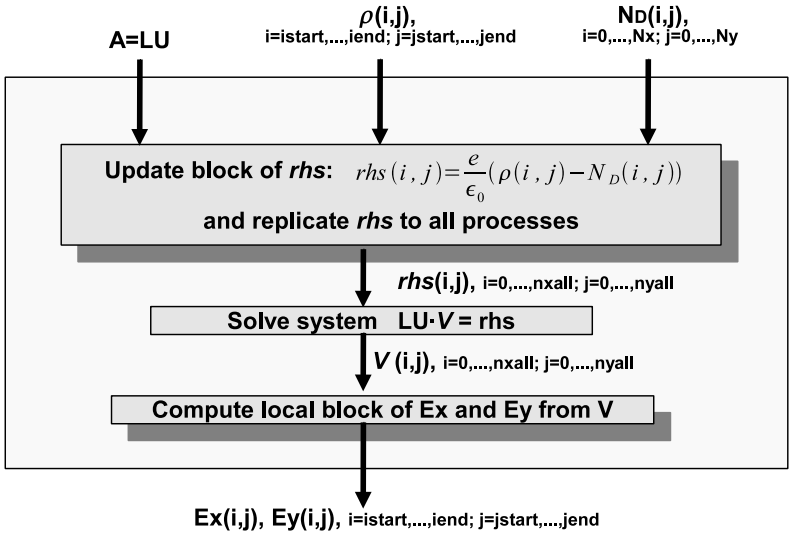


Fig. 7. Computation of the electric field components E_x and E_y.

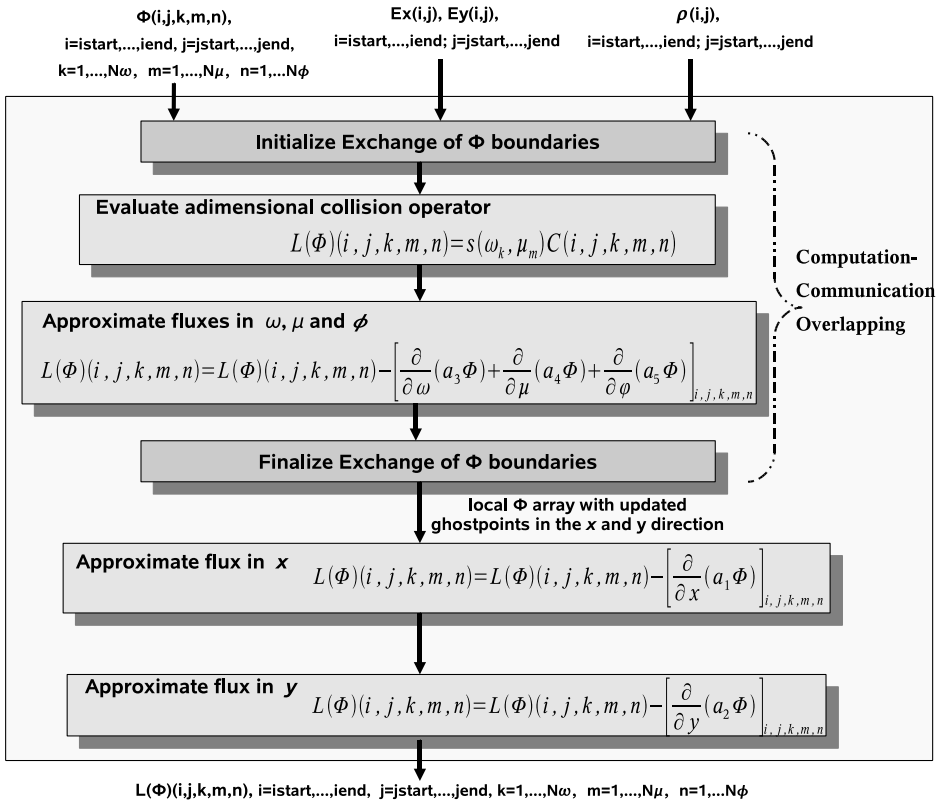


Fig. 8. Data Flow for the Computation of $\boldsymbol{L}(\Phi)$ with computation-communication overlapping.

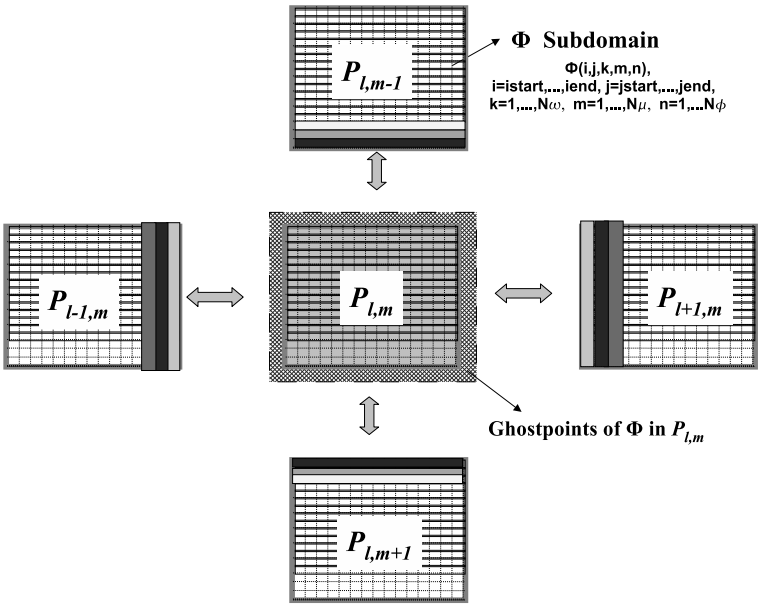


Fig. 9. Remote communication pattern to approximate the fluxes in the x and y directions.

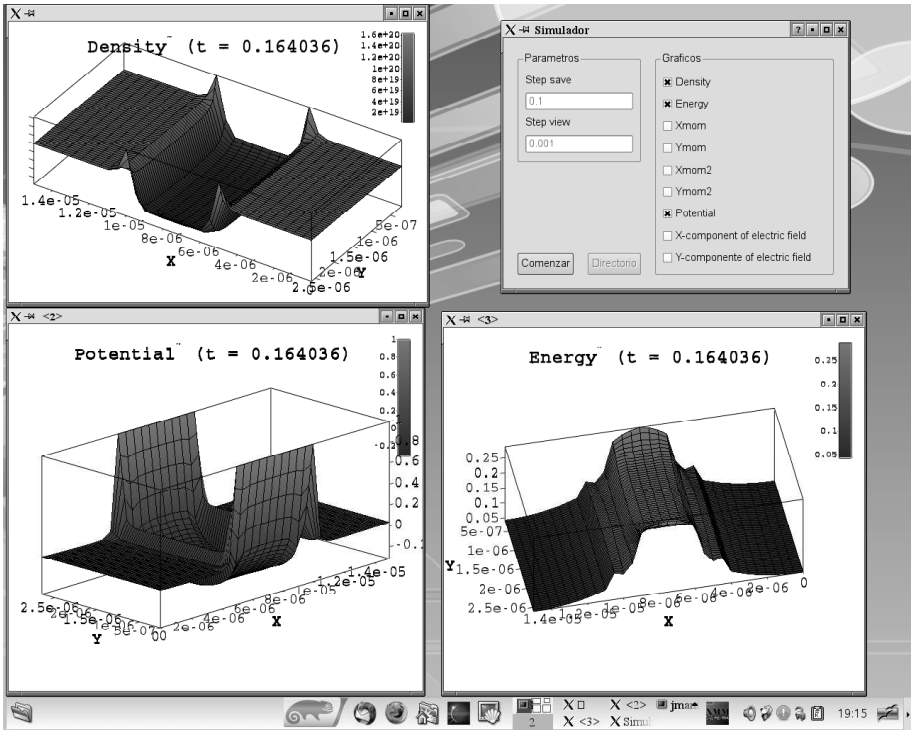


Fig. 10. Full screenshot showing the use of the visualization system for a DG-MOSFET device.

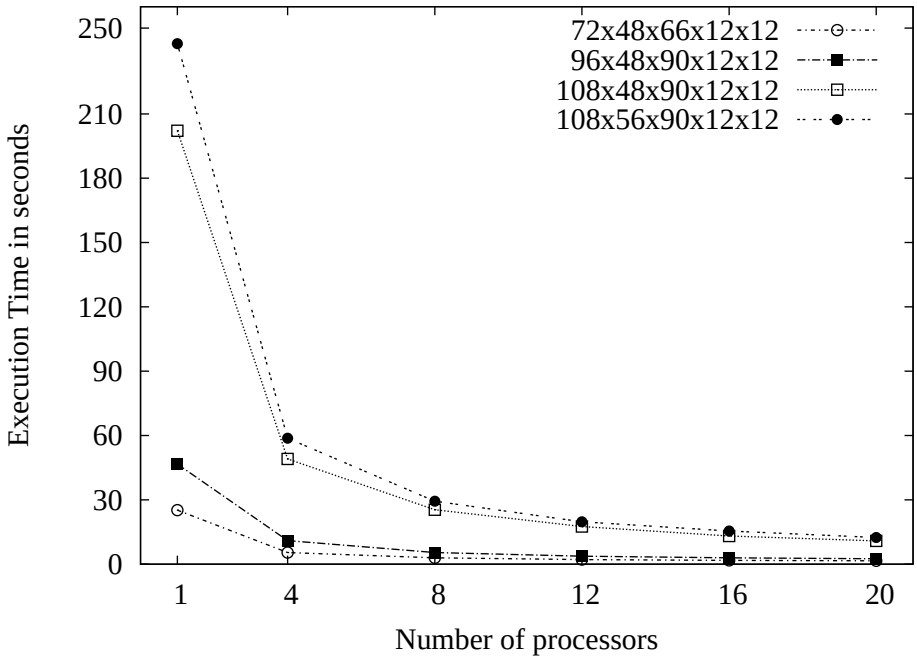


Fig. 11. Execution times when the number of processors is increased for a MOSFET device.

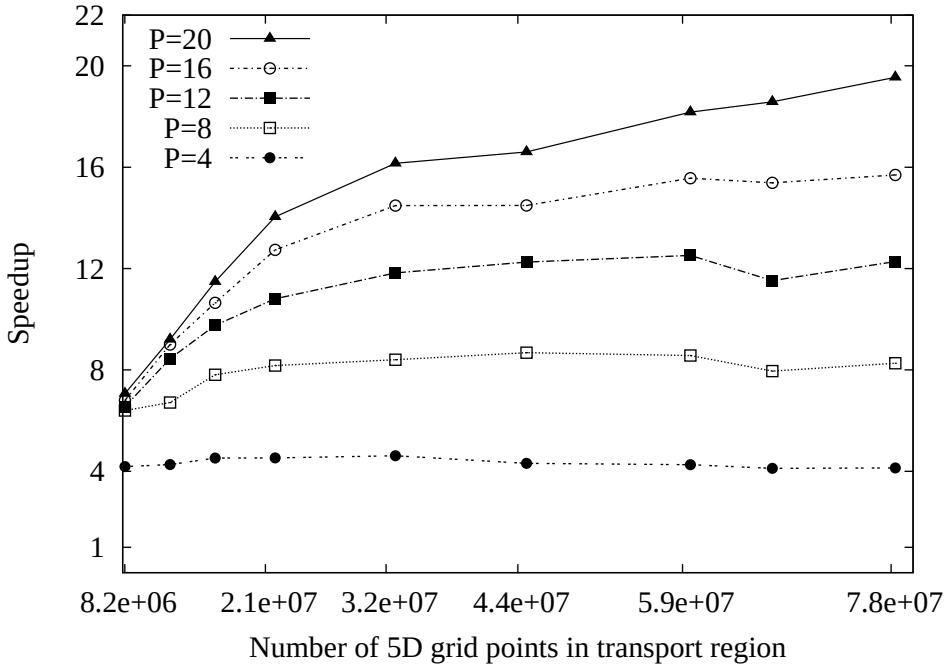


Fig. 12. Parallel speedup when the number of 5D mesh points is increased for a MOSFET device.

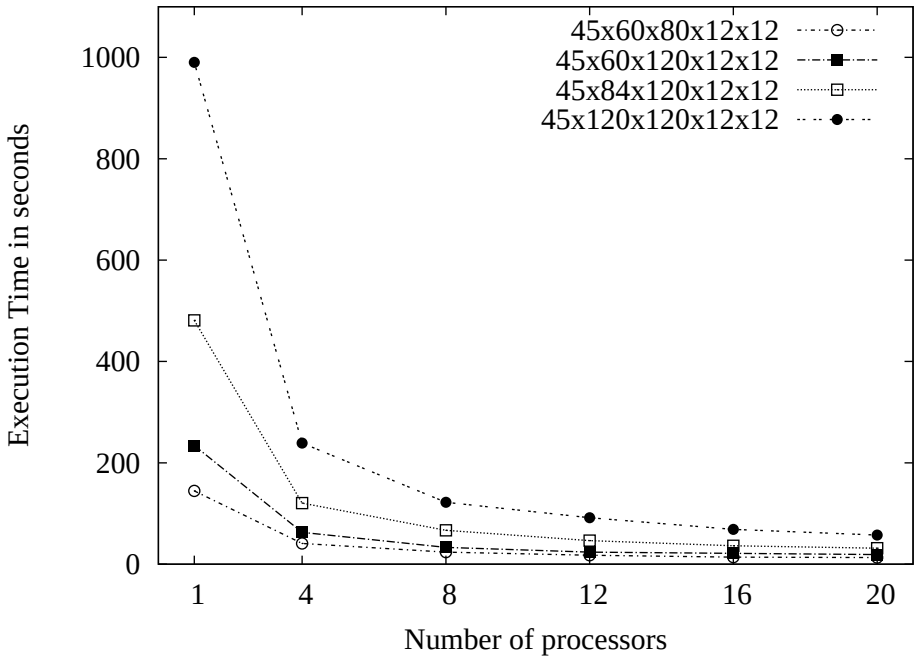


Fig. 13. Execution times when the number of processors is increased for a DG-MOSFET device.

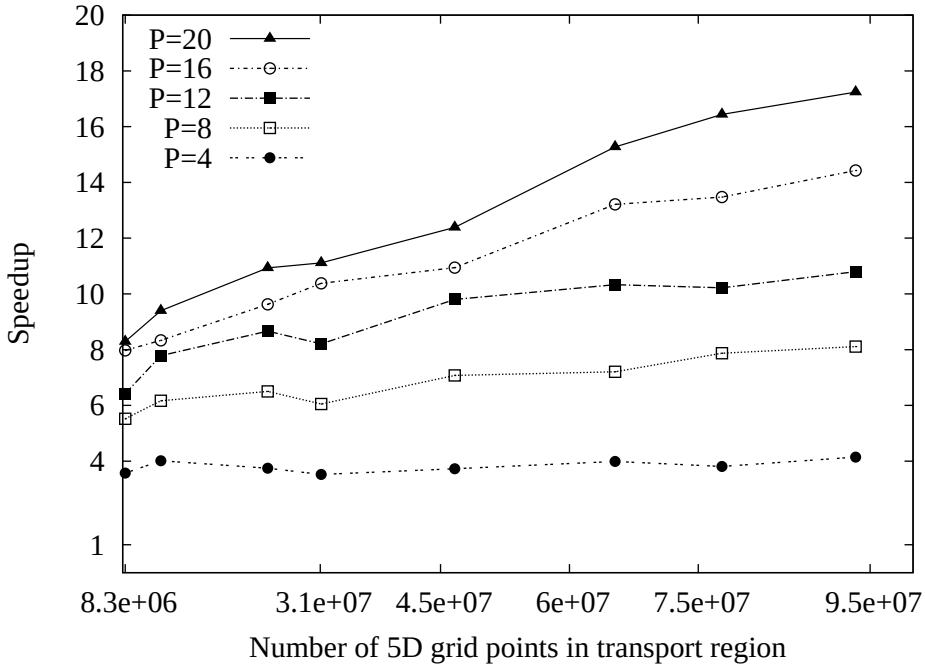


Fig. 14. Parallel speedup when the number of 5D mesh points is increased for a DG-MOSFET device.

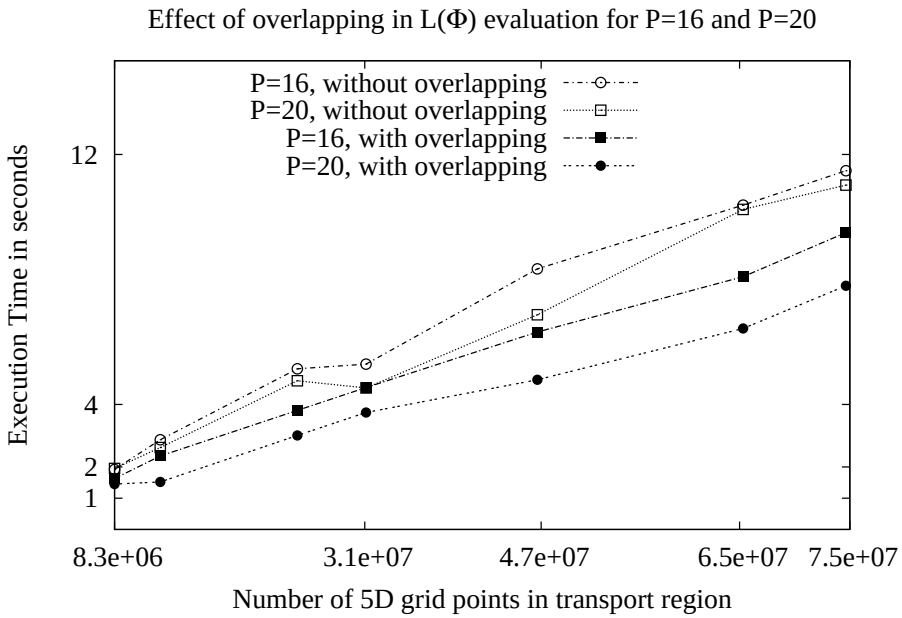


Fig. 15. Effect of overlapping communication with computation in the evaluation of $\boldsymbol{L}(\Phi)$ at one time step with $P = 16$ and $P = 20$ for a DG-MOSFET device.

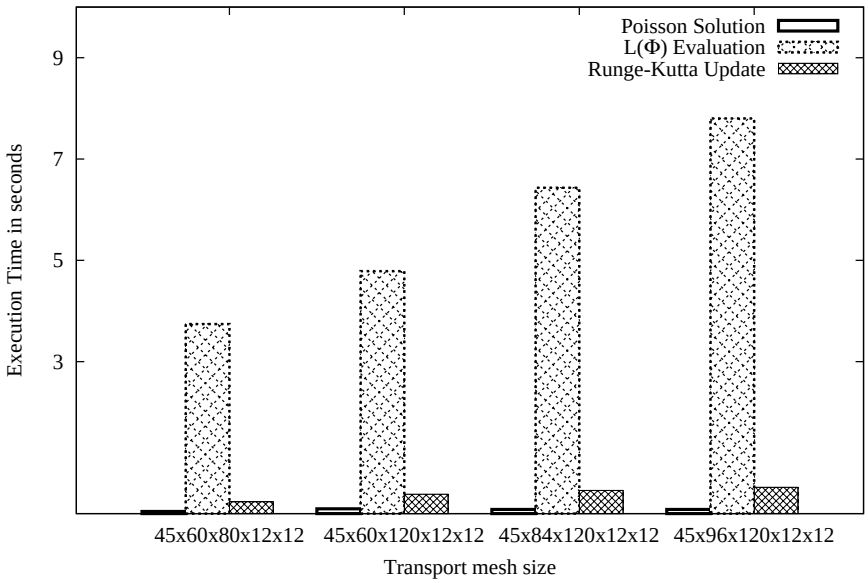


Fig. 16. Contribution of the most important computing phases to the execution time at one time step for different transport mesh sizes and $P = 20$ when a DG-MOSFET device is simulated.