Computer Physics Communications 142 (2001) 285–289

Computer Physics Communications

www.elsevier.com/locate/cpc

A new parallel adaptive finite volume method for the numerical simulation of semiconductor devices

Yiming Li [a,*], Jinn-Liang Liu [b], Tien-Sheng Chao [c], S.M. Sze [a]

[a] *Department of Electronics Engineering and Institue of Electronics, National Chiao Tung University, Hsinchu 300, Taiwan*
[b] *Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan*
[c] *National Nano Device Laboratories, Hsinchu 300, Taiwan*

Abstract

Based on adaptive finite volume approximation, a posteriori error estimation, and monotone iteration, a novel system is proposed for parallel simulations of semiconductor devices. The system has two distinct parallel algorithms to perform a complete set of I–V simulations for any specific device model. The first algorithm is a domain decomposition on 1-irregular unstructured meshes whereas the second is a parallelization of multiple I–V points. Implemented on a Linux cluster using message passing interface libraries, both algorithms are shown to have excellent balances on dynamic loading and hence result in efficient speedup. Compared with measurement data, computational results of sub-micron MOSFET devices are given to demonstrate the accuracy and efficiency of the system.

PACS: 73.40.Ty; 73.40.Qv; 02.70.Fj; 02.70.-c

Keywords: Adaptive FVM; Parallel semiconductor device simulation; Load balancing

1. Introduction

Parallel numerical simulation of semiconductor devices has been proven to be an indispensable tool for fast characterization and optimal design of semiconductor devices (see [1] and references therein). Adaptive computation is currently one of the major concepts in large-scale simulations [2]. Considerable efforts have been directed to the development of high-performance computational techniques for semiconductor physics and devices. We propose here a prototype of parallel system for semiconductor device simulation. The main features of the system are adaptive finite volume method (FVM) with 1-irregular mesh refinement strategy, a posteriori error estimation, constructive monotone iteration [3], domain decomposition, and parallel I–V computations. Implemented on a Linux-cluster with message passing interface (MPI), the system has been tested on, such as PN diode, MOSFET, and SOI devices [4].

For most practical semiconductor devices, the physical quantities such as potential and electron densities exhibit extreme jump layers particularly in the neighborhood of *p-n* junctions [4]. The presence of layers results in highly unstructured grids and hence in the complexity of coding structure and parallelization, which can be alleviated by exploiting object-oriented programming (OOP) principles [5,6] and by developing suitable parallel algorithms with good dynamic load balancing. Two distinct parallel algorithms are proposed in this paper. They are designed to perform

* Corresponding author.
E-mail address: ymli.ee87g@nctu.edu.tw (Y. Li).

0010-4655/01/$ – see front matter
PII: S0010-4655(01)00347-2

a complete set of I–V simulations for any specific device model. The first algorithm is a domain decomposition on 1-irregular unstructured meshes whereas the second is a parallelization of multiple I–V points. Compared with measurement data, simulation results of LDD N-MOSFET [4] are given to show the accuracy and effectiveness of the system.

2. Semiconductor device model

Hydrodynamic (HD) equations are used to model submicron MOSFET devices in which hot electron and non-local effects are main concerns [4,7]. The following is a commonly used model [8,9]:

$$\Delta\phi = \frac{q}{\varepsilon_s}\left(n - p + N_A^- - N_D^+\right), \tag{1}$$

$$\frac{1}{q}\nabla \cdot J_n = R(n, p), \tag{2}$$

$$\frac{-1}{q}\nabla \cdot J_p = R(n, p), \tag{3}$$

$$\nabla \cdot S_n = J_n \cdot E - n\left(\frac{\omega_n - \omega_0}{\tau_{nw}(T_n)}\right), \tag{4}$$

$$\nabla \cdot S_p = J_p \cdot E - p\left(\frac{\omega_p - \omega_0}{\tau_{pw}(T_p)}\right), \tag{5}$$

where ϕ is the electrostatic potential, n and p carrier concentrations, N_A^- and N_D^+ ionized doping profiles, J_n and J_p carrier current densities, S_n and S_p carrier energy fluxes, $R(n, p)$ the generation recombination rate, $E = -\nabla\phi$ the electric field, ω_n and ω_p carrier energies, τ_{nw} and τ_{nw} carrier energy relaxation times, and $\omega_0 = \frac{3}{2}k_B T_L$ the thermal equilibrium carrier energy. Explicitly, J_n, J_p, S_n, and S_p are as follows:

$$J_n = -q\mu_n n\nabla\phi + qD_n\nabla n + n\mu_n k_B \nabla T_n, \tag{6}$$

$$J_p = -q\mu_p p\nabla\phi - qD_p\nabla p - p\mu_p k_B \nabla T_p, \tag{7}$$

$$S_n = \frac{J_n}{-q}\omega_n + \frac{J_n}{-q}k_B T_L + Q_n, \tag{8}$$

$$S_p = \frac{J_p}{q}\omega_p + \frac{J_p}{q}k_B T_L + Q_p. \tag{9}$$

Here μ_n, μ_p, D_n, D_p, Q_n, and Q_p are the carrier mobility, diffusion coefficient, and heat flow, respectively. The model is subject to suitable boundary conditions. In this simulation, based on Fermi–Dirac statistics [4, 10,11], the model also can be expressed in terms of quasi-Fermi levels instead of carrier concentrations.

3. Adaptive numerical methods

We now briefly outline the adaptive algorithm and numerical methods that are implemented in our device simulation system.

Adaptive Algorithm.

Step 1. Initialization and initial mesh generation.
Step 2. Construction of data structure on the current mesh.
Step 3. Outer loop iteration (i.e. Gummel's iteration [3,10]).
  Step 3.1. Inner loop iteration on FV solution of Eq. (1) for ϕ.
  Step 3.2. Inner loop iteration on FV solution of Eq. (2) for n.
  Step 3.3. Inner loop iteration on FV solution of Eq. (3) for p.
  Step 3.4. Computation of J_n and J_p.
  Step 3.5. Inner loop iteration on FV solution of Eq. (4) for T_n.
  Step 3.6. Inner loop iteration on FV solution of Eq. (5) for T_p.
Step 4. A posteriori error estimation.
Step 5. Run mesh refinement and go to Step 2 if stopping criteria aren't satisfied.
Step 6. Postprocessing.

A description of FV approximation and a posteriori error estimation for linear elliptical partial differential equations with unstructured mesh can be found in [5]. For semiconductor device simulation, the error estimation has to be modified to account for the fundamental principle of flux and charge conservation. For each decoupled equation, FVM results in a system of nonlinear algebraic equations which are solved by a monotone iterative scheme similar to that of [3]. The refinement process is guided by local error indicators that are based on element-by-element calculations of the maximum gradient of electrostatic potential ϕ and the variation of current densities J_n and J_p.

4. Parallel algorithms

Due to the nature of p-n junction properties in semiconductor device, meshes adaptively generated by the refinement procedure are highly unstructured and consequently lead to complicated data structures. This

causes the load balancing a difficult task among multiple processors. In connection with special properties of the monotone iterative method [3], two parallel algorithms are proposed here. The first algorithm is a domain decomposition on 1-irregular unstructured meshes whereas the second is a parallelization of multiple I–V points. Note that the I–V curve of a submicron device is the main objective of device simulation which in general require tremendous amount of working-time for various parameters. The constructed Linux-cluster system and network configuration are including, such as cluster, NFS, NIS, UDP, server, TCP/IP, and Internet in this work. The cluster contains 8 PCs in this study; files access and share are through network file system (NFS) and network information system (NIS). The user datagram protocol (UDP) that controlled by MPI is applied to the short distance fast communication.

Domain Decomposition Algorithm.

Step 1. Initialize the MPI environment and configuration parameters.

Step 2. Generate a tree data structure of the current mesh.

Step 3. All nodes are numbered in accordance with refinement levels and the critical nodes are identified. Count the total number of regular nodes of the mesh.

Step 4. In a server, uniformly partition the nodes into two categories. One corresponds to the bulk region and another to the surface region. Dynamically assign nodes in each category to different processors (clients) in which the same iterative solver is installed. The assignment is performed along x- or y-direction (from left to right and bottom to top) in 2D device domain. In the neighborhood of p-n junction one may have to change the assignment direction for obtaining a better load balancing configuration if necessary.

Step 5. All processors perform inner and outer iterations in a synchronized way.

Step 6. Computed data that is relevant to the relevant neighboring processors is exchanged among the processor via MPI protocol until the stopping criteria of both iterations are satisfied.

Step 7. Each processor computes local error indicators on an element-by-element basis and subdivides each one of elements that exhibit large errors into four sub-elements provided that the global stopping criteria are not satisfied.

Step 8. Repeat steps 2–7, if the refinement process is invoked. Otherwise, stop the iteration and run postprocessing for the final data.

Parallel I–V Algorithm.

Step 1. Initialize the MPI environment and configuration parameters for all processors.

Step 2. Corresponding to a set of I–V points that are to be calculated for some device model, a queue of jobs with various biasing voltages is created in the server. Each job represents a complete process of adaptive computations as described above.

Step 3. The server assigns a job to the next available processor until the queue is empty.

Step 4. Each processor performs its own job (an I–V point) independent of the other (another I–V point corresponding to a set of different boundary conditions) due to the global convergent property of the monotone iterative method [3].

5. Numerical results

Organized into two examples, we now present some numerical results of our device simulation. The first example is given to show the effectiveness of the adaptive algorithm and the efficiency of the domain decomposition algorithm. A 0.35 μm LDD N-MOSFET device model with biasing conditions $V_{DS} = 2V$ and $V_{GS} = 2V$ is used for this example. The adaptive process begins with an initial mesh of 16 elements on the solution domain and ends with the final mesh as shown in Fig. 1. The

Table 1
Efficiency and CPU time for parallel domain decomposition simulation on a 6-processors Linux-cluster

Nodes	Sequential time (s)	Parallel time (s)	Speedup	Efficiency
2000	29	8	3.62	60.4%
4000	101	28	3.61	60.1%
8000	1250	306	4.08	68.0%
16,000	5233	1192	4.39	73.1%
22,000	9878	2054	4.81	80.2%

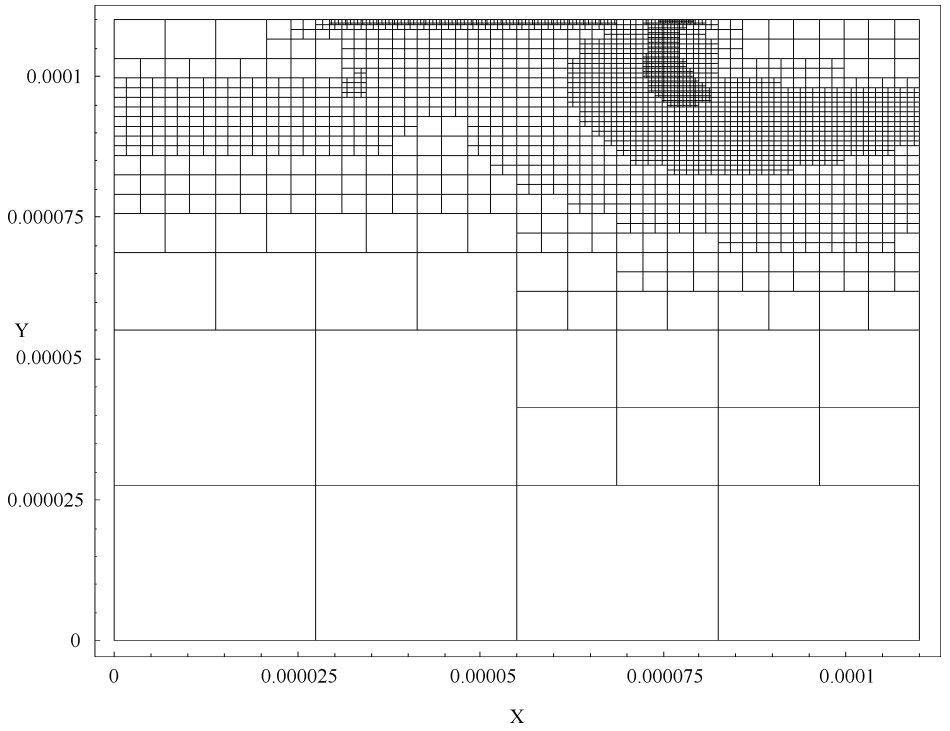

Fig. 1. Adaptive final refined mesh for the simulated device.

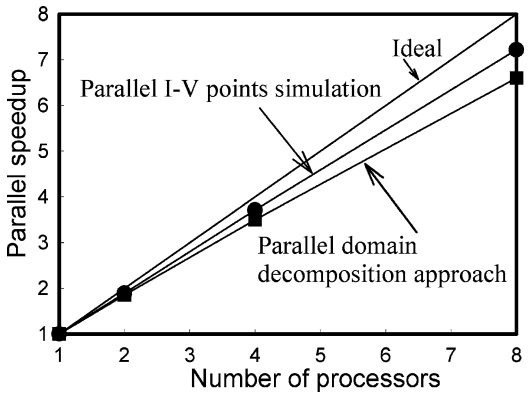

Fig. 2. Speedup for parallel domain decomposition and I–V points calculation algorithms.

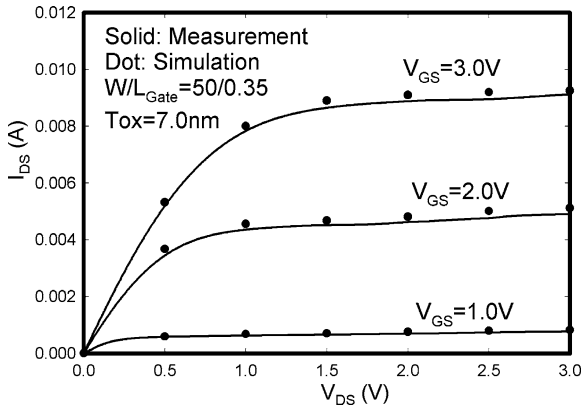

Fig. 3. Measured and simulated device I–V characteristics.

same adaptive process was first performed on a single processor then on a Linux cluster of 6 processors. Table 1 shows a comparison of the performance in CPU time and efficiency. The lower line, in Fig. 2, indicates the achieved speedup for a typical mesh with 22,000 nodes on an 8-processors Linux-cluster.

For the same device, the second example presents the excellent performance of the parallel I–V algorithm applied to compute a set of 189 I–V points on

the Linux-cluster. The speedup factor is larger than 7 as shown in Fig. 2 (middle line). A subset of those simulated I–V points are plotted in Fig. 3 along with the fabricated and measured data obtained from National Nano Device Laboratories, Taiwan.

Acknowledgements

This work was supported in part by the National Science Council of Taiwan under contract numbers NSC-89-2215-E-317-009.

References

[1] N.R. Aluru et al., IEEE Trans. CAD 15 (1996) 1029–1047.
[2] V. Verfurth, A Review of a Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques, Teubner-Wiley, Stuttgart, 1996.
[3] Y. Li et al., in: Proc. IEEE Int. Symp. VLSI-TSA, 1999, pp. 27–30.
[4] S.M. Sze, Physics of Semiconductor Devices, 2nd edn., Wiley-Interscience, New York, 1981.
[5] T. Gallonet et al., SIAM J. Num. Anal. 37 (2000) 1935.
[6] J.-L. Liu et al., Appl. Num. Math. 21 (1996) 439–467.
[7] M. Ieong, T.-W. Tang, IEEE Trans. ED 44 (1997) 2242–2251.
[8] P. Degond et al., SIAM J. Sci. Comp. 22 (2000) 986–1007.
[9] K. Bløtekjer, IEEE Trans. ED 17 (1970) 38–47.
[10] W. Jerome, SIAM J. Appl. Math. 45 (1985) 565–590.
[11] J.W. Slotboom, IEEE Trans. ED 20 (1973) 669–679.