ELSEVIER

Computer Physics Communications 147 (2002) 697–701

Computer Physics Communications

www.elsevier.com/locate/cpc

A domain partition approach to parallel adaptive simulation of dynamic threshold voltage MOSFET

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

[a] *National Nano Device Laboratories, Hsinchu, Taiwan*
[b] *Department of Electrophysics, National Chiao Tung University, Hsinchu 300, Taiwan*
[c] *Institute of Electronics, National Chiao Tung University, Hsinchu 300, Taiwan*

Abstract

In this paper, we present a dynamic domain partition simulation technique for parallel numerical solutions of semiconductor device equations. Based on the adaptive finite volume method, a posteriori error estimation, and monotone iterative algorithm, this dynamic load balancing approach has been successfully developed and implemented on a Linux cluster with message passing interface library. The developed simulator is then applied to calculate the physical characteristics of deep submicron dynamic threshold voltage MOSFET (DTMOS). We simulate DTMOS with two different parallel algorithms: (1) 2D dynamic load balancing for parallel domain decomposition; (2) parallel I–V point simulation. Benchmark results show that a well-designed load balancing simulation can reduce the execution time up to an order of magnitude. Compared with the measured data, the simulated results for a 0.08 μm DTMOS are demonstrated to show the accuracy and efficiency of the method.

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

Keywords: Semiconductor device simulation; DTMOS; Parallel adaptive FVM; Dynamic load balancing

1. Introduction

Computer-aided simulation for semiconductors provides the capability for a software-driven approach to explore new physics and devices [1]. In recent years, a novel device structure the so-called dynamic threshold voltage metal oxide semiconductor field effect transistor (DTMOS), has been studied for an alternative way to improve the conventional submicron MOSFET's I–V characteristics at ultralow supply voltage applications [2–4]. Experimental fabrication and measurement have illustrated primarily that the DTMOS scheme appears to be very promising for future low-power and high-speed circuit applications [3,4]. Some theoretical studies for this new device have been of great interest [5] with simplified compact or conventional numerical modeling approaches.

In this paper, using the dynamic load balancing approach we propose a new parallel adaptive DTMOS simulation method and apply it to rapidly simulate the physical characteristics of 0.08 μm DTMOS. This simulator based on the adaptive finite volume (FV) [6,7], domain decomposition, error estimation,

* Corresponding author. Yiming Li is with Microelectronics and Information Systems Research Center, National Chiao Tung University, Hsinchu 300, Taiwan.
E-mail address: ymli@cc.nctu.edu.tw (Y. Li).

0010-4655/02/$ – see front matter
PII: S0010-4655(02)00368-5

and monotone iterative (MI) methods [8–14] has been developed and implemented on a 16-processor Linux cluster with message passing interface (MPI) library. Two different parallel algorithms to simulate a DTMOS are proposed; one is a 2D dynamic load balancing for parallel domain decomposition, and the other is a parallel I–V point simulation. The practical implementation shows that a well-designed load balancing simulation can reduce significantly the execution time up to an order of magnitude.

Hydrodynamic (HD) DTMOS equations are discretized first with the FV method and hence a large-scaled system of nonlinear algebraic equations is obtained. The nonlinear system is then directly solved with the MI method. This method for solution of the nonlinear system derived from semiconductor equations has been successfully developed and applied by us in various earlier devices simulations [8–14]. To refine the mesh adaptively, a physical-based unstructured mesh refinement rule with a posteriori error estimation has also been developed for the quality control of computed results. Compared with measured data, simulated results for a 0.08 μm DTMOS are demonstrated to show the accuracy and efficiency of the method. The parallel methodology developed here shows that it is a quite feasible computing alternative for providing a fast characterization of semiconductor physics and devices with complicated structures. This paper is organized as follows. Section 2 states the device model and the computational algorithms. Section 3 presents the simulation results. Section 4 draws the conclusion.

2. Device model and simulation methods

2.1. A semiconductor device model

We simulate 0.08 μm DTMOS with a HD model. This nonlinear model is often applied to study hot carrier and non-local effects for deep submicron semiconductor devices [15–18].

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

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

$$\nabla \cdot S_n = J_n \cdot E - n\left(\frac{\omega_n - \omega_0}{\tau_{nw}(T_n)}\right), \tag{3}$$

where ϕ is the electrostatic potential, n and p are the carrier concentrations, N_A^- and N_D^+ are the ionized doping profiles. For electrons, J_n is the current density, S_n is the energy flux, and $R(n, p)$ is the generation recombination rates. The E is the electric field, ω_n is the energy, τ_{nw} is the energy relaxation time, and ω_0 is the thermal equilibrium carrier energy. The J_n and S_n are as follows:

$$J_n = -q\mu_n n\nabla\phi + qD_n\nabla n + n\mu_n k_B \nabla T_n, \tag{4}$$

$$S_n = \frac{J_n}{-q}\omega_n + \frac{J_n}{-q}k_B T_L + Q_n, \tag{5}$$

were μ_n, D_n, and Q_n are electron mobility, diffusion coefficient, and heat flow, respectively [2–5,15–18]. There is a similar model for holes. The model is subject to proper boundary conditions and the unknowns to be solved are ϕ, n, and T_n.

2.2. Numerical methods

We now state the adaptive FV and MI computing procedures for the DTMOS simulation. The adaptive FV algorithm is outlined as follows:

Step A1. Initialization and initial mesh generation for DTMOS.
Step A2. Construction of the data structure for the specified mesh.
Step A3. Outer (Gummel's) loop iteration [19].
  Step A3.1. Inner (MI) loop iteration of Eq. (1) for ϕ.
  Step A3.2. Inner (MI) loop iteration of Eq. (2) for n.
  Step A3.3. Inner (MI) loop iteration of Eq. (3) for T_n.
Step A4. A posteriori error estimation.
Step A5. Perform mesh refinement and go to A2, if stopping criteria are not satisfied.
Step A6. Postprocessing.

For each decoupled equation, the system of nonlinear algebraic equation results from FV discretization is solved by the MI method. 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 carrier density and temperature. Our error estimation

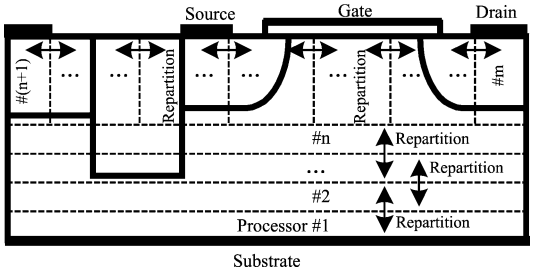

Fig. 1. Domain decomposition for sub 0.1 μm DTMOS. Achieved speedup for parallel SP simulation.

for DTMOS simulation shows good refinement efficiency.

2.3. Parallel algorithms

The adaptation of the mesh produces imbalances in the jobs assigned to the processor. Because of the irregular load requirements of parallel adaptive computation, a mesh must also be repartitioned dynamically for processors during runtime. Based on the monotone convergent property of the MI method, we propose two different parallel algorithms for DTMOS simulation; the first algorithm developed with the dynamic partition technique is for the parallel domain decomposition approach, and the other is a novel parallel I–V point simulation. The constructed Linux cluster contains 16 PCs; file access and share are through a network file system and a network information system. The user datagram protocol controlled by MPI is applied to short-distance fast communication.

Algorithm 1. Parallel domain decomposition with dynamic partition.

As shown in Fig. 1, we state a computational procedure for parallel domain decomposition steps:

Step P1. Initialize MPI environment and configuration parameters.
Step P2. Based on 1-irregular meshing rule, generate a tree structure and mesh.
Step P3. Count number of all regular mesh points. Apply dynamic partition algorithm for each simulation. Number all nodes, and identify boundary and critical points.
Step P4. All processors perform inner and outer iteration loops synchronously, and all assigned jobs are solved with MI solver independently. The computed data communicates with MPI protocol.
Step P5. Perform error estimation for all elements and run adaptive refinement for those elements having large errors.
Step P6. Repeat steps P3–P5 until errors of all elements are less than a specified error bound.
Step P7. Host collects all computed results and stops the MPI environment.

The dynamic partition algorithm applied for load balancing in *Step* P3 is:

Step D1. Count number of total nodes.
Step D2. Find out the optimal number of processors based on the number of nodes.
Step D3. Calculate how many nodes should be assigned to each processor by dividing total nodes by the optimal number of processors.
Step D4. Along the x- or y-direction in the 2D DTMOS domain, search (from left to right and bottom to top) and assign nodes to these processors sequentially. Repeat this step until all nodes have been assigned.
Step D5. In the neighborhood of the p-n junction, if necessary, one may have to change the search and assign the direction for obtaining a better load balancing configuration.

Algorithm 2. Parallel I–V point simulation.

With the robust and globally convergent behavior of the MI method, a parallel I–V point simulation technique is also proposed to rapidly compute full I–V curves for DTMOS simulation. This approach simulates I–V points independently. Each job represents a complete process of adaptive computations as described above. The outline of this approach is:

Step I1. Initialize MPI environment and configuration parameters.
Step I2. Corresponding to all I–V points to be calculated for DTMOS, server creates required I–V job queue.
Step I3. Server sends out I–V points to available processors.
Step I4. Client calculates assigned I–V points independently.

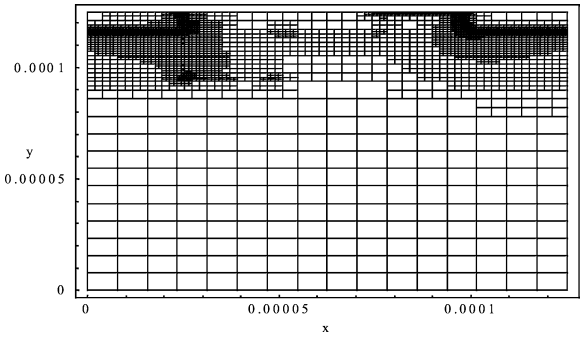

Fig. 2. A refined mesh for DTMOS simulation.

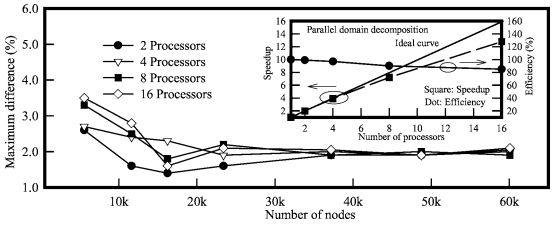

Fig. 3. Achieved dynamic loading for DTMOS simulation.

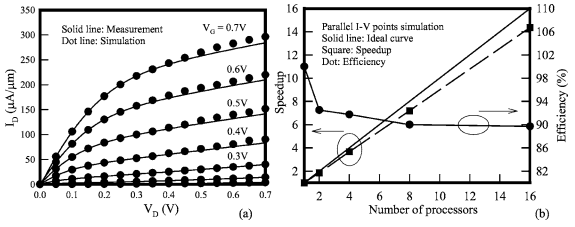

Fig. 4. (a) Measured and simulated DTMOS I–V curves. (b) Speed up and efficiency of I–V point simulation.

Step I5. If the job is done, the client sends back computed data to the server and calls for next computation.

Step I6. Repeat I3–I5 until all jobs are done.

Step I7. Stop MPI environment.

3. Results and discussion

Fig. 2 shows an adaptive refinement mesh for a DTMOS at $V_D = V_G = 1.0$ V. The final mesh indicates the efficiency of the adaptive method. The speed up, efficiency (the inserted figure) and maximum difference of the dynamic load balancing algorithm for domain decomposition are shown in Fig. 3. The refined node is about 24,000. Speed up is the ratio of simulation time on a processor to that on multiple processors. Efficiency equals to speed up divided by the number of processors. Maximum difference percentage is maximum difference (in simulation time) divided by maximum simulation time. Fig. 4 shows the accuracy of the method for the 0.08 μm DTMOS I–V curve simulation. Fig. 4(b) indicates that the parallel I–V point simulation has higher speed up and efficiency than parallel domain decomposition.

4. Conclusion

We have presented a parallel adaptive DTMOS simulation using the dynamic load balancing approach. The 2D dynamic load balancing for parallel domain decomposition and the parallel I–V point simulation have been successfully developed for DTMOS simulation. Achieved benchmarks showed an excellent performance with respect to the number of processors on a Linux cluster with MPI library. Simulation results for 0.08 μm DTMOS have been presented to show the accuracy and efficiency of the method.

Acknowledgements

This work was supported in part by the National Science Council of Taiwan under Contract No NSC 90-2112-M-317-001.

References

[1] R.W. Dutton et al., IEEE T. CAD 19 (2000) 1544–1560.
[2] S.M. Sze, Physics of Semiconductor Devices, Wiley-Interscience, New York, 1981.
[3] S.J. Chang et al., IEEE T. ED 47 (2000) 2379–2384.
[4] S.J. Chang et al., IEEE EDL 21 (2000) 127–129.
[5] F. Assaderaghi et al., IEEE T. ED 44 (2000) 414–422.
[6] R.S. Varga, Matrix Iterative Analysis, Springer, New York, 2000.
[7] T. Gallouet et al., SIAM J. Numer. Anal. 37 (2000) 1936–1972.37 (2000)
[8] Y. Li et al., Comput. Phys. Commun. 142 (2000) 285–289.
[9] Y. Li et al., in: Proc. IEEE Int. Symp. VLSI Tech., Systems, and Appl., 1999, pp. 27–30.
[10] Y. Li et al., in: Proc. IEEE 15th Int. Parallel and Distributed Processing Symposium, 2001, pp. 17.3.1–17.3.6.

[11] N. Mastorakis et al. (Eds.), Advances in Scientific Computing, Computational Intelligence and Applications, WSES Press, 2001, pp. 54–59.
[12] Y. Li et al., in: Tech. Proc. Int. Conf. Modeling and Simulation of Microsystem, 2001, pp. 562–566.
[13] Y. Li et al., in: Proc. 2nd WSEAS Int. Multiconf. on Applied and Theoretical Math., 2001, pp. 6201–6206.
[14] Y. Li, WSEAS Trans. on Systems 1 (2002) 68–73.
[15] K. Blotekjaer, IEEE T. ED 17 (1970) 38–47.
[16] P. Degond et al., SIAM J. Sci. Comp. 22 (2000) 986–1007.
[17] M. Ieong et al., IEEE T. ED 44 (1997) 2242–2251.
[18] T.W. Tang, IEEE T. ED 31 (1984) 1912–1914.
[19] H.K. Gummel, IEEE T. ED 11 (1964) 455–465.