Massively Parallel Computation Using a Splitting-Up Operator Method for Three-Dimensional Device Simulation

Shinji Odanaka and Tatsuo Nogi

Abstract— **This paper presents a new parallel algorithm and performance results of iterative solution methods for three-dimensional MOSFET simulation with Gummel's method. A splitting-up operator method is proposed for incomplete factorization of sparse matrices arising from semiconductor device equations, suitable for parallel computations. This method is combined with the conjugate gradients and BiCGSTAB procedure to obtain a new parallel version of the iterative solution methods. Natural parallelism is realized by developing the solution method according to the natural ordering. In large-scale simulations of greater than 100 000 grid nodes, the high parallel efficiency level over 90% can be achieved using a new-type massively parallel computer: ADENART with up to 256 processors. The real performance of the solution methods is superior to those calculated by the vectorized version of the ICCG and ILUBiCGSTAB methods using a vector-type supercomputer.**

I. Introduction

RECENT ADVANCES in VLSI technology and manufacturing are creating the need of the three-dimensional simulation, physics-based modeling, and Technology CAD design environment. High-performance computing makes these computation-intensive numerical simulations practical. The vector-type supercomputer had an impact on the development of three-dimensional process and device CAD softwares [1], [2]. To achieve the high performance of the vector-type supercomputer, the vectorized versions of preconditioned iterative methods were developed by using the hyperplane ordering. In the iterative solution method for process and device simulations, the Incomplete Cholesky Conjugate Gradients (ICCG) algorithm [3] has been widely used for solving symmetric matrix problems and two algorithms, Incomplete LU BiConjugate Gradients (ILUBiCG) [4] and Incomplete LU Conjugate Gradients Squared (ILUCGS) [5], have been used for solving nonsymmetric matrix problems.

Another approach to the computation-intensive numerical simulations is to develop iterative solution methods using massively parallel computers. In general, parallel versions of the solution method are realized by subdividing the matrix problem into disjoint blocks of equations and assigning each processor the task of handling one or more blocks. In an earlier work of device simulation using parallel computation, the nested dissection ordering [6] was applied to the ICCG method for solving the Poisson equation in three-dimensions on the Intel hypercube [7]. Red-black ordering [8] was used in a regular grid problem to realize the parallel computation of the ILUCGS method on the Connection Machine [9]. However, the parallel efficiency and the convergence behavior of the iterative solution method strongly depend on the matrix partitioning scheme and the number of blocks [10], [11]. In such matrix problems, the ordering modifies the quality of the incomplete factorization preconditioner, which results in change of the convergence behaviors of the preconditioned algorithms. Even in irregular grid problems the parallelization has been attempted by solving the matrix problem, where the rows of the matrix are reordered [12]. In addition, it should be noted that the serial nature of computation in incomplete LU decomposition remains in both regular and irregular grid problems. To overcome the matrix problem with ordering for parallelization, a grid-based parallelization of the iterative solution method, which utilizes some regularity of the grid structure, should be discussed. Considering the serial nature of incomplete LU decomposition, incomplete factorization preconditioners suitable for parallel computations should also be investigated.

This paper presents a new parallel algorithm and performance results of the iterative solution method for three-dimensional device simulations on a massively parallel computer: ADENART (**A**lternating **D**irection **E**diting **N**exus **AR**ray sys**T**em). A grid-based parallelization of iterative solution method is described. A splitting-up operator method is proposed as an incomplete factorization preconditioner suitable for parallel computations. This method is combined with the conjugate gradients (CG) and BiCGSTAB acceleration to obtain a new solution method. In Section II, the preconditioned iterative solution method is described on the basis of the splitting-up operator method. An extension to irregular grid is briefly discussed. Section III gives an overview of the ADENART computer. Section IV realizes a straightforward implementation of the parallel algorithms according to the natural ordering. Section V discusses performance results for the new parallel version of the solution method using

Manuscript received April 19, 1993; revised December 13, 1994. This paper was recommended by Associate Editor S. G. Duvall.

S. Odanaka is with the Semiconductor Research Center, Matsushita Electric Industrial Co., Moriguchi, Osaka 570 Japan.

T. Nogi is with the Division of Applied Systems Science, Faculty of Engineering, Kyoto University, Kyoto 606 Japan.

IEEE Log Number 9411260.

a three-dimensional device simulator: SMART-II [13]. The convergence behavior and computational performance are investigated by using three-dimensional device simulations for a scaled 0.25 μm LDD n-MOSFET. The results are compared with those calculated by the vectorized versions of ICCG and ILUBiCGSTAB methods on Fujitsu VP-200. The parallel efficiency of the new solution method is discussed.

II. Splitting-Up Operator Method as Incomplete Factorization

We consider a linear system $Au = k$, which stems from the seven-point finite difference discretization of drift-diffusion device equations over a rectangular grid in three dimensions:

$$(Au)_{ijk} = a_{ijk}u_{ijk} - b_{ijk}u_{i-1jk} - c_{ijk}u_{i+1jk} - d_{ijk}u_{ij-1k} - e_{ijk}u_{ij+1k} - f_{ijk}u_{ijk-1} - g_{ijk}u_{ijk+1}. \quad (1)$$

In the natural ordering, the coefficient matrix A has a super-block tridiagonal form where each super-block is a block tridiagonal matrix. Then the blocks are tridiagonal matrices. Each row of the matrix A has at most seven nonzero elements. By extracting essential parts of a typical row, the matrix A is written in the convenient form

$$A = [(\mathbf{0}, (0, -f_{ijk}, 0), \mathbf{0})((0, -d_{ijk}, 0), (-b_{ijk}, a_{ijk}, -c_{ijk}), (0, -e_{ijk}, 0))(\mathbf{0}, (0, -g_{ijk}, 0), \mathbf{0})]. \quad (2)$$

For the solution of $Au = k$, the iterative process with CG acceleration is described as

$$C \cdot (u^{n+1} - u^n) = -\tau_{n+1}(Au^n - k) \quad (n = 0, 1, 2, \cdots) \quad (3)$$

where the matrix C is an approximate factorization of the matrix A and τ is determined by the conjugate gradients (CG) acceleration [14]. Then the linear system $Au = k$ has the same solution as

$$(AC^{-1}) \cdot Cu = k. \quad (4)$$

If the matrix AC^{-1} has a small condition number, the CG algorithm converges to a solution quickly.

Since the coefficient matrix arising from the Poisson equation in device simulation is symmetric and positive definite, the incomplete Cholesky (IC) decomposition has been successfully used as the preconditioning for the CG algorithm. The incomplete LU (ILU) decomposition is used for nonsymmetric matrices from the current continuity equations. In incomplete Cholesky and LU decompositions, the linear system $Cz = r$ can be solved much easier than $Au = k$. The preconditioning, however, includes a serial computation of triangular backward/forward substitutions. For vector-type supercomputers the high speed computation of such a serial computation was realized using the list vector method on the basis of a so-called hyperplane defined by all triples (i, j, k) for which $i + j + k =$ constant [15].

High-performance computing of the iterative solution method on massively parallel computers is more complicated. In general, the parallelism for matrix solution methods is realized by subdividing the sparse matrix into disjoint blocks of equations and assigning each processor while minimizing the communication overhead. In an earlier work [7], a parallel version of the ICCG method was developed on the basis of the nested dissection ordering to solve the Poisson equation in three-dimensions using the Intel hypercube. The matrix was partitioned into disjoint blocks and separators. However, the parallel efficiency decreases as the number of processors increases and hence the efficiency was reduced to 77.5% for the regular grid problem even when using only 16 processors. Parallelization of the preconditioned iterative method was attempted to solve a full set of drift-diffusion device equations in the rectangular grid [9]. Red-black ordering was applied to realize the parallel computation of the ILUCGS for the Connection Machine. The performance of the solution method was estimated to be 120 Mflops on a full 65536 b serial processors. However, this approach suffers from a degradation in the convergence rate of the preconditioned iterative method. The ordering was modified to improve the convergence property in the CGS method with the red-black ordering [11]. Although for the nonsymmetric matrix the minimum degree ordering with multiple elimination [16] is proposed, this ordering has not been applied to the device simulation. Recently, Pommerell *et al.* further developed parallelization of ICCG and ILUCGS applied to irregular finite-element grids using a distributed-memory parallel computer [12]. In numerical experiments of device simulation, a degradation in the convergence rate of the multicolor preconditioned iterative method is reported [17]. A recursive spectral approach for partitioning is developed to achieve good load balancing and to minimize the communication overhead [18]. The expense of partitioning operation should be discussed for device simulations [19].

We realize a natural parallelism by developing a new preconditioned iterative method according to the natural ordering, which is used for sequential machines. To achieve the high-level parallelism of this method on a massively parallel computer, a decomposition of "a product type" significantly different from the incomplete Cholesky and LU decompositions is proposed. Such a product type decomposition in the two-dimensional case has been proposed as an incomplete AD (Alternating Direction) decomposition for the symmetric matrix [20]. This approach is extended to the symmetric and nonsymmetric matrices in the three-dimensional problem. The incomplete factorization of A is defined by a product of three super-block tridiagonal matrices X, Y, and Z as follows:

$$C = XYZ, \quad (5)$$

where the matrices X, Y, and Z are given by

$$(Xu)_{ijk} = \alpha_{ijk}u_{ijk} - \frac{\beta_{ijk}}{\alpha^2_{i-1jk}}u_{i-1jk} - \frac{\gamma_{ijk}}{\alpha^2_{i+1jk}} \cdot u_{i+1jk}, \quad (6a)$$

$$(Yu)_{ijk} = \alpha_{ijk}u_{ijk} - \frac{\delta_{ijk}}{\alpha_{ijk}\alpha_{ij-1k}}u_{ij-1k} - \frac{\varepsilon_{ijk}}{\alpha_{ijk}\alpha_{ij+1k}} \cdot u_{ij+1k}, \quad (6b)$$

$$(Zu)_{ijk} = \alpha_{ijk}u_{ijk} - \frac{\xi_{ijk}}{\alpha^2_{ijk}}u_{ijk-1} - \frac{\eta_{ijk}}{\alpha^2_{ijk}}u_{ijk+1}. \quad (6c)$$

$C = XYZ$ is the natural "splitting" of C into its one-dimensional components. From the product XYZ, the incomplete factorization C can be expressed in the form

$$C = (D + \hat{A}_x)D^{-1}(D + \hat{A}_y)D^{-1}(D + \hat{A}_z) \tag{7}$$

where D is a diagonal matrix and $D + \hat{A}_x$, $D + \hat{A}_y$, and $D + \hat{A}_z$ are super-block tridiagonal matrices. These matrices are represented by the same form as (2).

$$D^{-1} = [(\mathbf{0},\mathbf{0},\mathbf{0})(\mathbf{0},(0,1/\alpha^3_{ijk},0),\mathbf{0})(\mathbf{0},\mathbf{0},\mathbf{0})], \tag{8a}$$

$$D + \hat{A}_x = [(\mathbf{0},\mathbf{0},\mathbf{0})(\mathbf{0},(-\beta_{ijk},\alpha^3_{ijk},-\gamma_{ijk}),\mathbf{0}),\ (\mathbf{0},\mathbf{0},\mathbf{0})] \tag{8b}$$

$$D + \hat{A}_y = [(\mathbf{0},\mathbf{0},\mathbf{0})((0,-\delta_{ijk},0),(0,\alpha^3_{ijk},0),\ (0,-\varepsilon_{ijk},0))(\mathbf{0},\mathbf{0},\mathbf{0})], \tag{8c}$$

$$D + \hat{A}_z = [(\mathbf{0},(0,-\xi_{ijk},0),\mathbf{0})(\mathbf{0},(0,\alpha^3_{ijk},0),\mathbf{0}),\ (\mathbf{0},(0,-\eta_{ijk},0),\mathbf{0})]. \tag{8d}$$

This incomplete factorization represents a generalized splitting-up operator method [15], which includes the alternating direction implicit (ADI) method introduced by Douglas, Peasman, and Rachford. The factorization is consistent with the computer architecture of the ADENART computer as mentioned later. For the coefficients α^3_{ijk}, β_{ijk}, γ_{ijk}, δ_{ijk}, ε_{ijk}, ξ_{ijk}, and η_{ijk} we further get a simple and effective incomplete factorization of A [21]:

$$\alpha^3_{ijk} = a_{ijk},\quad \beta_{ijk} = b_{ijk},\quad \gamma_{ijk} = c_{ijk},\quad \delta_{ijk} = d_{ijk},$$
$$\varepsilon_{ijk} = e_{ijk},\quad \xi_{ijk} = f_{ijk},\quad \text{and}\quad \eta_{ijk} = g_{ijk}.$$

Then, the incomplete factorization C is written as

$$C = (D + A_x)D^{-1}(D + A_y)D^{-1}(D + A_z) \tag{9}$$

where A_x, A_y, and A_z are off-diagonal matrices corresponding to partial differences in x-, y-, and z-directions, respectively. In the incomplete factorization, there are nonzero elements filled-in where corresponds to zero elements of A. These terms are given by two types of nonzero elements, bc/a and bcd/a^2. Fill-in is ignored in the incomplete factorization.

The solution of $Cz = r$ is easily calculated by solving block tridiagonal systems in the x-, y-, z-directions, respectively:

$$(D + A_x)\cdot z_i = r, \tag{10a}$$
$$(D + A_y)\cdot z_j = D\cdot z_i, \tag{10b}$$
$$(D + A_z)\cdot z_k = D\cdot z_j. \tag{10c}$$

The iterative solution method is developed by using the new incomplete factorization. When the matrix A is positive definite and symmetric, the CG algorithm is used for increasing the convergence rate of the splitting-up operator method. For the nonsymmetric matrix, this method is combined with the variants of BiCG acceleration. Recently, the Bi-CGSTAB has been proposed as a new method by van der Vorst [22]. A number of parameters in the algorithm are investigated in the device simulations to further improve the convergence behavior of this method [23].

The right-preconditioning of $Au = k$ leads to the following scheme for preconditioned Bi-CGSTAB. In this case, the residual corresponds to the vector r of the original system.

u_0 is an initial guess. Let $r_0 = k - Au_0$ and $p_0 = r_0$, then

$$\alpha_k = (r_0, r_k)/(r_0, A\hat{p}), C\hat{p} = p_k. \tag{11a}$$
$$s_{k+1} = r_k - \alpha_k \cdot A\hat{p} \tag{11b}$$
$$C\hat{t} = s_{k+1}, t_{k+1} = A\hat{t} \tag{11c}$$
$$\omega_k = (s_{k+1}, t_{k+1})/(t_{k+1}, t_{k+1}) \tag{11d}$$
$$u_{k+1} = u_k + \alpha_k\hat{p} + \omega_k\hat{t} \tag{11e}$$
$$r_{k+1} = s_{k+1} - \omega_k t_{k+1} \tag{11f}$$
$$\beta_k = (r_0, r_{k+1})/(\omega_k \cdot A\hat{p}) \tag{11g}$$
$$p_k = r_k + \beta_k(p_k - \omega_k \cdot A\hat{p}). \tag{11h}$$

Here the parameter ω is defined according to Bi-CGSTAB-P in [22]. This method requires two extra inner-products when compared with the CGS method, but the convergence behavior is very attractive. The splitting-up (SP) methods with the CG and BiCGSTAB algorithms (SPCG and SPBiCGSTAB) are easily parallelized on the new-type massively parallel computer: ADENART.

The SP method may be applied to other methods, which has been proposed to allow for modeling of the complex device geometry. The geometry modeling is strongly related to the grid generation problem. In MINIMOS-5 [24], the box integration method is extended to allow nonplanar interfaces in the rectangular grid. The SP method discussed in the rectangular grid is directly applied to this approach. Another approach is to use the elliptic grid generator which was first developed by Thompson [25]. The recent several works [26], [27] realize the complex device geometry using the variants of the boundary-fitted curvilinear coordinate system. It is sure that the SP method is basically effective in parallelizing the solution on finite-element grids resulting from the boundary-fitted curvilinear coordinate system. In fact, the present method can be tailored to the grid-based parallelization of the finite element method using tetrahedron elements of Friedrichs–Keller type on irregular grids [28]. Such a two-dimensional triangulation on the curvilinear coordinate system is shown in Fig. 1 as an example. Then the resulting discretization is a linear system with a regular 7-diagonal structure for the two-dimensional case and 15-diagonal structure for the three-dimensional case. Instead of the coefficient matrix with full lines, those with only essential 5-diagonal lines in two dimensions and 7-diagonal lines in three dimensions are dedicated to an incomplete factorization. In such cases, the complexity of implementation is identical to the regular implementation.

III. ADENART Computer Architecture

The ADENART is a massively parallel MIMD (Multiple-Instruction Multiple Data) computer [29], [30]. The physical system has been realized by using a three-dimensional network and two-dimensional array of processors [31]. The machine used here has 256 Processing Element (PE's) and each PE

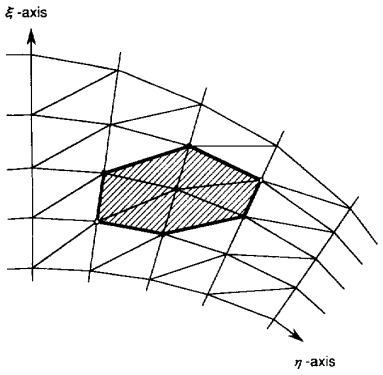


Fig. 1. Triangulation on the curvilinear coordinate system in two dimensions.

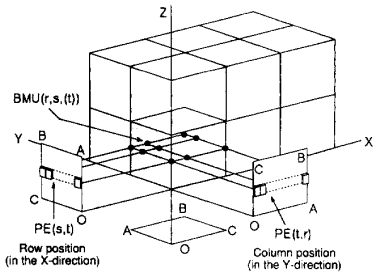


Fig. 2. An illustration for three-dimensional simulation using the ADENART.

consists of a 64-b floating point processor [32] and 2 Mbyte of local data memory. First-in-first-out Buffer Memory Unit (BMU) elements of 16 × 16 are arranged in 16 layers, respectively, which are used for the data transfer. The BMU elements are placed on all cross points of row and column. Each processor is connected to each row and column, and can access the BMU elements on such row and column. The network is designed to transfer from the data array shared among processors seeing BMU cube in a direction to those doing alternatingly in another direction. The data transfer between different situations of the processors is performed through the BMU elements:

$$\mathrm{PE}(s,t) \leftrightarrow \mathrm{BMU}(r,s,(t)) \leftrightarrow \mathrm{PE}(t,r),$$
$$\mathrm{PE}(t,r) \leftrightarrow \mathrm{BMU}(u,t,(r)) \leftrightarrow \mathrm{PE}(r,u).$$

The first expression means that there is a tth buffer memory layer $(B(r,s,(t)); r,s = 1,2,\cdots 16)$ between a pair of tth row of processors, $(\mathrm{PE}(s,t); s = 1,2,\cdots 16)$ and tth column of processors, $(\mathrm{PE}(t,r); r = 1,2,\cdots 16)$. The network is so powerful to allow the data transfer between any pair of two processors by using the transfer operation twice, as seen in the above two expressions.

An illustration for a three-dimensional simulation using the ADENART is shown in Fig. 2. The square OABC presents the two-dimensional array of 256 processors. The simulation is performed by the multiple use of the physical system. The complex of processors rotates against the fixed space axes and takes three situations: a) row position in the x-direction and column position in the y-direction; b) row position in the y-direction and column position in the z-direction; and c) row position in the z-direction and column position in the x-direction. Each PE has responsibility for serial computations in the x-, y-, and z-directions. In each computation step, the PE updates data corresponding to one of three directions and exchanges these data between PE's through BMU elements. Each PE receives the data as other directional data and computes the new directional data. All PE's repeat these operations by changing directions, in parallel.

IV. Implementation of Parallel Algorithms

The iterative process of SPCG and SPBi-CGSTAB methods involves serial computations as well as the ICCG and ILUBi-CGSTAB. The major step is a computation of backward/forward substitutions to solve the block tridiagonal systems (10a)–(10c). The ADENART computer more naturally realizes the high-level parallelism of this calculation in the SP method. In fact, the backward/forward substitutions are performed with the high-level parallelism in x-, y-, and z-directions, respectively. Since the SP method is implemented with the natural ordering on the ADENART computer, the parallel version of the iterative solution method exhibits the same convergence behavior as that on the sequential machine.

The ADENART supports parallel programming language ADETRAN, which is an expanded version of FORTRAN-77 [33]. It allows three types of data expression $x(i,j,k)$ according to three situations taken by processors as shown in Fig. 2, which are expressed as the three-dimensional array $x(i,/j,k/)$, $x(i/,j,/k)$, and $x(/i,j/,k)$. These are x-, y-, and z-direction variables in y–z, z–x, and x–y PE planes, respectively. The form shown at the bottom of the page expresses the parallel computing of the backward/forward substitutions in the y-direction (10b), where a pair of pdo and pend assigns a parallel computation paragraph. As shown in Fig. 3, the real x–z plane composed of the $L \times N$ grid nodes

```
pdo i = 1,L,k = 1,N
      do 100 j = 1,L
         q(i/,j,/k) = p(i/,j,/k) * (zj(i/,j,/k) - d(i/,j,/k) * q(i/,j - 1,/k))
      100 continue
      do 101 j = M,1,-1
         zj(i/,j,/k) = q(i/,j,/k) - p(i/,j,/k) * e(i/,j,/k) * zj(i/,j + 1,/k)
      101 continue
pend
```

is covered with several sections corresponding to the 16 × 16 PE array. The parallel command "pdo" allows the parallel computing over i and k. A serial computation is performed along j. As a result, the serial computation for one-dimensional processing is concurrently performed by 256 processors. To calculate the backward/forward substitutions in the z-direction (10c), the Alternating Direction Execution (ADE) operation is needed for the data transfer between the processors. This operation is expressed by the following form using the parallel command "pass."

```
pass i = 1,L, j = 1,M,k = 1,N
        zk(/i, j/,k) = zj(i/, j, /k)
pend
```

This means that the corresponding processors exchange the y-direction data to the z-direction data. The backward/forward substitutions in the z-direction are followed after the ADE operation. The command "pass" is executed twice to obtain the solution of $Cz = r$ and then the communication cost is incurred. In the case of 131072 grid nodes, the communication cost was 9.6% of the total execution time for solving $Cz = r$. The result indicates that the ADE operation significantly reduces the communication overhead in the massively parallel algorithm.

Moreover, the CG and BiCGSTAB algorithms involve serial computations of the matrix-vector and inner products. The parallel computation of the matrix-vector product is obtained by the same approach as the backward/forward substitutions. The parallel programming is as shown at the bottom of the page, where a, b, c, d, e, f, g are the coefficients of the matrix A. p is the search vector in the CG algorithm. The command "pass"

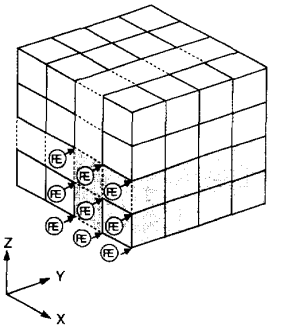


Fig. 3. Schematic explanation of parallel computing in the x-z plane. In this case, a serial computation is performed along the y-direction.

is executed twice to calculate the matrix-vector product. In this case, the one-dimensional processing and data transfer can be done concurrently using the feature of the three-dimensional network. In the case of 131072 grid nodes, the communication cost is reduced to 5.2% of the total execution time for the matrix-vector product. The straightforward implementation of inner products is also achieved using the parallel commands pdo and pass.

V. Results and Discussions

The parallel versions of the SPCG and SPBi-CGSTAB methods were implemented into a three-dimensional device simulator: SMART-II [14]. The convergence behavior and the computational performance were investigated by three-dimensional drift-diffusion device simulations without includ-

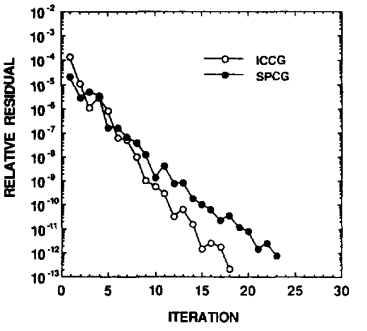
```
pdo i = 1,L, j = 1,M
    do 200 k = 1,N
      ap(/i, j/, k) = a(/i, j/,k) * p(/i, j/,k)
                +f(/i, j/,k) * p(/i, j/,k − 1) + g(/i, j/,k) * p(/i, j/,k + 1)
    200 continue
pend
pass i = 1,L, j = 1,M,k = 1,N
      ap(i/, j, /k) = ap(/i, j/,k)
pend
pdo i = 1,L,k = 1,N
    do 201 j = 1,M
      ap(i/, j, /k) = ap(i/, j, /k)
                +d(i/, j, /k) * p(i/, j − 1, /k) + e(i/, j, /k) * p(i/, j + 1, /k)
    201 continue
pend
pass i = 1,L, j = 1,M,k = 1,N
      ap(i, /j,k/) = ap(i/, j, /k)
pend
pdo j = 1,M,k = 1,N
    do 202 i = 1,L
      ap(i, /j,k/) = ap(i, /j,k/)
                +b(i, /j,k/) * p(i − 1, /j,k/) + c(i, /j,k/) * p(i + 1, /j,k/)
    202 continue
pend
```

ing an electron temperature model in SMART-II. Poisson's equation is discretized using the usual seven-point central difference approximation in the nonuniform spacing. In discretizing the continuity equations, the Scharfetter and Gummel's scheme [34] is used. The coupled equations are solved with the Gummel iterative method [35] to determine the electrostatic potential ψ, and the electron and hole densities n and p. In SMART-II, the model is mainly developed to design CMOS devices and DRAM cells [36]–[38]. In MOSFET simulations, the Gummel iterative method is effective while the Newton method requires an additional computation cost to obtain the initial guess for the Newton loop.

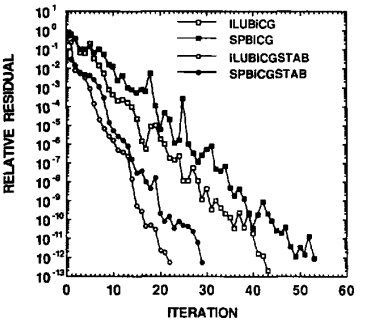
Fig. 4(a)–(c) shows relative residual norms $||Au - k||_2/||k||_2$ versus iteration for the SPCG and SPBiCGSTAB methods. The simulated device is a scaled 0.25 μm LDD n-MOSFET having 7 nm thick gate oxide and sidewall spacer of 0.15 μm. The LDD regions have a doping of 2.0×10^{18} cm^{-3} and the channel doping concentration reaches 2.0×10^{17} cm^{-3}. The mobility model used in the simulation includes the normal electric field dependence in an inversion layer [14] together with the doping and parallel electric field dependences taken from Scharfetter and Gummel's expression [34]. The bias conditions were $V_G = 2.0$ V and $V_D = 2.0$ V. The results are compared with those of ICCG and ILUBiCGSTAB. When compared with the IC and ILU decompositions, the SP method exhibits the small increase of 20–30% in the number of iterations. The method, however, provides stable convergence and high-level parallelism of the iterative solution method. Also, the number of iterations for solving the electron and hole continuity equations, which is a major part of computation time, is significantly reduced by the BiCGSTAB algorithm. The convergence rate of the BiCGSTAB with the SP method is superior to that of the ILUBiCG method.

Fig. 5(a) and (b) compares the convergence behaviors of BiCG, CGS, and BiCGSTAB with the SP method. It is found that the convergence behaviors of BiCG, CGS, and Bi-CGSTAB with the SP method are very similar to those with the incomplete LU factorization as the preconditioning. Both BiCG and CGS methods exhibit many local peaks of residual in the convergence history. Although the oscillations of residual in CGS are larger in amplitude than that in BiCG, the convergence of CGS is faster than that of BiCG. The Bi-CGSTAB with the SP method converges much more smoothly than the CGS and BiCG as well as ILUBiCGSTAB. These results were obtained from the MOSFET simulations with Gummel iterative method. The convergence property of the SP method for Newton matrices should be further investigated for bipolar simulations.

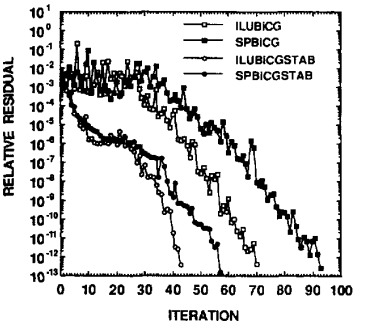
The real computational performance of the SPCG and SPBi-CGSTAB methods on the ADENART-256 is shown in Fig. 6(a) and (b). CPU time per iteration is plotted as a function of the number of grid nodes. The number of grid nodes ranges given Nx, Ny, and Nz from $64 \times 2 \times 64$–$64 \times 16 \times 64$ and $64 \times 18 \times 64$–$64 \times 32 \times 64$. The results are also compared with those calculated using the vectorized versions of the ICCG and ILUBiCGSTAB methods on a vector-type supercomputer Fujitsu VP-200 (the peak performance is 533 Mflops). The vectorized rate of both solution methods was



(a)



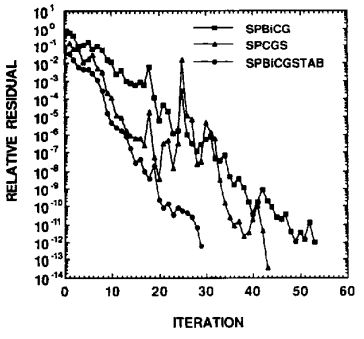
(b)



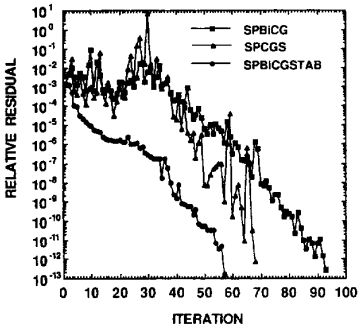
(c)

Fig. 4. Relative residual versus iteration for the SPCG and SPBiCGSTAB methods. (a) Poisson equation. (b) Electron continuity equation. (c) Hole continuity equation. The results are compared with those calculated using the ICCG and ILUBiCG methods. The simulated device structure is a scaled 0.25 μm LDD n-MOSFET.

99.3%. The computation time on the vector processor is reduced to less than 1/32 of that on the scalar processor. As shown in Fig. 6(a) and (b), the high-speed computing of the SPCG and SPBiCGSTAB is realized by using the ADENART computer. The maximum parallel efficiency is achieved when all of Nx, Ny, and Nz are multiples of 16. However, there is a jump in the CPU time for $64 \times 18 \times 64$ grid nodes. This is because some processors are idle due to the multiple use of 16 $\times$ 16 processors. In this case the load balancing is improved from that for $64 \times 2 \times 64$ grid nodes. For the vector-type supercomputer the computation time is rapidly increased with

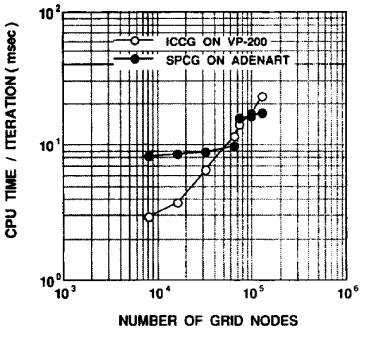


(a)

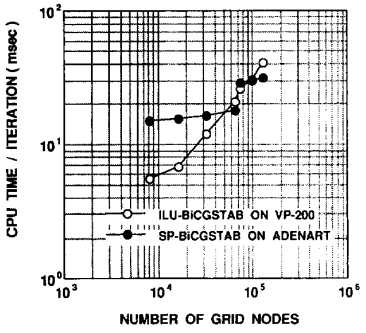


(b)

Fig. 5. Convergence behaviors of BiCG, CGS, and BiCGSTAB with the splitting-up method for solving (a) electron continuity equation and (b) hole continuity equation.



(a)



(b)

Fig. 6. CPU time per iteration of (a) SPCG and (b) SPBiCGTAB as a function of the number of grid nodes. The number of grid nodes ranges from $64 \times 2 \times 64$–$64 \times 16 \times 64$ and from $64 \times 18 \times 64$–$64 \times 32 \times 64$. The results are compared with those calculated by the vectorized versions of ICCG and ILUBiCGSTAB using Fujitsu VP-200.

increase of the number of grid nodes. It is found that in large-scale simulations of greater than 100 000 grid nodes the real computation speed of the new solution methods is superior to those on the vector-type supercomputer. The effective performance of SPCG and SPBiCGSTAB was 352 Mflops and 339 Mflops on the ADENART computer, respectively. The results indicate that the new parallel version of the iterative solution method allows the practical use of massively parallel computation for the three-dimensional MOSFET simulations.

The speedup achieved by the parallel version of the solution methods is shown in Fig. 7(a) and (b) for the fixed-sized problems of 65536 and 131072 unknowns. The dased lines show the ideal performance. The standard speedup is written as

$$S(p) = \frac{\text{Sequential CPU time}}{\text{CPU time using } p \text{ processors}}. \qquad (12)$$

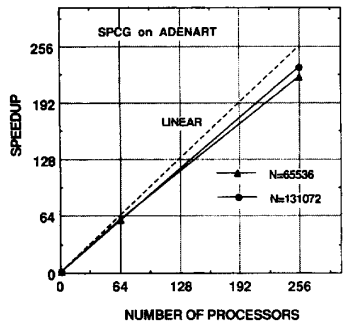
Then, the parallel efficiency is defined by $E(p) = S(p)/p$. For 256 processors, the SPCG and SPBiCGSTAB show the speedup of 222 and 223 and hence the parallel efficiencies are 86.5 and 87.2%, respectively. In the case of 131072 unknowns, the parallel efficiencies reach up to 90.9 and 90.6% for the SPCG and SPBiCGSTAB, respectively. In both methods, the high efficiency level over 90% can be achieved even when using up to 256 processors. To achieve the maximum parallel efficiency in practical applications, all of Nx, Ny, and Nz are recommended to be multiples of 16.

VI. Conclusions

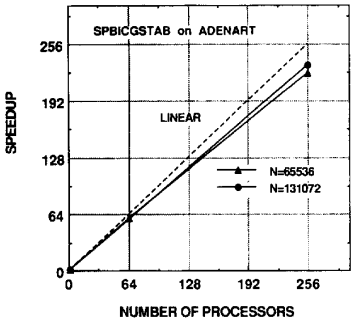
The splitting-up CG and BiCGSTAB methods have been proposed to realize a grid-based parallelization of iterative solution methods for the three-dimensional drift-diffusion device simulations. The splitting-up operator method allows the parallel computation according to the natural ordering, which is realized by the one-dimensional processing in the x-, y-, and z-directions. In the Gummel iterative method the Bi-CGSTAB with the splitting-up method converges much more smoothly than the other tested methods. The parallel versions of the splitting-up CG and BiCGSTAB methods provide a natural and high-level parallelism of iterative solution method using a new type of massively parallel computer: ADENART-256. In large-scale device simulation of greater than 100 000 grid nodes, parallel efficiency level over 90% was achieved even when using up to 256 processors. The real computational performance of the new solution method is superior to those calculated by the vectorized version of the ICCG and ILUBiCGSTAB methods on a vector-type supercomputer.

Acknowledgment

The authors would like to thank Dr. T. Takemoto, H. Ezaki, Y. Mano, Y. Terui, and Dr. H. Kadota for their encouragement through this work. The authors also wish to thank A. Hiroki and K. Zaiki for technical discussions, K. Kaneko, T.



(a)



(b)

Fig. 7. Speedups of (a) the parallel SPCG algorithm and (b) SPBiCGSTAB for the fixed sized problem of $N = 65536$ and $N = 131072$. The dotted lines represent ideal speedup.

Okamoto, A. Wakatani, Y. Obayashi and S. Tomida for the ADENART computer and parallel programming. One of the authors wish to thank Dr. S. Hashimoto for careful reading of this manuscript.

REFERENCES

[1] T. Toyabe, H. Masuda, Y. Aoki, H. Shukuri, and T. Hagiwara, "Three-dimensional device simulator CADDETH with highly convergent matrix solution algorithm," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 4, pp. 482–488, Oct. 1985.

[2] S. Odanaka, H. Umimoto, M. Wakabayashi, and H. Esaki, "SMART-P: Rigorous three-dimensional process simulator on a supercomputer," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 675–683, June 1988.

[3] J. A. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Mathematics of Computation*, vol. 31, no. 137, pp. 148–162, 1977.

[4] R. Fletcher, "Conjugate gradient methods for indefinite systems," in *Lecture Notes in Mathematics 506*. New York: Springer–Verlag, 1976, pp. 73–89.

[5] P. Sonneveld, "CGS, a fast Lanczos-type solver for nonsymmetric linear systems," Dep. of Math. and Inform., Delft Univ. of Technol., Delft, The Netherlands, Rep. 84-16, 1984.

[6] J. A. George, "Nested dissection of a regular finite element mesh," *SIAM J. Numerical Anal.*, vol. 10, no. 2, pp. 345–363, Apr. 1973.

[7] R. Lucas, "Solving planar systems of equations on distributed-memory multiprocessors," Ph.D. dissertation, Dep. of Elect. Eng., Stanford Univ., Stanford, CA, Dec. 1987.

[8] L. A. Hageman and D. M. Young, *Applied Iterative Methods*. New York: Academic, 1981.

[9] R. Guerrieri, A. Sangiovanni-Vincentelli, E. Tomacruz, T. Toyabe, and D. Webber, "Massively parallel algorithms for three-dimensional device simulation," NUPAD 1990, June 1990, pp. 35–36.

[10] K. Mayaram, P. Yang, J. Chern, R. Burch, L. Arledge, and P. Cox, "A parallel block-diagonal preconditioned conjugate-gradient solution algorithm for circuit and device simulations," in *ICCAD-90, Dig. Tech. Papers*, Nov. 1990, pp. 446–449.

[11] D. M. Webber, E. Tomacruz, R. Guerrieri, T. Toyabe, and A. Sangiovanni-Vincentolli, "A massively parallel algorithm for three-dimensional device simulation," *IEEE Trans. Computer-Aided Design*, vol. 10, no. 9, pp. 1201–1209, Sept. 1991.

[12] C. Pommerell, M. Annaratone, and W. Fichtner, "A set of new mapping and coloring heuristics for distributed-memory parallel procossors," *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 1, pp. 194–226, Jan. 1992.

[13] S. Odanaka, A. Hiroki, K. Ohe, K. Moriyama, and H. Umimoto, "SMART-II: A three-dimensional CAD model for submicrometer MOSFET's," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 619–628, May 1991.

[14] G. I. Marchuk, *Methods of Numerical Mathematics*, 2nd ed. New York: Springer–Verlag, 1982.

[15] Y. Ushiro, "ICCG method suitable for vector-type supercomputer," *Trans. Res. Inst. Math. Sci.*, pp. 110–134, 1984 (in Japanese).

[16] A. George and J. W. H. Liu, "The evolution of the minimum degree ordering algorithms," *SIAM Rev.*, vol. 31, no. 1, pp. 1–19, Mar. 1989.

[17] G. Heiser, C. Pommerell, J. Weis, and W. Fichtner, "Three-dimensional numerical semiconductor device simulation: Algorithm, architectures, results," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1218–1230, Oct. 1991.

[18] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM J. Matrix Anal. Applicat.*, vol. 11, no. 3, pp. 430–452, July 1990.

[19] D. J. Mavriplis, R. Das, J. Salts, and R. E. Vermeland, "Implementation of a parallel unstructured Euler solver on shared and distributed memory architectures," in *Proc. Int. Conf. Supercomputing*, May 1992, vol. 1, pp. 133–141.

[20] T. Nogi, "Incomplete AD decomposition," *Trans. Res. Inst. Math. Sci.*, no. 585, pp. 240–258, 1986 (in Japanese).

[21] S. Doi and N. Harada, "A preconditioning algorithm for solving non-symmetric linear systems suitable for supercomputers," in *Proc. Int. Conf. Supercomputing*, vol. 2, pp. 503–509, May 1987.

[22] H. A. Van der Vorst, "Bi-cgstab: A fast and smothly converging variant of bi-cg for the solution of nonsymmetric linear systems," *SIAM J. Sci. Statist. Comput.*, vol. 13, no. 2, pp. 631–644, Mar. 1992.

[23] M. Driessen and H. A. Van der Vorst, "BI-CGSTAB in semiconductor modeling," in *Proc. Simulation Semiconductor Devices and Processes*, Sept. 1991, pp. 45–54.

[24] M. Thurner, P. Lindorfer, and S. Selberherr, "Numerical treatment of nonrectangular field-oxide for 3-D MOSFET simulation," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1189–1197, Nov. 1990.

[25] J. F. Thompson, "Automatic numerical generation of body-fitted curvilinear coordinate system for field containing and number of arbitrary two-dimensional bodies," *J. Comp. Physics*, vol. 15, no. 3, pp. 299–319, 1974.

[26] H. Matsuo, J. Tanaka, A. Mishima, K. Tago, and T. Toyabe, "Three-dimensional device simulation with arbitrary curved boundaries using the voronoi discretization method," in *Proc. Simulation Semiconductor Devices and Processes*, Sept. 1991, pp. 157–163.

[27] Z. M.-V.-Kovacs and M. Rudan, "Boundary fitted coordinate generation for device analysis on composite and complicated geometries," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1242–1250, Oct. 1991.

[28] K. O. Friedrichs and H. B. Keller, "A finite difference scheme for generalized Neumann problems," *Numerical Solution of Partial Difference Equations*, J. H. Bramble, Ed. New York: Academic, 1966, pp. 1–19.

[29] T. Nogi, "Parallel machine ADINA," in *Computing Methods in Applied Sciences and Engineering V.*, R. Growinsky and J. Lions, Eds. Amsterdam: North-Holland, 1982, pp. 103–122.

[30] ——, "Parallel computation," *Patterns and Waves—Qualitative Analysis of Nonlinear Differential Equations*. Amsterdam: North-Holland, 1986, pp. 279–318.

[31] K. Kaneko, N. Nakajima, Y. Nakayama, J. Nishikawa, I. Okabayashi, and H. Kadota, "Processing element design for a parallel computer," *IEEE MICRO*, vol. 10, no. 2, pp. 26–38, Apr. 1990.

[32] K. Kaneko, T. Okamoto, N. Nakajima, Y. Nakakura, S. Gokita, J. Nishikawa, Y. Tanikawa, and H. Kadota, "A VLSI RISC with 20-MFLOPS Peak, 64-bit Floating-Point Unit," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1331–1340, Oct. 1989.

[33] T. Nogi, "Parallel programing language ADETRAN," *Memo. Fac. Eng. Kyoto Univ.*, vol. 51, pt. 4, pp. 235–290, 1989.

[34] D. Scharfetter and H. K. Gummel, "Large-signal analysis of a silicon Read diode oscillator," *IEEE Trans. Electron Devices*, vol. ED-16, pp. 64–77, 1969.

[35] H. K. Gummel, "A Self-consistent iterative scheme for one-dimensional steady state transistor calculations," *IEEE Trans. Electron Devices*, vol. ED-11, pp. 455–465, 1964.
[36] A. Hiroki and S. Odanaka, "Gate-oxide thickness dependence of hot-carrier-induced degradation in buried *p*-MOSFET's," *IEEE Trans. Electron Devices*, vol. 39, pp. 1223–1228, May 1992.
[37] K. Kurimoto and S. Odanaka, "A *T*-gate overlapped LDD device with high circuit performance and high reliability," in *IEDM-91 Dig. Tech. Papers*, Dec. 1991, pp. 541–543.
[38] N. Shimizu, Y. Naito, Y. Itoh, Y. Shibata, K. Hashimoto, M. Nishio, A. Asai, K. Ohe, H. Umimoto, and Y. Hirofuji, "A poly-buffer recessed LOCOS process for 256 Mbit DRAM Cells," in *IEDM-92 Dig. Tech. Papers,*, Dec. 1992, pp. 279–282.

Shinji Odanaka received the B.S., M.S., and Ph.D. degrees in applied mathematics and physics from Kyoto University, Kyoto, Japan, in 1978, 1980, and 1991, respectively.

In 1980, he joined the Semiconductor Research Center, Matsushita Electric Industrial Co., Ltd., Osaka, Japan, where he has been working on process/device modeling, simulation, and design. He has also developed a three-dimensional process/device integrated simulator "SMART," using a supercomputer. His research interests include VLSI process/device modeling, manufacturing science, and massively parallel computation for numerical simulation.

Dr. Odanaka served on the program committees of the International Electron Device Meeting from 1992–1993 and international workshops. He is a member of IEICE of Japan.

Tatsuo Nogi received the B.S., M.S., and Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1964, 1966, and 1972, respectively.

He is currently an Associate Professor of Applied Systems Science at Kyoto University. He was with Kyoto University as an Instructor from 1966, a Lecturer from 1973, and an Associate Professor from 1983. His research interests are in computational schemes and algorithms to solve PDE's, mathematical modeling for free boundary problems, and also parallel computers, softwares and algorithms. He invented parallel computer ADENA, and developed its practical compact machine ADENART through collaboration with Matsushita Electric Industrial Co., Ltd., and further continues to produce its most high-end supercomputer with preferable parallel computation environments.

Dr. Nogi has published several books on mathematics and about 30 articles. He has been a councilor of Japan SIAM since 1990.