Some experiences of solving 1-D semiconductor device equations on a Matrix coprocessor by a domain decomposition method

C.-H. Lai, H.J.J. te Riele

Department of Numerical Mathematics

Report NM-R9304 February 1993

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications. SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

P.O. Box 4079, 1009 AB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Some Experiences of Solving 1-D Semiconductor Device Equations on a Matrix Coprocessor by a Domain Decomposition Method

C.-H. Lai & H.J.J. te Riele
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract

In this report we implement a domain decomposition technique for the numerical solution of 1-D semiconductor device equations on a Cray S-MP System 500 Matrix Coprocessor with 28 processing elements. A total work expression is constructed for comparison with the actual computing time of the parallel technique. We examine the behaviour of the numerical method by using different configurations of the processing elements within the parallel machine. We perform experiments on a number of devices including p-n junctions and thyristors.

1991 Mathematics Subject Classification : 65-04, 65Y05, 65Y10
1991 CR Categories : G.1.5, J.2

Keywords & Phrases : Semiconductor devices, nonlinear two-point boundary value problems, domain decomposition, interface problems, shared memory machine, Cray S-MP.

Note : The work of the first author was supported by an ERCIM fellowship and his current address is School of Mathematics, Statistics, & Computing, University of Greenwich, Wellington Street, Woolwich, London SE18 6PF, U.K.
email : c.h.lai@greenwich.ac.uk

Report NM-R9304
ISSN 0169-0388
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. Introduction

Although the numerical simulation of 2-D semiconductor devices is well established, many of the numerical techniques used are not suited to implementation on parallel computers. Divide-and-conquer algorithms have been demonstrated most suitable for such simulations on coarse-grained parallel computers. A typical example of divide-and-conquer algorithm is domain decomposition methods. Ideally, one would require that the computational or the physical domain be split into a number of subdomains, yielding a family of independent subproblems of lower computational complexity. The solutions of these independent subproblems form the contributions to an interface problem of which the numerical solution is required. Such a technique has been employed to solve a series of 1-D semiconductor device problems on a serial machine [2]. In this report, we only deal with nonoverlapped subdomains and study the performance of such a method on a shared memory coarse-grained parallel computer, namely, the Cray S-MP System 500 Matrix Coprocessor with 28 processing elements located at CWI [4].

The organisation of this report is as follows. First, we briefly describe the mathematical model governing the electrical behaviour of semiconductor devices in an off state and obtain a nonlinear boundary value problem which describes the physical situation. Second, we describe a domain decomposition method which, when applied to the nonlinear boundary value problem, gives rise to an interface problem. This is solved numerically by means of a fixed point iteration technique. Third, we briefly describe the main features of the Cray S-MP System 500 Matrix Coprocessor and construct a total work expression and compare this with the actual computing time on the Coprocessor. We also present a number of basic computational properties for this kind of parallel architecture. Finally, we present a number of numerical examples including p-n junctions and thyristors.

2. The Mathematical Model

The set of partial differential equations which governs the electrical behaviour of 1-D semiconductor devices [1] is given by

$$\nabla^2\psi = -\frac{q}{\epsilon}(\Gamma + p - n) \tag{1}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q}\nabla\cdot\mathbf{J}_p - R \tag{2}$$

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla\cdot\mathbf{J}_n - R \tag{3}$$

Here ψ denotes the electrostatic potential, $\mathbf{J}_p$ and $\mathbf{J}_n$ the hole and electron current concentrations, respectively, R the net recombination rate, q the

electric charge, ϵ the permitivity of the device material, Γ the doping function, p the hole concentration, and n the electron concentration.

The hole and electron current concentrations are derived from the Boltzmann transport equation [1] and are given by

$$\mathbf{J}_p = -qD_p\nabla p - q\mu_p p\nabla\psi \tag{4}$$

$$\mathbf{J}_n = qD_n\nabla n - q\mu_n n\nabla\psi \tag{5}$$

where D_p and D_n are the diffusion constants for holes and electrons, respectively, and μ_p and μ_n are the mobilities for holes and electrons. We introduce the quasi-Fermi potentials ϕ_p and ϕ_n defined by

$$p = n_i e^{(\phi_p-\psi)/V_T} \tag{6}$$

$$n = n_i e^{(\psi-\phi_n)/V_T} \tag{7}$$

where V_T denotes the thermal volts ($V_T = KT/q$), K is Boltzmann's constant, T is the absolute temperature, and n_i is the intrinsic concentration of electrons in the device. Substituting into the current equations (4) and (5) yields

$$\mathbf{J}_p = -q\mu_p p\nabla\phi_p \tag{8}$$

$$\mathbf{J}_n = -q\mu_n n\nabla\phi_n \tag{9}$$

In this report, we only consider junctions in an *off state*, in which case there is no current flowing through the contacts and it is sufficient to model [1] the physics by using the nonlinear elliptic partial differential equation in (1). For computational reasons, the equations and variables are scaled to obtain dimensionless quantities. The symbols of the above unscaled variables are adopted as the symbols of the scaled variables in the subsequent scaled equations. Therefore for 1-D semiconductor devices in an off state, we require the solution of the following scaled nonlinear two-point boundary value problem :

$$\frac{d^2\psi}{dx^2} + Q(x,\psi) = 0 \qquad \in \qquad \Omega = \{x : 0 < x < w\} \tag{10}$$

subject to the Dirichlet boundary conditions $\psi(0) = V_0$ and $\psi(w) = V_w$ where V_0 and V_w are given constants. Here we have measured ψ in units of thermal volts, x in units of the Debye length ($L^2 = \epsilon KT/q^2 n_i$), $Q(x,\psi)$ in units of n_i. The source term, $Q(x,\psi)$, is given by

$$Q(x,\psi) = \Gamma(x) + e^{(\phi_p-\psi)} - e^{(\psi-\phi_n)}$$

where $\Gamma(x)$ is measured in units of n_i. Assuming Boltzmann statistics, we apply the condition $\Gamma + p + n = 0$ together with the scaled thermal equilibrium condition $pn = 1$ at the boundary [1] to obtain the scaled quasi-Fermi potential boundary values as

$$\phi_p(0) = V_0 + \ln\left[-\Gamma(0)/2 + \sqrt{(\Gamma(0)/2)^2 + 1}\right] \tag{11}$$

and

$$\phi_n(w) = V_w - \ln\left[\Gamma(w)/2 + \sqrt{(\Gamma(w)/2)^2 + 1}\right] \tag{12}$$

Since the current through the contact is zero, we have from (8) and (9) that $d\phi_p/dx = 0$ and $d\phi_n/dx = 0$ which imply ϕ_p and ϕ_n are constant. Therefore the quasi-Fermi potential boundary values given in (11) and (12) fix the values of the quasi-Fermi potentials throughout the device assuming that the device is in an off state.

3. The Numerical Scheme

3.1 The Domain Decomposition Method

We consider a nonoverlapped domain decomposition method for the nonlinear two-point boundary value problem (10). The domain Ω is split into $s+1$ nonoverlapped subdomains, Ω_k, $k = 1, 2, \cdots, s+1$, such that

$$\Omega = \{\cup_{k=1}^{s+1}\Omega_k\} \cup \{\cup_{k=1}^{s}\Gamma_k\} \tag{13}$$

where $\Omega_k = \{x | x_{k-1} < x < x_k\}$ and $\Gamma_k = \{x_k\}$. Under such decomposition of the physical/computational problem, the resulting subproblems defined in the subdomains can be completely decoupled from each other and are particularly suited to implementation in a coarse-grained parallel computational environment. Each of the subdomains has the following related nonlinear two-point boundary value subproblem,

$$\frac{d^2u_k}{dx^2} + Q(x, u_k) = 0 \quad \in \quad \Omega_k \tag{14}$$

subject to boundary conditions $u_k(x_{k-1}) = \lambda_{k-1}$ and $u_k(x_k) = \lambda_k$, and $x_0 = 0$ and $x_{s+1} = w$, $u_1(0) = V_0$ and $u_{s+1}(w) = V_w$. Let $u_k = u_k(x; \boldsymbol{\lambda})$ denote the solution of (14) where $\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_s]$. In order to obtain unique values of $\psi'(x_k)$, $k = 1, 2, \cdots, s$, we require a vector $\boldsymbol{\lambda}$ such that the following vector defect equation is satisfied,

$$\mathbf{D}(\boldsymbol{\lambda}) = [D_k(\boldsymbol{\lambda})] \equiv [\frac{\partial}{\partial x}u_k(x_k; \boldsymbol{\lambda}) - \frac{\partial}{\partial x}u_{k+1}(x_k; \boldsymbol{\lambda})] = 0 \tag{15}$$

The continuity of the function ψ across the interfaces is implicit in (14). The vector defect equation represents the reduced interface problem and guarantees the continuity of ψ' across the interfaces. In the two subdomain case, the defect equation is a scalar equation involving one interface and thus only one unknown. In the multidimensional case, the Jacobian matrix $J(\boldsymbol{\lambda}) = \mathbf{D}'(\boldsymbol{\lambda})$ is a nonsymmetric tridiagonal matrix. If $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ is a root of $\mathbf{D}(\boldsymbol{\lambda}) = 0$, then the function

$$\psi(x) = \begin{cases} \lambda^*_{k-1} & x = x_{k-1} \\ u_k(x; \boldsymbol{\lambda}^*) & x_{k-1} < x < x_k, \\ \lambda^*_k & x = x_k \end{cases} \qquad k = 1, 2, \cdots, s+1 \tag{16}$$

where $\lambda^*_0 = V_0$ and $\lambda^*_{s+1} = V_w$, is a solution of (10).

3.2 Solution of the Interface Problem

In order to solve the interface problem, i.e. the defect equation (15), without computing the matrix coefficients of the Jacobian matrix, $J(\boldsymbol{\lambda})$, a fixed point iteration technique is applied. The general fixed point iteration scheme for the solution of $\mathbf{D}(\boldsymbol{\lambda}) = 0$ is given by

$$\boldsymbol{\lambda}^{(m+1)} = \boldsymbol{\lambda}^{(m)} - \alpha_m^{-1}\mathbf{D}(\boldsymbol{\lambda}^{(m)}), \quad m = 0, 1, 2, \cdots \tag{17}$$

Here α_m is an adaptive parameter given by

$$\alpha_m = \alpha_{m-1}\frac{\|\mathbf{D}(\boldsymbol{\lambda}^{(m)}) - \mathbf{D}(\boldsymbol{\lambda}^{(m-1)})\|}{\|\mathbf{D}(\boldsymbol{\lambda}^{(m-1)})\|}, \quad m = 1, 2, \cdots \tag{18}$$

where $||.||$ denotes some norm and α_0 is chosen as the reciprocal of an arbitrary small positive real number. For comparison purpose we use a quasi-Newton scheme, i.e. by choosing a matrix update for α_m based on Schubert's scheme [3],

$$\alpha_m = \alpha_{m-1} + \frac{\mathbf{D}(\boldsymbol{\lambda}^{(m)})\mathbf{s}_{m-1}^T}{< \mathbf{s}_{m-1}, \mathbf{s}_{m-1} >}, \quad m = 1, 2, \cdots \tag{19}$$

where $< ., . >$ denotes an inner product and $\mathbf{s}_{m-1} = \boldsymbol{\lambda}^{(m)} - \boldsymbol{\lambda}^{(m-1)}$, and α_0 is chosen as a diagonal matrix such that every diagonal element is equal to the reciprocal of an arbitrary small positive real number.

3.3 Solution of the Subproblems

In order to evaluate $\mathbf{D}(\boldsymbol{\lambda}^{(m)})$, we need to solve $s+1$ subproblems each of which is defined by (14). Since the boundary value subproblem is nonlinear, we apply a Newton iteration scheme to (14) which leads to

$$\left(\frac{d^2}{dx^2} + \frac{\partial Q}{\partial u_k}\right)(u_k^{(\nu)} - u_k^{(\nu-1)}) = -\frac{d^2 u_k}{dx^2}^{(\nu-1)} - Q(x, u_k^{(\nu-1)}) \tag{20}$$

Here ν denotes the number of the Newton iteration steps for the solution of a subproblem, and clearly $\nu = \nu(m, k)$. We use a second order finite difference scheme to discretise (20) which leads to a set of tridiagonal equations to be solved. Let U_k denote the discrete approximation of u_k; we use the initial approximation $U_k^{(0)} = U_k(x; \boldsymbol{\lambda}^{(m-1)})$ in the iteration scheme (20) and iterate until $\|U_k^{(\nu)} - U_k^{(\nu-1)}\|_1 < \delta$. Our experience shows that the number of updates required to obtain a converged solution for $\boldsymbol{\lambda}$ using either (18) or (19) does not vary a lot for different values of $\delta \leq 0.01$. Furthermore ν is usually 1, except when $m = 1$, if δ is chosen as 0.01. Therefore in choosing $\delta = 0.01$, one can minimise the computational work involved in the Newton iterations.

Note that in the case $s = 0$, the original problem given in (10) is also solved by the method described in this subsection, in which case a Newton

iteration is applied to (10) instead of to (14). The notation $\nu(0,1)$ is used to denote the number of Newton iteration steps required to solve such a discretised system.

4. The Matrix Coprocessor

The Cray S-MP System 500 Matrix Coprocessor at CWI is connected to a SPARC processor. The Coprocessor is a shared memory MIMD parallel computer with an 8-Kbyte data cache in each of the processing elements. The architecture of the Coprocessor at CWI consists of seven matrix buses, each of which supporting up to a maximum of four processing elements. The Coprocessor thus has a maximum of twenty eight processing elements for parallel computation. Let B denote the number of matrix buses and P the number of processing elements per bus. The processing elements can be configured [4] in such a way that $1 \leq BP \leq 28$, for all integers $1 \leq B \leq 7$ and $1 \leq P \leq 4$. Each processing element in the Matrix Coprocessor is based on an Intel's i860 64-bit microprocessor. The processing element is a single VLSI chip that produces up to 80 MFLOPS of single-precision performance, 60 MFLOPS of double-precision performance. Each matrix bus provides a peak bandwidth of 160 MBytes per second. At a given time, only one processing element on each bus can access the 32 MBytes shared data storage area. In a 7-bus Matrix Coprocessor, up to seven processing elements can simultaneously access the shared data storage area. The total data transfer rate is thus 1.12 GBytes per second. More details of the hardware can be found in [4] and [5].

The machine described above provides a suitable environment for the domain decomposition method in such a way that each subproblem is allocated to one of the processing elements. An advantage of the machine is that each processing element has a data cache of 8 KBytes. This data cache allows a subproblem to be solved within a processing element once the necessary data has been transferred from the shared memory to the cache. Suppose N_k denotes the number of grid points in the subdomain Ω_k. From Section 3.3, we know that each subproblem involves the solution of tridiagonal linear systems which requires $O(N_k)$ floating point operations and that the data, in words, transferred from the main storage area to the cache is also $O(N_k)$. Therefore the data locality [5], which is defined as the ratio between the number of floating point operations and the words transferred, is a constant and so independent of N_k and the number of subdomains. When N_k is small, the data cache is often not fully occupied. As a result, only a small amount of computation is performed in the first processing element of a matrix bus, and it will remain idle until the second processing element of the same bus has finished communicating with the main storage area before it is allowed to communicate with the main storage area again. Therefore even though the subproblems require an equal amount of computational

work, the overall computational time will be affected more and more by the accumulation of the idle time when the number of subdomains increases. We expect that the idle time will become significant in this case. When N_k increases, the communication time between the processing element and the main storage area increases. Therefore we expect the communication time becomes significant when N_k is large.

We now construct an expression for the total work in order to compare with the actual computing time of solving nonlinear elliptic boundary value problems on the Coprocessor. Let N denote the total number of nodes in the entire computational domain and N_k the total number of nodes in the subdomain Ω_k so that

$$N = 1 + \sum_{k=1}^{s+1} (N_k - 1)$$

One work unit is defined as the computational work required to solve the sparse finite difference discretisation of an equation similar to (20) obtained by applying a Newton iteration scheme to (10) with $N-2$ unknowns on one processor. Let $\nu(0,1)$ denote the number of Newton iteration steps required to solve the nonlinear two-point boundary value problem (10) without domain decomposition. Now suppose the original problem is subdivided as given in (13) and that each subproblem is assigned to one of the processing element of the Matrix Coprocessor; then we may assume that most of the computational work comes from the solving of subproblems. Assuming that $s+1$ is equal to the number of matrix buses, and that each matrix bus contains one processing element, the total work for solving the defect equation $\mathbf{D}(\boldsymbol{\lambda}) = 0$ using the iteration scheme (17) is given by

$$\tau = \sum_{m=1}^{m_{it}} \max_{1 \le k \le s+1} \{ \frac{N_k - 2}{N - 2} \nu(m,k) \} \tag{21}$$

where $\nu(m,k)$ is the number of Newton iterations required to solve the k-th subproblem during the m-th update of the interfaces, m_{it} is the total number of updates along the interfaces. Thus, it follows that the parallel algorithm described in this section is an efficient algorithm if $\tau < \nu(0,1)$. However, this argument does not hold for more than one processing element along a matrix bus.

In view of the architecture, we expect that the computing time of a general domain decomposition algorithm strongly depends on B and P, and exhibits the following properties. The first property is due to the fact that at any time only one processing element along a matrix bus is allowed to access the shared storage area. The second property is a consequence of Property 1 for the case when the configuration satisfies $BP = s+1$. The third property is also a consequence of Property 1 for a general configuration BP. In such cases, the subdomains are divided into a number of blocks, each block consisting of BP subdomains except possibly the last one. In order for two

different configurations with a constant value of P to exhibit approximately equal computing time, the last block should be able to distribute across any of the two configurations in such a way that the maximum number of processing elements along a matrix bus is equal for both configurations.

Property 1 : If P is a constant then the computing time for solving a problem with $s+1$ subdomains on a configuration BP, in general, increases as B decreases except for the case as described in Property 3.

Property 2 : The computing time for solving a problem with $s+1$ subdomains on a configuration $BP = s+1$ is minimal by choosing B maximal (and hence P minimal).

Property 3 : Suppose that in solving a problem with $s+1$ subdomains on a configuration BP, the number of processing elements P per matrix bus is taken to be a constant. Let $b \in \mathcal{N}$ be fixed and consider the set $\mathcal{B}$ of B-values such that $\lceil \frac{s+1}{BP} \rceil = b$ [1]. Then for any two different configurations of B_1P and B_2P, with $B_1, B_2 \in \mathcal{B}$, the computing times are approximately equal if $\lceil \frac{s+1}{B_1} \rceil = \lceil \frac{s+1}{B_2} \rceil$.

Note that in Property 3, b is the number of times that BP processing elements are working in parallel on the solution of BP subproblems, but in the b-th step, there may not be enough work to keep all the processing elements busy. The maximal number r of processing elements per bus used in that step is given by $r = \lceil \frac{s+1}{B} \rceil - (b-1)P$. We perform tests to demonstrate the above Properties. Although it is possible to decrease the computing time by using Property 1, a similar property by keeping a constant value of B while decreasing P is not trivial for a general domain decomposition algorithm. We provide experiments to exploit the efficiency in this respect of the present parallel algorithm.

5. Numerical Examples

Three different doping functions taken from Ref. [2] are tested. First,

$$\Gamma_{10}(x) = \begin{cases} -N_a & 0 \le x \le w_1 \\ -N_a + \frac{(N_a+N_d)}{(w_2-w_1)}(x-w_1) & w_1 < x < w_2 \\ N_d & w_2 \le x \le w \end{cases} \tag{22}$$

where N_a is the acceptor concentration and N_d is the donor concentration. Second,

$$\Gamma_{12}(x) = -N_c e^{-m_1 x^2} + N_c e^{-m_2(w-x)^2} \tag{23}$$

where $m_1 = \frac{1}{w_a^2} \ln N_c$, $m_2 = \frac{1}{w_d^2} \ln N_c$, the width of the device is $w = w_a + w_d$,

[1] $\lceil x \rceil$ denotes the smallest integer $\ge x$.

$\Gamma(x)$	$\psi(0)$ $\times V_T$	$\psi(w)$	wL (μm)	other widths (μm)	acceptor/donor concentration $(\mu\text{m})^{-3}$
Γ_{10}^S	0	10	180	$w_1L = 30$ $w_2L = 150$	$N_an_i = N_dn_i = 1480$
Γ_{10}	0	10	180	$w_1L = 10$ $w_2L = 130$	$N_an_i = N_dn_i = 1480$
Γ_{12}	0	10	180	$w_aL = 70$ $w_dL = 110$	$N_cn_i = 1480/(1-e^{-1})$
Γ_{30}	10	0	700	$w_{ne}L = 27.5$ $w_{pb}L = 37.5$ $w_{pe}L = 125$ $w_{nb}L = 510$ $w_{off}L = 60$	$N_{ne}n_i = 10^9$ $N_{pb}n_i = 2 \times 10^7$ $N_{nb}n_i = 40$ $N_{pe}n_i = 2 \times 10^7$

Table 1: Numerical constants used in the tests.

and the acceptor and donor concentrations are equal to N_c. Third,

$$\Gamma_{30}(x) = N_{ne}e^{-m_1x^2} + N_{pb}e^{-m_2(x+w_{off})^2} + N_{nb} - N_{pe}e^{-m_3(w-x)^2} \qquad (24)$$

where

$$m_2 = \frac{1}{(w_{pb}+w_{ne}+w_{off})^2}\ln(N_{pb}/N_{nb})$$

$$m_1 = m_2\left(1+\frac{w_{off}}{w_{ne}}\right)^2 + \frac{1}{w_{ne}^2}\ln(N_{ne}/N_{pb})$$

$$m_3 = \frac{1}{w_{pe}^2}\ln(N_{pe}/N_{nb})$$

and the total width of the device is $w = w_{ne} + w_{pb} + w_{nb} + w_{pe}$. Here Γ_{10} and Γ_{12} are p-n junctions and Γ_{30} is a thyristor. In the subsequent tests, we have used silicon as the semiconductor material of which the permitivity is $\epsilon = 1.1 \times 10^{-16}\text{F}(\mu\text{m})^{-1}$. We restricted the ambient temperature as a constant at 300°K, thus we have $n_i = 1.48 \times 10^{-2}(\mu\text{m})^{-3}$. Taking $K = 1.38 \times 10^{-23}\text{J°K}^{-1}$ and $q = 1.6 \times 10^{-19}\text{C}$, we can evaluate the normalisation constants.

Table (1) shows various numerical constants used in the doping functions. Various dimensionless quantities are multiplied by their respective normalisation constants to give the corresponding physical values. All tests were performed using double precision and solved on the entire domain without decomposition as a standard set of results for comparison with the subsequent domain decomposition results. We use superscript S to denote a symmetric doping profile. Table (2) shows the total number of grid points N and its corresponding mesh size h that were used in the tests. In the case of computations performed on the entire domain, the stopping criterion is

Problem	Mesh					
Γ_{10}^S	$N =$	91	181	361	721	1441
Γ_{10}	$h =$	2.000	1.000	0.500	0.250	0.125
Γ_{12}						
Γ_{30}	$N =$	701	1401	2801	5601	
	$h =$	1.000	0.500	0.250	0.125	

Table 2: Meshes used in the tests.

$\|\Psi^{(\nu)}-\Psi^{(\nu-1)}\|_2 < \eta$, where Ψ is the discrete approximation of ψ. In the case of domain decomposition, the stopping criterion is $\|\boldsymbol{\lambda}^{(m)} - \boldsymbol{\lambda}^{(m-1)}\|_2 < \eta$. Here η is chosen as 0.5×10^{-10} to avoid getting too close to the machine accuracy of double precision arithmetic which is usually around 12 digits. The initial approximation, $\boldsymbol{\lambda}^{(0)}$, is a vector $[\lambda_1^{(0)}\lambda_2^{(0)} \cdots \lambda_s^{(0)}]^{\mathrm{T}}$ with the values of these vector components being taken as the values of $\Psi^{(0)}$ at the corresponding positions, such that $\Psi^{(0)}(c) = (\psi(0) + \psi(w))/2$ where c satisfies $\Gamma(c) = 0$ and that it is either $\Psi^{(0)}(x) = \min\{\psi(0), \psi(w)\}$ if $\Gamma(x) < 0$ or $\Psi^{(0)}(x) = \max\{\psi(0), \psi(w)\}$ if $\Gamma(x) > 0$. Timings are obtained by means of Cray S-MP built-in timing routines [5] and are recorded in seconds.

	$s+1$	N				
		91	181	361	721	1441
Number of Iterations m_{it}	2	1	1	1	1	1
	4	N.A.	6	6	6	7
	6	11	11	10	10	10
	10	17	17	16	16	14
	20	N.A.	29	30	23	22
	30	oscillate	129	108	87	69
Total Work τ	1	11	13	13	13	13
	2	4.449	4.475	4.986	4.993	4.997
	4	N.A.	3.441	3.719	3.734	3.992
	6	2.989	3.078	3.123	3.145	3.156
	10	2.247	2.374	2.437	2.469	2.385
	20	N.A.	1.654	1.847	1.558	1.579
	30	oscillate	4.134	3.799	3.231	2.776

Table 3: m_{it} and τ for Problem Γ_{10}^S.

We present results for Problem Γ_{10}^S with the subdomains being evenly distributed plus the restriction that $x = w/2$ being an interface. Table (3) shows the valuess of n_{it} and τ obtained for Problem Γ_{10}^S. Note that the first row of of the total work given in Table (3) is in fact $\nu(0,1)$ and that the

figures given there are independent of the number of processing elements. From Table (3), the present algorithm is faster than solving the original problem without domain decomposition when $B = s+1 = 2$, 4, 6, assuming one processing element per matrix bus. However, we cannot comment, without taking into account the communication costs and the configurations, on the cases when $s+1 > 7$ because the maximum number of buses available is seven. A number of tests as shown below are used to verify the Properties given in Section 4.

First, Table (4) shows the computing time obtained by varying B and keeping a constant value of P for $s+1 = 10$, 20. In general, the computing time increases as B decreases for a constant value of P, thus verify Property 1. It is observed that the computing times for $BP = 7 \times 1$, 6×1, and 5×1 when $s+1 = 10$ are similar, and also for $BP = 6 \times 3$, 5×3 when $s+1 = 20$. In the former case, the subdomains are divided into two blocks ($b = 2$) for parallel computations and we find that $r = \lceil \frac{s+1}{B} \rceil - (b-1)P = 1$ for $B = 7$, 6, and 5. Similarly in the latter case, the subdomains are divided into two blocks ($b = 2$) and we find that $r = 1$ for $B = 6$ and 5. A further decrease in B causes the subdomains being divided into more blocks, which in turn increases the computing time following Property 1. It is also observed that the computing times for $BP = 7 \times 2$, 6×2, and 5×2 when $s+1 = 10$ are similar, and that for $BP = 7 \times 4$ and 6×4 when $s+1 = 20$ these are not, but follow Property 1. In the former case we find that $r = \lceil \frac{s+1}{B} \rceil = 2$ for $B = 7$, 6, and 5 but in the latter case $r = \lceil \frac{s+1}{B} \rceil = 3$ and 4 for $B = 7$ and 6, respectively. This verifies Property 3. Finally it is found that, by removing the similar timings discussed above from the Table, the ratio of the computing times for $(B-1)P$ and BP lies in between 1.2 and 1.5, except when $B = 2$ where the ratio obtained is about 1.9. In fact the values 1.2 and 1.5 are approximately the ratios 7:6 and 3:2 and the value 1.9 is approximately the ratio 2:1.

Second, Table (5) shows the computing time obtained by taking $s+1 = BP$. Property 2 can easily be verified. The ratios of computing times follow the same pattern as that mentioned above despite P is not a constant.

Third, Table (6) shows the computing time obtained by keeping a constant value of B and varying P for $s+1 = 10$ and 20. It is obvious that the computing times are similar when $s+1 < BP$. In the cases when $s+1 \geq BP$, the computing time decreases, in general, as P decreases for a moderate number of grid points. However, as the number of grid points increases, data transfer between the data cache and the shared memory increases which increases the total computing time as demonstrated by taking $N = 1441$.

We now restrict to $P = 1$ in the tests for Problems Γ_{10}, Γ_{12}, and Γ_{30} in order to compare the pattern of the actual computing times and the pattern of the total work given in (21). In Problem Γ_{10}, the subdomains are evenly distributed without any restriction. Table (7) shows the values of m_{it}, τ,

$s+1$	$B \times P$	N				
		91	181	361	721	1441
10	7×2	0.027	0.040	0.072	0.117	0.206
	6×2	0.027	0.040	0.072	0.117	0.211
	5×2	0.028	0.041	0.074	0.120	0.217
	4×2	0.035	0.055	0.100	0.165	0.296
	3×2	0.043	0.067	0.122	0.202	0.373
	2×2	0.053	0.083	0.152	0.254	0.465
	1×2	0.097	0.154	0.287	0.485	0.899
10	7×1	0.024	0.037	0.070	0.117	0.210
	6×1	0.023	0.037	0.071	0.117	0.220
	5×1	0.024	0.038	0.071	0.120	0.220
	4×1	0.032	0.051	0.097	0.164	0.300
	3×1	0.039	0.064	0.121	0.205	0.384
	2×1	0.047	0.079	0.151	0.256	0.475
	1×1	0.088	0.150	0.289	0.495	0.922
20	7×4	N.A.	0.066	0.100	0.127	0.219
	6×4	N.A.	0.081	0.123	0.157	0.270
	5×4	N.A.	0.081	0.124	0.159	0.277
	4×4	N.A.	0.095	0.148	0.193	0.340
	3×4	N.A.	0.125	0.196	0.256	0.448
	2×4	N.A.	0.169	0.271	0.356	0.632
	1×4	N.A.	0.320	0.519	0.691	1.232
20	7×3	N.A.	0.065	0.098	0.126	0.218
	6×3	N.A.	0.077	0.121	0.157	0.273
	5×3	N.A.	0.077	0.121	0.159	0.280
	4×3	N.A.	0.093	0.146	0.192	0.339
	3×3	N.A.	0.122	0.194	0.257	0.454
	2×3	N.A.	0.166	0.267	0.355	0.632
	1×3	N.A.	0.316	0.515	0.688	1.232

Table 4: Computing time for Problem Γ_{10}^{S} keeping P constant.

$s+1$	$B \times P$	N				
		91	181	361	721	1441
1	1×1	0.030	0.071	0.142	0.285	0.570
2	2×1	0.016	0.031	0.067	0.134	0.268
	1×2	0.029	0.057	0.125	0.246	0.492
4	4×1	N.A.	0.026	0.053	0.103	0.218
	2×2	N.A.	0.038	0.074	0.143	0.343
	1×4	N.A.	0.069	0.135	0.262	0.566
6	6×1	0.016	0.026	0.047	0.090	0.175
	3×2	0.025	0.041	0.071	0.134	0.260
	2×3	0.034	0.055	0.094	0.177	0.343

Table 5: Computing time for Problem Γ_{10}^{S} taking $s+1 = BP$.

$s+1$	$B \times P$	N				
		91	181	361	721	1441
10	7×4	0.029	0.042	0.073	0.117	0.206
	7×3	0.028	0.041	0.072	0.116	0.205
	7×2	0.027	0.040	0.072	0.117	0.206
	7×1	0.024	0.037	0.070	0.117	0.210
10	5×4	0.029	0.042	0.074	0.119	0.214
	5×3	0.028	0.041	0.073	0.118	0.214
	5×2	0.028	0.041	0.074	0.120	0.217
	5×1	0.024	0.038	0.071	0.120	0.220
20	7×4	N.A.	0.066	0.100	0.127	0.219
	7×3	N.A.	0.065	0.098	0.126	0.218
	7×2	N.A.	0.061	0.096	0.127	0.224
	7×1	N.A.	0.054	0.090	0.123	0.223
20	5×4	N.A.	0.081	0.124	0.158	0.276
	5×3	N.A.	0.077	0.120	0.157	0.278
	5×2	N.A.	0.075	0.118	0.156	0.277
	5×1	N.A.	0.066	0.111	0.153	0.280

Table 6: Computing time for Problem Γ_{10}^{S} keeping B constant.

and computing time, obtained for Problem Γ_{10}^{S}. In Problem Γ_{12}, the subdomains are chosen with interface restriction. For $s+1=2$, we choose the interface at $xL = 70\mu$m, for $s+1=3$, we choose the interfaces at $xL = 50$ and 100μm, and for $s+1=4$, we choose the interfaces at $xL = 50$, 100, and 140μm. The reason for choosing these interfaces is to avoid putting any interface into a depletion layer, since a numerical divergence occurs if a depletion layer consists of an interface [2]. Table (8) shows the values of m_{it}, τ, and computing time obtained for Problem Γ_{12}. For Problem Γ_{30}, the subdomains are evenly distributed without any restriction. Table (9) shows the values of m_{it}, τ, and computing time. All Tables containing total work computed by using (21) reflect the actual computing times.

Finally, we apply a quasi-Newton scheme as given in (19) to solve the interface problems obtained by applying the domain decomposition method to Problems Γ_{12} and Γ_{30}. For Problem Γ_{12}, the interfaces are chosen as that given previously, and the results m_{it} and τ are presented in Table (10). For Problem Γ_{30}, the subdomains are distributed evenly, and the results m_{it} and τ are presented in Table (11). Unlike the results obtained for linear equations [3], the results here compare unfavorably with the adaptive α technique. In fact some of the results demonstrate that m_{it} increases when Schubert's scheme is used. Furthermore, a tridiagonal system of equations has to be solved during every update along the interface and thus increases the overhead. It should be noted that the scheme is not necessary a convergent scheme for nonlinear problems.

6. Conclusion

We have studied a nonoverlapped domain decomposition technique applied to a number of semiconductor devices on a Cray S-MP System 500 Matrix Coprocessor. A number of properties relating configurations and the number of subdomains are postulated and verified. A further property concerning the relation between the computing time and the number of matrix buses is also exploited. We have also demonstrated that the adaptive α technique is better than Schubert's scheme for nonlinear problems.

References

[1] Kurata, M., *Numerical Analysis for Semiconductor Devices.* Lexington Books, Massachusetts. (1982)

[2] Lai, C.-H. & Greenough, C., *Numerical solutions of some semiconductor devices by a domain decomposition method.* RAL Technical Report RAL-92-063, RAL. (1992)

[3] Lai, C.-H., *Comparing quasi-Newton methods for solving sparse interface problems.* CWI Technical Report NM-9303, CWI. (1993)

	$s+1$	N				
		91	181	361	721	1441
Number of Iterations m_{it}	2	7	7	7	8	9
	4	N.A.	12	11	10	11
	6	26	28	24	21	20
Total Work τ	1	11	13	13	13	13
	2	7.416	7.458	7.978	8.488	8.994
	4	N.A.	4.916	4.958	4.730	5.239
	6	5.348	5.832	5.423	4.965	4.983
Computing Time on Configuration $BP = s+1$	1×1	0.030	0.071	0.142	0.285	0.570
	2×1	0.028	0.053	0.110	0.230	0.484
	4×1	N.A.	0.038	0.072	0.135	0.286
	6×1	0.030	0.051	0.084	0.144	0.278

Table 7: m_{it}, τ, and computing time for Problem Γ_{10}.

[4] Lai, C.-H., Te Riele, H.J.J., & Ualit, H., *Parallel experiments with simple linear algebra operations on a Cray S-MP System 500 Matrix Coprocessor.* To appear as a CWI Technical Report, CWI. (1993)

[5] *System 500 Matrix Coprocessor Overview.* Hardware Documentation Number 860-0100-001, FPS Computing. (1991)

	$s+1$	N				
		91	181	361	721	1441
Number of Iterations m_{it}	2	8	8	9	8	8
	3	16	15	14	16	14
	4	36	31	28	26	21
Total Work τ	1	16	16	16	16	16
	2	12.742	12.788	14.031	13.433	13.438
	3	10.079	9.737	9.451	12.028	11.154
	4	12.944	11.771	11.306	11.624	10.259
Computing Time on Configuration $BP = s+1$	1×1	0.053	0.107	0.216	0.432	0.866
	2×1	0.046	0.089	0.191	0.362	0.722
	3×1	0.041	0.072	0.133	0.329	0.603
	4×1	0.059	0.093	0.165	0.324	0.562

Table 8: m_{it}, τ, and computing time for Problem Γ_{12}.

	$s+1$	N			
		701	1401	2801	5601
Number of Iterations m_{it}	2	31	31	30	30
	4	31	31	31	30
	5	37	36	37	36
	7	66	56	52	48
Total Work τ	1	29	29	29	29
	2	27.960	27.980	28.990	29.495
	4	13.940	13.970	14.734	14.742
	5	12.528	12.564	13.181	13.191
	7	12.888	11.522	11.262	10.988
Computing Time on Configuration $BP = s+1$	1×1	0.801	1.603	3.206	6.543
	2×1	0.740	1.467	3.025	6.145
	4×1	0.378	0.744	1.553	3.091
	5×1	0.346	0.676	1.399	2.781
	7×1	0.374	0.639	1.222	2.358

Table 9: m_{it}, τ, and computing time for Problem Γ_{30}.

	$s+1$	N				
		91	181	361	721	1441
Number of Iterations m_{it}	3	29	23	23	21	19
	4	31	24	30	21	21
Total Work τ	1	16	16	16	16	16
	3	15.775	13.268	13.437	14.246	13.374
	4	11.596	9.855	11.858	10.241	10.259

Table 10: m_{it} and τ for Problem Γ_{12} using Schubert's scheme.

	$s+1$	N			
		701	1401	2801	5601
Number of Iterations m_{it}	4	30	30	30	29
	5	Oscillate			
	7	Oscillate			
Total Work τ	1	29	29	29	29
	4	13.691	13.721	14.484	14.492
	5	Oscillate			
	7	Oscillate			

Table 11: m_{it} and τ for Problem Γ_{30} using Schubert's scheme.