Available online at www.sciencedirect.com
ScienceDirect
INTEGRATION, the VLSI journal 40 (2007) 447–460
INTEGRATION the VLSI journal
www.elsevier.com/locate/vlsi

# Substrate model extraction using finite differences and parallel multigrid

João M.S. Silva[a,b,*], L. Miguel Silveira[a,b,c]

[a]*Instituto Superior Técnico, Technical University of Lisbon, Lisboa, Portugal*
[b]*INESC ID Lisboa, Systems and Computers Engineering Institute: Research and Development, Lisboa, Portugal*
[c]*Cadence Laboratories, INESC ID Lisboa, Lisboa, Portugal*

Received 2 September 2005; received in revised form 2 May 2006; accepted 23 May 2006

**Abstract**

Substrate noise in integrated circuits is one of the most important problems in high-frequency mixed-signal designs, such as communication, biomedical and analog signal processing circuits and systems. Fast-switching digital blocks inject noise into the common substrate, hindering the performance of high-precision sensible analog circuitry. Miniaturization trends require increasing the accuracy in substrate coupling simulation environments. However, model extraction and evaluation times should not increase, which demands for fast and still accurate substrate model extraction tools.

In this work, a three-dimensional finite difference extraction methodology is presented. The resulting three-dimensional mesh is efficiently reduced to a circuit-level contact-based model by means of a fast multigrid-based algorithm. Moreover, this contact-based model extraction is shown to be efficiently computed in a parallel environment, resulting in extremely useful extraction speedups. Extraction results prove the proposed method to be very efficient, providing linear time and space complexity, and a constant number of iterations, outperforming competing algorithms.


*Keywords:* Substrate coupling model extraction; Multigrid; Parallel computing; Finite difference discretization

## 1. Introduction

Substrate electromagnetic behavior in integrated circuits presents a finite resistivity, thus causing undesired coupling between different devices to occur [1,2]. Coupling manifests itself by current migration between transistor active and channel areas, and substrate and well contact ties. For cost-saving reasons, different cells and blocks are implanted close to each other, sometimes without any kind of guard-ring or trenching isolation, causing substrate noise to influence different blocks on the same die [1,3,4].

As deep sub-micron MOS processes reach further in miniaturization, and with the increase in operating frequencies, fast-switching digital blocks inject high-frequency noise into the substrate. In purely logic circuits, where gate density is high, this means that increasingly larger levels of noise injection are present. These substrate currents can be collected through the power supply connectors and cause local voltage fluctuations that affect the delay of logic gates and the overall time performance of the circuits.

It is however in the context of mixed-signal designs that the issue of substrate coupling is most crucial. Industry trends aimed at integrating higher levels of circuit functionality have triggered a proliferation of mixed analog–digital systems. The design of such systems is an increasingly difficult task owing to the various coupling problems that result from the combined requirements for high-speed digital and high-precision analog components. Analog circuitry relies on accurate levels of currents and voltages, so that analog transistors are correctly biased and the projected performance is met. When substrate-injected currents migrate through it, substrate voltages fluctuate causing havoc in sensitive analog transistors and possibly leading to malfunctioning circuitry [1,2,5,6].

*Corresponding author. INESC ID Lisboa, Rua Alves Redol, 9, Sala 133, 1000-029 Lisboa, Portugal. Tel.: +351 213 100 260; fax: +351 213 145 843.
*E-mail addresses:* jmss@algos.inesc-id.pt (J.M.S. Silva), lms@inesc-id.pt (L.M. Silveira).

0167-9260/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2006.05.002

Analyzing the effects of substrate coupling requires a model of such couplings to be obtained and used in a verification framework. Such a verification is typically done at the electrical level by means of a circuit simulator which is given a substrate model, together with the models of the devices. Common simplifications assume that the major coupling mechanism is owed to the finite resistivity of the substrate, thus deriving resistive models. Such an approximation is valid when the dielectric relaxation time of the layers composing the substrate yields an insignificant susceptance at the frequencies of interest. Consequently, such an approximation becomes questionable beyond a few gigahertz, specially since harmonics of significant amplitude, generated by circuit nonlinearities, may fall in the range of frequencies where reactive effects are of importance [7,8].

In this paper, however, we will only focus on resistive model coupling extraction, for the sake of simplicity. Dynamic model extraction can be performed by using complex number arithmetic and is extensively analyzed in [9,8]. Herein, an efficient methodology is proposed for generating arbitrarily accurate substrate coupling models. The methodology proposed for model extraction, based on a finite difference formulation and multigrid (MG) based solution of the resulting mesh network, is detailed and several methods for system solution are compared. Moreover, the presented methodology is easily parallelizable, and can be used in a parallel computing environment, such as multiprocessor workstations, clusters or grid-computing networks, which are common in actual large design houses.

In Section 2, the mechanisms for substrate coupling are briefly discussed and we review background work in the area of substrate model extraction. The proposed model extraction algorithm is explained in Section 3 and the numerical details are discussed in Section 4. In Section 5 two distributed computing approaches are presented. Performance results are presented in Sections 6 and 7 for sequential and parallel environments, respectively. Finally, in Section 8 conclusions are drawn.

## 2. Previous work

Several substrate model extraction methodologies have been previously studied and, based on them, several extraction tools were developed. The simplest modeling methodologies consist on directly finding coupling elements based on heuristic rules. Such methods are very attractive due to the minimal extraction overhead and lead to simple first order models, which also have low simulation costs [1,6,10,11]. However, such models are generally very imprecise. Furthermore, heuristic models are only really useful to the designer, for they are unable to account for higher order effects and, in fact, rely on designer's experience to prune out the expected relevant couplings [12]. Moreover, once that is accomplished they do not provide any form of verification as to whether the performed approximation enables correct circuit simulation.

On the other hand, methodologies that avoid a priori heuristic pruning and work directly at the electrical level are typically based on a full description of the media and all the possible couplings. A problem that arises from model extraction in those cases is the extraction time and the size of the final model. There are two major classes of methods which have been proposed to generate such a model: boundary element methods (BEM) and finite difference (FD) or finite element methods (FEM).

In BEM, only the surface of the substrate contacts and diffusions[1] is discretized which leads to a system of equations that corresponds to large and dense matrices. Extraction of these models requires intensive computations, which restrains the applicability range of this methods to small- and medium-sized problems [2,5,13]. Fortunately, significant progress in BEM performance has been achieved [14,15].

In FD or FEM, the whole three-dimensional (3D) volume of the substrate is discretized leading to large but sparse matrices. FD/FEM produce even larger matrices than BEM which require extensive memory resources, although they are typically very sparse [16,17]. This type of methods has also been enhanced with fast solution techniques [18]. We should point out that it is possible to combine this type of extraction methodologies with additional information about the circuit layout or some knowledge about the circuit in terms of the noise injection and reception mechanisms. Such techniques allow for careful pruning of the number of contacts, lead to faster extraction procedures and still produce very accurate models (the interested reader should see for example the discussion in [19,20] and the references herein). Nevertheless, extraction of a substrate model that reflects the coupling effects through the substrate is still required in order to account for the propagation effect.

In our work, a fast FD-based method for the extraction of couplings between substrate contacts was developed. The large 3D mesh resulting from the discretization is reduced using a fast MG-based algorithm with linear complexity. Furthermore, we introduce the possibility of two types of parallel extraction of the coupling between the contacts, which can introduce highly useful speedups in the design flow.

## 3. Substrate model extraction

In our work, a model of the couplings between substrate contacts is extracted using a FD-based method. Moreover, the large 3D mesh resulting from this discretization is reduced using a fast MG-based algorithm. We assume that a set of contacts has been defined on the substrate, possibly consisting of active devices, connections to the power rails, backplane, etc. It is possible to use circuit layout information to generate constraints among the physical

[1] In the remaining we will use the word *contacts* to indicate for substrate and well contacts and diffusions.

contacts which will impact the definition and the number of contacts to be considered. As an example, it is possible to impose that certain substrate connections are equipotential due to the design of the interconnect structure above the substrate. In such cases, these connections should be handled as being part of a single contact. Whenever available, such information can be used to reduce the actual number of contacts without affecting the flow of the computation discussed or the accuracy of the model. In the following, we assume implicitly, that such information, if available, has been taken into account.

### 3.1. Finite difference tridimensional model

Applying the FD method implies a discretization of the substrate volume into a large number of small cuboid elements. Obviously, the finer the discretization, the more accurate is the generated model. An example of such a discretization is shown in Fig. 1.

FD discretization is able to handle any number of substrate vertical profiles, deep trenches, buried and epitaxial layers, guard-rings, and so on, as long as the mesh spacing is accurate enough. In the case of wells, assuming the coupling to and from the wells is capacitive, one has to extract the model inside the wells separately from the remaining substrate model and then connect all models through the coupling capacitances.

In order to obtain an electric model of the mesh we start with Maxwell's first law using the quasi-static approach:

$$\sigma \nabla E + \varepsilon \frac{\partial \nabla E}{\partial t} = 0. \tag{1}$$

In this equation, $E$ is the electric field, $\sigma$ the conductivity of the medium and $\varepsilon$ its permittivity. If we consider a node $i$ resulting from the FD discretization of the substrate (cf. Fig. 2), $\nabla E$ for the cuboid involving that node can be approximately calculated as

$$\nabla E \approx \frac{1}{\mathscr{V}} \sum_j E_{ij} S_{ij}, \tag{2}$$

where $S_{ij}$ is the surface common to nodes $i$ and $j$, $E_{ij}$ the electric field normal to that surface and $\mathscr{V}$ the volume of the cuboid and the summation takes into account all cuboid surfaces.

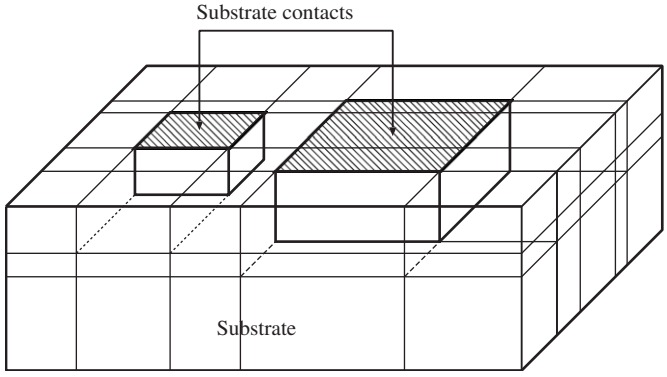


Fig. 1. Finite difference discretization.

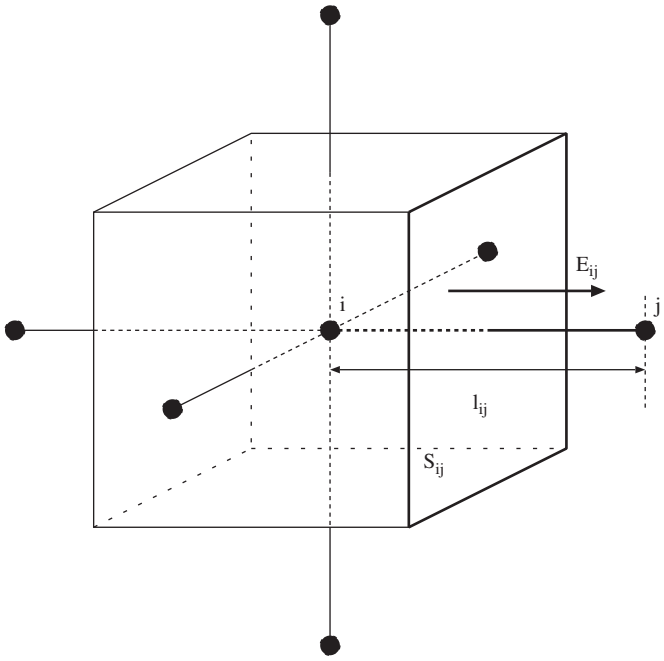


Fig. 2. Cuboid resulting from the finite difference discretization of the substrate.

Using FD, the electrical field $E_{ij}$ can also be approximated by[2]

$$E_{ij} \approx \frac{V_i - V_j}{l_{ij}}, \tag{3}$$

where $l_{ij}$ is the distance between adjacent nodes $i$ and $j$, and $V_i$ and $V_j$ the scalar potential at those nodes. Using Eq. (3) in Eq. (2) and such result in Eq. (1) we obtain

$$\sum_j \left[ G_{ij}(V_i - V_j) + C_{ij}\left(\frac{\partial V_i}{\partial t} - \frac{\partial V_j}{\partial t}\right)\right] = 0, \tag{4}$$

where $G_{ij} = \sigma S_{ij}/l_{ij}$ and $C_{ij} = \varepsilon S_{ij}/l_{ij}$.[3] Fig. 3 depicts the equivalent electrical model of Eq. (4) for each mesh node.

In order to compute the model using Eq. (4), boundary conditions are set to indicate substrate contact nodes, as well as the physical limits of the substrate. Hence, active areas (contacts, devices and possibly the backplane) are treated as Dirichlet boundaries, with constant fixed voltages, while Neumann boundary conditions (or reflective conditions) are imposed on all other physical boundaries. As a consequence, in our FD formulation, the voltage, $V_i$, at every node $i$ lying on or inside a Dirichlet boundary is known and does not need to be solved for. Furthermore, the effect of any such node lying on a Dirichlet boundary can be accounted for in its

[2]The accuracy of such an approximation increases as mesh spacing tends to zero.

[3]Eq. (4), derived from Maxwell's Laws, is in fact Kirchoff's Current Law applied to node $i$. If node $i$ had a current source connected to it, the right-hand side of Eq. (4) would equal the value of that current source.

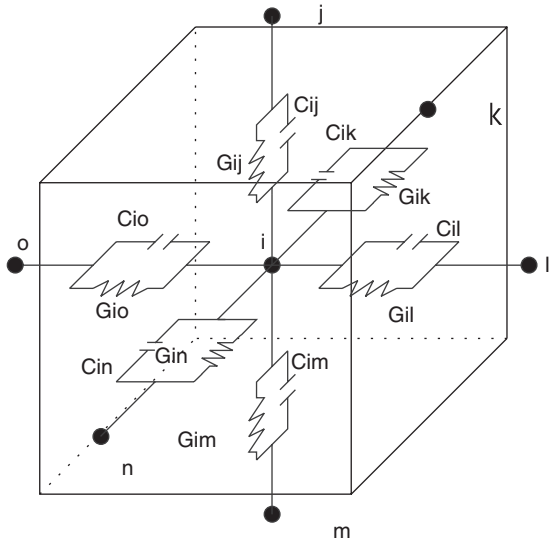


Fig. 3. Equivalent RC mesh for modeling the substrate: conductances and capacitances around a mesh node in the electrical substrate mesh.

neighbor's equation by means of a Norton equivalent. On the other hand, setting Neumann boundary conditions implies that nodes lying on the substrate's physical boundaries have fewer neighbors to account for. Therefore, the respective equations (cf. Eq. (4)) involve a smaller number of terms.[4]

For typical values of $\sigma$ and $\varepsilon$, the dielectric relaxation time of the substrate is of the order of tens of picoseconds, which is much smaller than the typical time scales of the circuit. Thus, it is reasonable to neglect intrinsic substrate capacitances for frequencies of operation up to a few gigahertz. Experimental comparisons conducted with detailed device simulators have shown that such an approximation does not affect the precision of the results for frequencies in the gigahertz range [16,17]. Hence, the general trend in the area of substrate model extraction is toward the generation of resistive coupling macro-models. Notwithstanding, mixed-mode systems with aggressive fast digital components may require more accurate modeling. In [7], a method for substrate dynamic model extraction is proposed that takes into account the intrinsic physical-level capacitance elements from Eq. (4) and produces an RC contact-based macro-model (see Section 3.2 for discussion of the equivalent resistive-only contact-level macro-model). It is shown that the extraction of such an RC model increases extraction complexity only by a constant factor. Consequently and for the sake of simplicity, in this paper we will focus only on resistive model extraction. Nevertheless, the methodology herein proposed can also be applied to RC model extraction.

[4]For nodes lying in the substrate physical boundaries, terms corresponding to neighbors that would be outside the 3D mesh are dropped. Furthermore, terms corresponding to neighbors that are also themselves in the physical boundary of the substrate are weighted appropriately to compensate for a smaller cuboid volume.

### 3.2. Circuit-level model extraction

Using the 3D mesh model from Eq. (4) directly in any electrical simulator is prohibitive due to the model's sheer size. Furthermore, the model size would be critically dependent on the spacing of the discretization used, and thus on its accuracy, which is highly undesirable. We seek a macro-model whose size is proportional to the number of substrate contacts and whose accuracy is independent of such size. Consequently, we will use the typical substrate contact-based macro-model, which is depicted in Fig. 4 for a simple three contact configuration.

Considering a system with $m$ contacts and using nodal analysis (NA), the corresponding system of equations can be written as

$$G_c U = J, \tag{5}$$

where $G_c \in \mathbb{R}^{m\times m}$ is the matrix of resistive coupling elements between the $m$ *contacts* and $U, J \in \mathbb{R}^m$ are the vectors of *contact* voltages and *contact* injected currents, respectively. $G_c \in \mathbb{R}^{m\times m}$ is the macro-model that we seek for further analysis. The procedure we use to compute $G_c$ is quite standard. Suppose we set the voltage in contact $k$ ($k = 1 \ldots m$) to 1 V and the voltage in all other contacts $i$ ($i \neq k$) to 0 V. In that case, the $k$th element of $U$ in Eq. (5) is 1 while all others are 0 and $J$ trivially equals the $k$th column of $G_c$. Thus, if given such a $U$ we compute the corresponding $J$, we recover the $k$th column of $G_c$. By repeating this procedure $m$ times, once for each contact, we construct $G_c$ one column at a time.

All that is required to compute the substrate macro-model is to, given a set of contact voltages, determine the contact-injected currents. In order to perform this computation we resort to the 3D mesh model. Applying NA to the 3D model, assuming a discretization leading to $n$ mesh nodes, originates a similar system of equations:

$$GV = I, \tag{6}$$

where $G \in \mathbb{R}^{n\times n}$ and $V, I \in \mathbb{R}^n$ are now the voltage vector for all *nodes* in the discretization mesh and the corresponding injected currents, respectively. This system is analogous to the previous one, but much larger since, in general, $n \gg m$. The two systems are nevertheless related and in fact the substrate contact model defined by $G_c$ in (5) can be

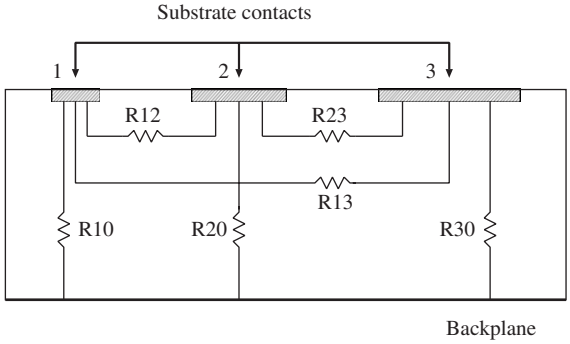


Fig. 4. Resistive model for a three contact configuration.

obtained from the 3D model in (6) by means of simple computations. Since NA is used, the inputs to Eq. (6) should be injected currents. Since one has set the voltage at contact nodes to a specified value (0 or 1 in this case), such voltages can be transformed into input currents to neighbor nodes by means of a Norton equivalent at those nodes. The complete transformation can be written as

$$I = G_{\text{adj}} U, \tag{7}$$

where $G_{\text{adj}} \in \mathbb{R}^{n \times m}$ is a matrix, that describes the Norton equivalent admittances at the mesh nodes adjacent to the nodes on the contacts. Clearly, most of the entries in $G_{\text{adj}}$ are zero, with the exception of lines related to the nodes adjacent to contacts. On the other hand, the output of the system are the currents on the contacts, which are given by

$$J = G_{\text{adj}}^{\text{T}} V \tag{8}$$

since the current injected into contacts will come from all neighboring nodes.

Therefore, combining (8), (6) and (7), we obtain

$$J = G_{\text{adj}}^{\text{T}} V = G_{\text{adj}}^{\text{T}} G^{-1} I = \underbrace{G_{\text{adj}}^{\text{T}} G^{-1} G_{\text{adj}}}_{G_c^{-1}} U, \tag{9}$$

which exposes the conductance model of the system of contacts (obviously no matrix inversion is performed; Eq. (6) is solved instead once the inputs are available).

The process of a single column extraction is therefore repeated as many times as the number of contacts, so that the full conductance matrix $G_c$ is formed, one column at a time. The cost of computing the contact model, $G_c$, for a system of $m$ contacts is thus equal to $m$ times the cost of solving the 3D mesh to determine the node voltages. This can be performed very efficiently by means of a fast MG algorithm with a cost of $\mathcal{O}(n)$ per solve as shown in Section 6. The full extraction algorithm is presented in Algorithm 1.

**Algorithm 1.** Conductance model extraction algorithm.

1. For each contact $k$:
   (a) Put nodes in contact $k$ at a specified voltage (e.g. 1 V) and all other contact nodes at 0 V;
   (b) Using Norton's equivalent, obtain the currents injected into adjacent nodes to nodes on contacts; form the conductance matrix, $G$, for the 3D system;
   (c) Solve the 3D system (6) obtaining the voltages for all the 3D mesh nodes, $V$;
   (d) Given $V$ and $G$, use Ohm's Law to compute the currents injected in all contact nodes;
   (e) Use Gauss' Law and sum injected node currents to obtain contact injected currents $J$;
   (f) By Eq. (5) and since only the nodes on contact $k$ had a fixed voltage, the $k$th column of $G_c$ equals $J$;
2. Assemble all columns into $G_c$ to form the contact-level macro-model.

### 3.3. Finite difference mesh solving methods

Several methods were considered for solving the large 3D system (6) whose solution is required to obtain the reduced circuit-level model. We propose the use of MG methods [21] and in order to determine their efficiency we compare them to Krylov-subspace methods [22], such as the conjugate gradient (CG).

Since it is well known that the performance of Krylov-subspace methods can be further improved if a good pre-conditioner is used [22], we have also implemented a preconditioned version of CG, preconditioned conjugate gradient (PCG). For this effect, the incomplete Cholesky factorization showed to yield the most efficient behavior when used as a pre-conditioner. Moreover, the MG method itself can also be used as a pre-conditioner to CG, method which we denote by multigrid preconditioned CG (MGPCG).

Besides the above-mentioned methods, other were studied which conducted to worse or similar results, like the generalized minimal residual [23] (GMRES), and will only be mentioned for benchmark reasons.

## 4. MG-based substrate model extraction

Multilevel methods rely on the availability of different accuracy levels (or grids in the case of MG [21]). Starting with an accurate 3D mesh (referred to as the $h$-mesh, where $h$ is related to the mesh spacing), the initial problem is recursively projected to coarser grids (e.g., $2h$, $4h$, etc.), until direct solution (by Gaussian elimination or some iterative method) is efficient. Solutions obtained at any coarser level are then interpolated to finer levels where local solutions are adjusted. This is called a MG V-cycle, due to its shape, and constitutes a single MG iteration. An explicit algorithm and details can be obtained from [21]. A pseudo-code description is presented in Algorithm 2.

**Algorithm 2.** Multigrid V-cycle.

1. If in the coarsest level solve $G_h \tilde{x}_h = b_h$.
2. Otherwise:
   (a) Apply $n_1$ Gauss–Seidel relaxation iterations to the system, $\tilde{x}_h = \text{GS}(G_h, b_h, n_1)$;
   (b) Compute the resulting residue, $r_h = b_h - G_h \tilde{x}_h$;
   (c) Smooth the residue vector by means of minimal residue smoothing, $r_h = \text{MRS}(r_h, \tilde{x}_h)$;
   (d) Project the residue vector to the inner level, $b_{2h} = P_h^{2h} r_h$;
   (e) Recursively compute $x_{2h} = MG(G_{2h}, b_{2h})$;
   (f) Interpolate the inner level solution, $e_h = I_{2h}^h x_{2h}$;
   (g) Adjust the current level solution, $x_h = x_h + e_h$;
   (h) Apply $n_2$ Gauss–Seidel relaxation iterations to the system, $x_h = \text{GS}(G_h, b_h, n_2)$.

Step 2c in Algorithm 2 is an optional step and not an integral part of the traditional MG V-cycle. Nonetheless,

minimal residue smoothing [24] yields very good results in reducing the number of iterations needed for MG convergence.

As said, MG methods use different accuracy grids. They operate by first decomposing the original problem into a set of sub-problems, each associated with a specific discretization grid, or level. Then, a relaxation (smoothing) scheme is applied to each sub-problem to reduce error components in that level. The sub-problems relate with one another via restriction and prolongation operators (also known as projection and interpolation operators, respectively), called inter-grid transfer operators. Since the work associated with the relaxation at each level decreases geometrically as the problem is coarsened, the total work required for going through each level once, i.e. for one MG sweep, is bounded by a small multiple of the work at the finest level. Furthermore, since the relative error reduction resulting from a relaxation iteration at each level is uniform across all levels, the error reduction for a MG sweep is equal to the error reduction at a single level. Hence, the MG convergence rate is independent of discretization which is the main reason for the efficiency of such methods in elliptic problems such as the one we are dealing with.

There are two principal algorithmic components needed for a successful MG method for solving a system $\boldsymbol{G}\boldsymbol{x} = \boldsymbol{b}$, namely the smoothing operator and the inter-grid transfer, or restriction-prolongation, operators. At every level $h, 2h, ..., 2^L h$ in the MG cycle, corresponding to a certain length scale, the error is smoothed by carefully solving a series of local problems. This first step is typically called smoothing or relaxation, and results in an intermediate guess $\tilde{\boldsymbol{x}}_h$. Next, we compute the corresponding residual $\boldsymbol{r}_h = \boldsymbol{b}_h - \boldsymbol{G}_h\tilde{\boldsymbol{x}}_h$ and project it onto the coarse grid via $\boldsymbol{b}_{2h} = \boldsymbol{P}_h^{2h}\boldsymbol{r}_h$, where $\boldsymbol{P}_h^{2h}$ is a restriction or projection operator. This procedure is repeated until the predetermined lowest level is reached, in which case we solve explicitly the coarse-grid problem and project the result onto the grid one level higher, via $\boldsymbol{e}_h = \boldsymbol{I}_{2h}^h\boldsymbol{x}_{2h}$, where $\boldsymbol{I}_{2h}^h$ is a prolongation or interpolation operator. This procedure is repeated until the highest level, corresponding to the finest-grid is reached, where the intermediate guess is updated to yield the $(k+1)$th iterate $\boldsymbol{x}_h = \boldsymbol{x}_h + \boldsymbol{e}_h$. This second stage, consisting, at each level, of a projection and an interpolation, is termed coarse-grid correction and is responsible for long-range interactions. The fine grid smoothing/coarse-grid correction cycle is repeated until the norm of the fine grid residual $\boldsymbol{r}_h$ is below some tolerance. Next, the smoothing and inter-grid transfer operators are discussed, as they play an important role in MG convergence.

### 4.1. Smoothing operator

Given $\boldsymbol{G}_h$, at level $l$, and corresponding length scale $h$, assume that one can write

$$\boldsymbol{G}_h = \boldsymbol{R}_h + \boldsymbol{S}_h, \tag{10}$$

such that $\boldsymbol{R}_h$ captures the short-range, sharply peaked portion of $\boldsymbol{G}_h$, and $\boldsymbol{S}_h$ captures the long-range, smooth portion of $\boldsymbol{G}_h$. Given (10), we define the smoothing operator as the result of solving

$$\boldsymbol{R}_h\boldsymbol{x}_h^* = -\boldsymbol{S}_h\boldsymbol{x}_h^{(k)} + \boldsymbol{b}_h \tag{11}$$

for the vector $\boldsymbol{x}_h^*$. In essence $\boldsymbol{x}_h^*$ is the "smoothed" local solution of the original system. Eq. (11) defines a *fixed-point* iteration [25], since the condition $\boldsymbol{x}_h^{(k)} = \boldsymbol{x}_h$, where $\boldsymbol{x}_h$ is the exact solution of (6), would lead to $\boldsymbol{x}_h^* = \boldsymbol{x}_h$. Since it is necessary that the above smoothing step be done cheaply, we require that $\boldsymbol{R}_h$ be easy to invert, or that $\boldsymbol{R}_h^{-1}$ has a *sparse* matrix structure, since

$$\boldsymbol{x}_h^* = \boldsymbol{R}_h^{-1}(-\boldsymbol{S}_h\boldsymbol{x}_h^{(k)} + \boldsymbol{b}_h). \tag{12}$$

In practice it is known that any relaxation method can be used as a smoothing operator and it is known that the Gauss–Seidel or SOR methods work fairly well. In matrix terms, the Gauss–Seidel method can be described as

$$\boldsymbol{x}^{(k)} = (\boldsymbol{L} + \boldsymbol{D})^{-1}(\boldsymbol{b} - \boldsymbol{U}\boldsymbol{x}^{(k-1)}), \tag{13}$$

where the splitting $\boldsymbol{G} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$ is implicitly used. The $\boldsymbol{L} + \boldsymbol{D}$ matrix corresponds to the lower triangular plus diagonal pieces of $\boldsymbol{G}$ and therefore its inversion corresponds merely to a process of forward substitution.

### 4.2. Restriction and prolongation operators

In addition to the smoothing operator, we require transfer operators $\boldsymbol{P}_h^{2h}$ and $\boldsymbol{I}_{2h}^h$ between the grids at two consecutive levels. The interpolation or prolongation operator consist in obtaining values for the fine grid points from the coarse-grid points. Any interpolation procedure can be used for this task. We have chosen the standard linear 3D interpolation which seems to provide good performance results. In this case, the interpolation matrix is a sparse, rectangular matrix and computing $\boldsymbol{e}_h = \boldsymbol{I}_{2h}^h\boldsymbol{x}_{2h}$ leads to a longer vector $\boldsymbol{e}_h$ at the finer level. The projection or restriction operator, on the other hand, restricts a vector from a fine grid onto a coarser grid. Direct projection or weighted projection can be used here. We have chosen instead to implement a Galerkin-type procedure which implies that $\boldsymbol{P}_h^{2h}$ is the matrix transpose of $\boldsymbol{I}_{2h}^h$ [21]. In this case it can be shown that

$$\boldsymbol{G_{2h}} = \boldsymbol{P}_h^{2h}\boldsymbol{G_h}\boldsymbol{I}_{2h}^h \tag{14}$$

and computing $\boldsymbol{b}_{2h} = \boldsymbol{P}_h^{2h}\boldsymbol{r}_h$ generates a smaller vector $\boldsymbol{b}_{2h}$ at the coarser level.

A pictorial depiction of the result of applying the two operators is shown in Fig. 5. It is instructive again to look at the structure of the inter-grid operators. In our case, since the restriction operator is the transpose of the prolongation operator, it is enough to look at one of them, namely the interpolator. To simplify, we consider only the 1D case depicted in Fig. 6. Clearly in this case, the fine grid points can be obtained from the coarser-grid

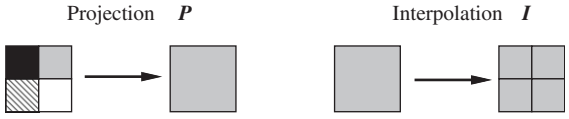


Fig. 5. Inter-grid transfers for simple domain.

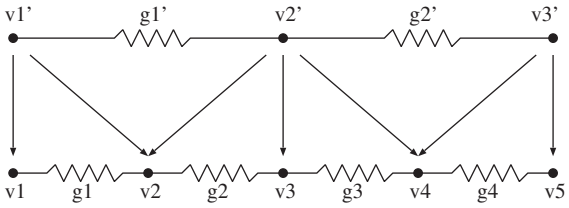


Fig. 6. Simple 1D interpolation.

nodes by means of the following operator:

$$I_{2h}^{h} = \begin{bmatrix} 1 & & \\ \frac{g_1}{g_1+g_2} & \frac{g_2}{g_1+g_2} & \\ & 1 & \\ & \frac{g_3}{g_3+g_4} & \frac{g_4}{g_3+g_4} \\ & & 1 \end{bmatrix} \tag{15}$$

For the 3D case, the situation is obviously more confusing to describe since now we have to consider interpolation in all three coordinates. However, conceptually it is the same. As one can see, the interpolation operator is again a rectangular, sparse matrix with a block-diagonal structure.

### 4.3. Coarse-level system solution

As described in step 1 of Algorithm 2, at the coarsest-level of representation, the problem is solved to a residual error. Depending on how many levels are chosen, this matrix can be very small and even a direct method can be applied. However, if the level projections are not carried out to the extreme and this matrix is of considerable size even at the coarsest-level, then a Krylov-type iterative algorithm should be used. In general, we have seen that for our problem better performance is obtained by carrying out the multilevel ideas to the extreme and solving (directly) only a very small matrix.

In Section 6 we show computational results from applying MG to our problem and comparisons to competing algorithms.

## 5. Parallel substrate model extraction

In this section we discuss the parallelization of the MG-based substrate model extraction algorithm. We start by presenting the computing network which we used to conduct our experiments. Precise definitions of what is meant by speedup and efficiency of a parallel computing environment are given. Finally, we look into Algorithms 1 and 2, and discuss how to break up the problem into the various processing units.

### 5.1. Distributed computing environment

In this work, the distributed computing environment was a cluster of machines interconnected with gigabit ethernet adapters. We used up to 16 machines of this cluster. Each machine is an Intel Pentium 4 @ 3.2 GHz with 1 GB of RAM. All the machines are running GNU/Linux.

Since all machines are equal, we avoid to some extent the discussion of relevant practical issues such as load balancing related to compute the computational capacity of each host. We consider such a discussion, while meritorious, to be outside the scope of this paper.

The network is setup using the MPICH implementation of MPI, the message passing interface standard (`http://www-unix.mcs.anl.gov/mpi`). The MPI standard is the de facto industry standard for parallel applications. It was designed by leading industry and academic researchers, and builds upon two decades of parallel programming experience. The relevant information here is that the MPI implementations provides a functional layer for communication between the various computing units. This communication is performed under the guise of message passing and the appropriate protocols are brought into the fray depending on the architecture of the distributed systems, namely using system V semaphores or spin-locks in shared memory multiprocessor machines and TCP for the networked machines.

Performance-related CPU times were measured with the function `times` from the GNU C library and consist on the sum of user and system times. The times presented in the results were taken on the machine which took the most time to run.

### 5.2. Parallel computing performance metrics

When considering the performance of a parallelized algorithm, one must compare it with a serial, single-processor implementation. To that end, if we denote by $T(p)$ the time to solve a problem on a parallel environment using $p$ processors, then we can define the following.

Parallelization *speedup* is defined as

$$S(p) = \frac{T(1)}{T(p)}. \tag{16}$$

Perfect utilization of resources is obtained when $S(p) = p$. Another relevant definition, containing similar information, is the measure of efficiency. Parallelization *efficiency* can be measured as

$$E(p) = \frac{T(1)}{p \times T(p)}. \tag{17}$$

Perfect utilization of resources is obtained when $E(p) = 1$.

To ascertain the performance of our algorithm under these metrics, and in particular to measure scalability, we assume a simplified performance model involving cost for computational tasks as well as communication. We define the *communication time* for sending $n$ `doubles` as

$$T_{\text{comm}} = \alpha + \beta n, \tag{18}$$

where $\alpha$ is the latency or startup time of the network and $\beta$ is the time to transfer a single `double`. The bandwidth of the communication channel is thus $1/\beta$. Similarly to the communication model, we define the *computation time*, measured again in terms of computation with `doubles` as

$$T_{\text{comp}} = \gamma n, \tag{19}$$

where now $1/\gamma$ is the number of flops achieved by the computing available unit.

Typically the bottleneck of most parallel implementations is the cost of communication which is usually larger than the computation time. Balancing these is one of the challenges of parallelizing an algorithm.

We will use the definitions described here to characterize the performance of the parallel implementations of our algorithm in Section 7.

### 5.3. Parallelizing MG-based model extraction

In order to exploit the available concurrency, the MG-based substrate model extraction was rewritten in order to allow execution in multiple computational units. We now look into the various computational steps of the extraction procedure described in Algorithm 1.

#### 5.3.1. Contact-level parallelism

Even a cursory examination of this procedure shows a high potential of coarse-grain parallelism. In fact, computation of the complete substrate model consists of computing each column of $\boldsymbol{G}_c$ separately. Therefore, computation of each column can clearly be performed in parallel since it corresponds to a system solution with a different right-hand side. Under this observation, parallelism can be readily exploited by committing each available processor unit to the solution of one such right-hand side. Referring to Algorithm 1, in this type of parallelism step 1 is conducted in parallel. The only nonconcurrent tasks involved in this procedure are related to putting together the results sent from the various processors, namely the data values for the various columns of the model (step 2 of Algorithm 1). Since this task is, by comparison, not very demanding computationally, the expected parallel performance will be high, meaning that the speedup achieved should be in line with the number of processors used. Of course, since we did not parallelize the setup operations of MG, this costs remains in essence unchanged. It will however get diluted in this approach for layouts with a large number of contacts (since the setup is computed only once). Exploiting this type of parallelism is perhaps not very interesting from a scientific standpoint since the parallelization is straightforward. It is nonetheless potentially of great practical relevance for cluster environments where several processors are available. We refer to this type of solution as coarse-grain parallelism.

#### 5.3.2. Full MG parallelism

More interesting is the possibility of achieving algorithmic parallelization by performing the various steps of each solve concurrently. MG-based codes have a high overhead, for setup time includes at least the computation of the projection and interpolation operators.

In our current implementation these computations are performed a priori, after setting up of the program data structures and before any solution steps commence. We have also chosen to pre-compute the projected system matrices as shown in (14). Given the special structure of the $\boldsymbol{G}$ matrix, we use a particular representation that is suited for sparse matrices, whereby we only store the nonzero matrix elements. This means that the total storage space is $\mathcal{O}(n)$. We omit the details of the representation here, since they are not crucial to the discussion at hand, nor are they essential to understanding our approach.

Each processor separately performs the task associated with the setup. It is possible to parallelize the setup tasks but we have chosen not to do it for the time being. The setup does not dominate the overall time and furthermore, it can be amortized along the various system solutions (for different contact configurations in the extraction problem). We therefore concentrate on the possibility of parallelizing the various steps in Algorithm 2.

First we consider the system solution at the coarsest level (cf. Algorithm 2, step 1). This step typically involves a very small matrix whose corresponding computations are not worth parallelizing. As such, this solution can be performed by a single processor and the result then updated where necessary. In fact, due to the fact that we chose to have all processors with copies of the various matrices at multiple levels, it makes sense to have all processors simultaneously computing this system solution, thus avoiding the need for communication. Step 2 of the algorithm contains all the remaining computations. In essence, three different types of operations are performed in this stage, corresponding to smoothing (Gauss–Seidel relaxation and MRS step), projection and interpolation. Before we discuss how to parallelize these operators we describe the partitioning of the problem. We assume that the system is partitioned in a way that each of the $p$ available processors is assigned $n/p$ system equations. If $n$ is not exactly divisible by $p$, the last processor gets the remaining equations (the resulting load unbalance is for all matters irrelevant). Now consider each operator in turn.

Given that each processor has a copy of all the projectors, interpolators and of the system matrices at all levels, tasks such as computing the piece of the residual corresponding to its own subset of equations, or updating the error for its piece of the solution, are trivially computed. Smoothing is required in order to reduce the

high-frequency error in the solution before projection to a lower level. For instance, in a serial implementation smoothing can be performed using a Gauss–Seidel-type relaxation. Unfortunately, Gauss–Seidel is not an easily parallelizable algorithm, since it requires too much communication and also enforces a strong serialization on the updates (although pipelining is achievable). In fact, initial results using this smoother yielded unimpressive results as the communication time tends to dominate from the onset. For the purpose of parallelization, it would be more adequate to perform the smoothing using a Gauss–Jacobi relaxation step. Recall that in matrix terms, the Gauss–Jacobi method can be described as

$$\boldsymbol{x}^{(k)} = \boldsymbol{D}^{-1}[\boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(k-1)}], \tag{20}$$

where the splitting $\boldsymbol{G} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$ is again used. Clearly, the inverse is trivial to perform since $\boldsymbol{D}$ is a diagonal matrix.

However, the Gauss–Jacobi relaxation is not a very effective smoother and may cause the MG algorithm not to converge. An alternative, which we pursue here, is to perform a block Gauss–Seidel variant, where each processor performs a few Gauss–Seidel relaxation steps on the equations assigned to it, without communicating with the other processors. After this, a single communication is performed to update the solution vector and a new set of Gauss–Seidel iterations is performed locally within each processor. At the end, a final update is performed and each processor computes a portion of the residue. Due to the particular structure of the system matrix and the inter-grid operators, it can easily be seen that the equations in each processor depend only on the variables contained in itself and those contained in its neighbor processors (i.e. those containing the previous and the following sets of equations). Therefore, only a partial update is required and that can be accomplished by having each processor sending its recomputed portion of the solution to its neighbors, and then receiving from them their portion of the solution. Once the residue is computed, it is also updated (now a full update is necessary due to MRS computation of inner products of vectors) and is then projected to the lower level. Once in the lower level, the same computations are applied recursively. After the lowest level is reached, and direct solution of the problem at the coarsest level is obtained, this solution needs to be interpolated to the level above. Since at the coarsest level all processors compute the solution concurrently, this solution has already been updated and interpolation can be performed without communication. Once the error is updated all corrections can be performed locally. Then, a new smoothing of the solution is performed. This time we chose to perform simple Gauss–Jacobi for it appears to be sufficient for error smoothing. Once the solution is smoothed, another partial update is performed, in order to guarantee consistency in the level above. Again, this is done locally to each processor.

A special synchronization is executed when the algorithm returns to the finest level. Here, once the new residue is computed the master processor needs to compute its norm to determine whether convergence has been achieved. To accomplish this, each processor computes its portion of the square of the norm (essentially the sum of the square of the residual components it controls) and sends it to the master. The master assembles the data, computes the norm of the residue and makes a decision on convergence.

A few more points should be noted about the implementation. We perform the updates as follows: first each processor sends its portions of the relevant data (this send is nonblocking) and then it waits until it receives the information from its neighbors (the receive operation is blocking). This places a concurrent synchronization point and ensures that no processor proceeds until its required information is indeed updated.

## 6. System solvers comparison

The presented methodology was implemented in the Substrate Model eXtractor (SMX) [26] extraction tool. SMX reads contact layout in a CIF-like format and outputs the matrix of conductance couplings, $G_c$, in a format suitable to be included in a spice-like description.

In the following subsections results are presented concerning model extraction complexity using the best suited Krylov-subspace method, CG, its preconditioned version, PCG, MG and MGPCG, all as FD mesh solvers. Results shown here concern only to extraction on a simple one contact configuration with backplane, as shown in Fig. 7, despite the tool having been tested with several other configurations, including a portion of an industrial design. Experiments were conducted on the test configuration using increasingly finer discretizations, which leads to increasingly larger matrices.

### 6.1. Iteration complexity

The first issue we look into is iteration count, which is directly related to computational complexity. As can be seen from Table 1 and Fig. 8, which shows the number of iterations and the residue norm plots, while CG and PCG

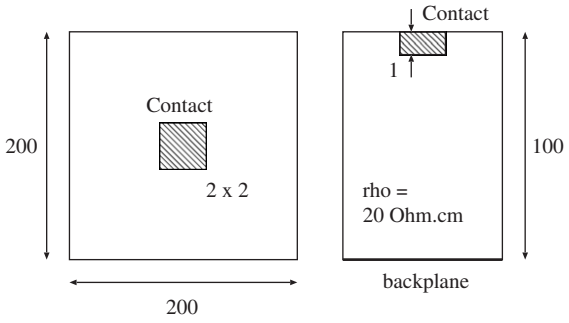


Fig. 7. Experimental layout and corresponding substrate profile (units in microns).

Table 1
Iteration count for increasing $x \times y \times z$ discretizations

| Method | $33 \times 33 \times 17$ | $65 \times 65 \times 33$ | $129 \times 129 \times 65$ |
|---|---|---|---|
| GMRES | 104 | 183 | 348 |
| CG | 132 | 189 | 290 |
| PCG | 29 | 46 | 88 |
| MG | 7 | 4 | 3 |
| MGPCG | 4 | 3 | 3 |

present rapidly degrading linear iteration complexity,[5] MG-based methods converge in a virtually constant number of iterations. This type of convergence has previously been reported for MG implementations and is the main advantage of using MG-based methods to solve the substrate coupling problem.

### 6.2. Time complexity

In spite of its superior convergence rate, MG-based methods pay some penalty due to setting up the projection and interpolation operators, as well as constructing inner level matrices. As a result, when setup times are taken into account the break-even points are likely to occur only for high levels of discretization, i.e., for high accuracy models.

As it can be seen from Table 2, MGPCG shows the best results followed closely by MG and by large surpassing nonpreconditioned Krylov-subspace methods. Both Krylov-subspace and MG methods have linear time complexity per iteration. However, the number of MG iterations is approximately constant while the number of CG/PCG iterations grows roughly with the square root of the condition number of the matrix of the system being solved. Thus, this time saving ratio will quickly increase for higher level discretizations.

### 6.3. Memory complexity

The main drawback of using MG-based methods is its memory requirements. These methods need to keep not only the projection and interpolation operators in memory, but all level matrices and vectors as well. Due to the geometric ratio of grid dimensions, the memory requirements of these structures are about the same as the memory required to solve the problem at the finer level [21]. This leads to MG-based methods with approximately the double of the memory requirements of Krylov-subspace methods, as it can be seen from Table 3.

### 6.4. Comparison conclusions

SMX incorporates MG-based methods which are very efficient, providing linear time and space complexity. Its approximately constant iteration count constitutes the advantage over using pure Krylov-subspace methods. The price to pay for this speedup is the slight increase in memory requirements, which can be overcome in a parallel implementation with distributed memory.

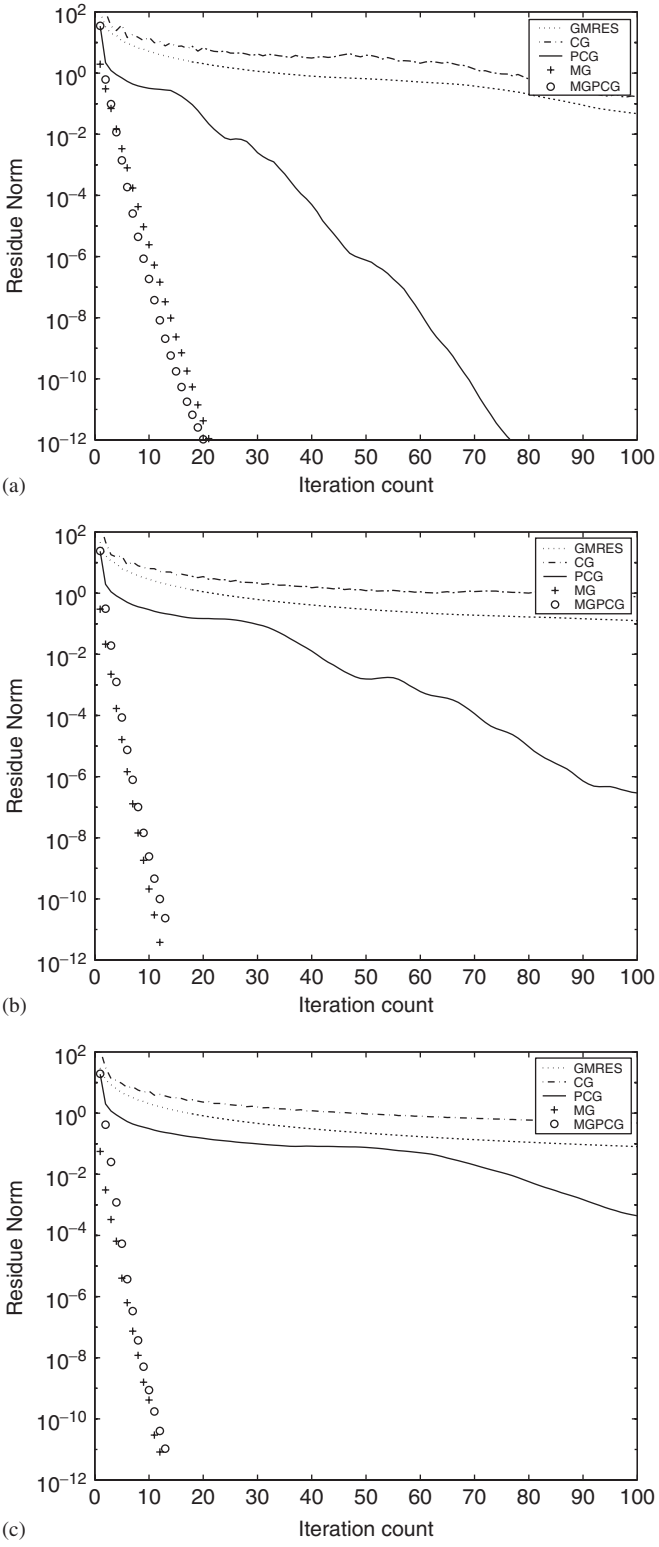


Fig. 8. Residue norm evolution for mesh discretizations of (a) $33 \times 33 \times 17$, and (b) $65 \times 65 \times 33$ and (c) $129 \times 129 \times 65$.

[5] It can be shown that the number of CG/PCG iterations grows roughly with the square root of the condition number of the matrix of the system being solved.

This methodology can thus be efficiently used in the extraction of large-scale full-chip design substrate models.

## 7. Parallelization results

In the following sections, results concerning the parallelization of the extraction of the substrate model are presented. We present the results related to the simple contact-level parallelization and the more complex full MG parallelization.

### 7.1. Coarse-grain parallelism example

For our first experiment we implemented a straightforward, coarse-grain parallelization of the extraction procedure described in Section 5.3.1. Computation of the full model is still done iteratively on the contacts, as described in Algorithm 1. However, each contact extraction is now performed separately and concurrently in one of the $p$ available processing units. Therefore, each unit solves, on average, $m/p$ contacts, where $m$ is the total number of contacts.

Table 2
Total extraction time (includes setup time) for increasing $x \times y \times z$ discretizations (in s)

| Method | $33 \times 33 \times 17$ | $65 \times 65 \times 33$ | $129 \times 129 \times 65$ |
|---|---|---|---|
| GMRES | 7.09 | 144.27 | 2125.82 |
| CG | 2.55 | 31.20 | 369.14 |
| PCG | 1.09 | 13.52 | 189.30 |
| MG | 2.65 | 18.55 | 140.20 |
| MGPCG | 2.26 | 17.09 | 136.08 |

Table 3
Memory requirements comparison (in kB)

| Method | $33 \times 33 \times 17$ | $65 \times 65 \times 33$ | $129 \times 129 \times 65$ |
|---|---|---|---|
| GMRES | 21 100 | 146 572 | 1 121 744 |
| CG | 6784 | 39 572 | 292 860 |
| PCG | 8924 | 55 776 | 419 088 |
| MG | 15 632 | 105 876 | 802 676 |
| MGPCG | 15 632 | 105 876 | 802 676 |

Table 4 shows results from this parallel implementation of the code on the distributed environment. We used $1, 2, 4, 8$ and 16 processors to extract the model of a layout with 64 contacts and show results for the elapsed times, as well as the respective parallelization efficiencies and speedups. The times shown are broken up into setup, computation and total time. We point out that the setup was not parallelized. Therefore, the efficiency of the total execution is not optimal. More interesting in this case is the efficiency of the parallelized solve time. All times are in seconds and were taken on the machine which took longest to complete for each experiment.

The results are quite encouraging with good, practical speedups being obtained even considering the total time which includes the nonparallelized setup procedure. The solve time follows a linear relationship with the number of processors as was expected, since the problems are completely independent and thus there is potentially full concurrency available.

Given the fact that the setup was not parallelized, the efficiency of the parallelization is bounded according to Amdahl's Law [27]. However, the setup time is amortized if the number of contacts is large, and the total speedup achieved can still be quite large. If we consider only the solve time, a potential source of unbalance happens when the number of contacts is not divisible by the number of available processing units. This was not the case in our setup and in practice this issue should become irrelevant when larger real industry problems are being solved. The most relevant factor in this case is the fact that due to inherent asymmetry in the layout and discretization features, some processors are given harder contacts to solve than others, leading to potential load unbalance. If necessary, these unbalances and consequent degradation of parallelization can be dealt with an appropriate load balancing procedure. We do not pursue this matter here as it is clearly outside the scope of this article.

### 7.2. Fine-grain parallelism

In this section we focus on the parallelization of the solution of a single contact. The MG system solver is parallelized according to the description in Section 5.3.2.

Table 4
Results from the coarse-grain parallelization of substrate model extraction (time in s)

| # of CPUs | Time | | | Computation | | Total | |
|---|---|---|---|---|---|---|---|
| | Setup | Computation | Total | Efficiency | Speedup | Efficiency | Speedup |
| 1 | 24.36 | 3745.46 | 3769.82 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 24.19 | 1879.56 | 1903.75 | 1.00 | 1.99 | 0.99 | 1.98 |
| 4 | 24.38 | 938.34 | 962.72 | 1.00 | 3.99 | 0.98 | 3.92 |
| 8 | 24.60 | 509.97 | 534.57 | 0.92 | 7.34 | 0.88 | 7.05 |
| 16 | 24.34 | 254.19 | 278.53 | 0.92 | 14.73 | 0.85 | 13.53 |

#### 7.2.1. Communication and computation costs

We start by estimating the costs of computation and communication according to the models described in Section 5.2. To that end we ran a standard Flops estimator (`ftp://ftp.nosc.mil/pub/aburto/flops`) and determined that the computers in the network had a performance of about 255 Mflops. This number appears to be somewhat smaller than the manufacturer's specifications, but we trust it since we had complete control over the measurement process. Going back to the model of Section 5.2 this means that $\gamma \approx 3.92$ ns. We also computed the broadcast communication times for messages of varying sizes and determined that we have a latency $\alpha \approx 0.05$ s and a bandwidth such that $\beta \approx 0.39$ μs. These results indicate that there is a high latency in the network, and therefore communication should be minimized, as expected. A more careful examination also tells us that one can perform around 13 Mflops in the same time that it takes to communicate a single `double` but just around 113 Mflops in the same time it takes to communicate a vector of $10^6$ `doubles`, as the latency time starts getting amortized.

#### 7.2.2. MG system solution parallelization

In Table 5, we show results from applying the parallel MG solver to partial model extraction (single solve, i.e. single column) of a practical example. Speedup and efficiency are computed versus the best algorithm running in a single processor (e.g. Gauss–Seidel is used as a smoother). The results are mildly encouraging. Clearly, some speedup is obtained which is of practical relevance. However, even though the speedup from the computational parts of the algorithm (the solve piece) are acceptable, the added communication time reduces that advantage. There are several reasons for the performance shown here. It is known that the MG algorithm applied to this problem is fairly sensitive to the smoothing operator used and may not even converge unless an appropriate smoother is used. The parallel implementation, as discussed in Section 5.3.2, uses a block-Jacobi smoother where Gauss–Seidel is used inside each block in a serialized fashion but the global iteration is Jacobi-like. This smoother degrades convergence somewhat, which leads to longer run times. Note that single processor extraction uses the usual Gauss–Seidel smoother and not block-Jacobi's.

The presented results show that the efficiency of the parallel version of MG is acceptable given the distributed computing environment being used. As expected with broadcast communication, used in vector updates, the efficiency drops off as the number of processors gets large. Notwithstanding, further parallelization efficiency can probably be obtained if some restrictions in the geometric partition of the problem are attended, which although depending strongly on the problem being solved, will be subject of future research. Still the results clearly show that this type of parallelization will not pay off for a large number of hosts since for finer granularity the number of computations performed in each processor is not enough to outweigh the resulting communication.

## 8. Conclusions

A methodology for the extraction of substrate coupling models has been presented. Substrate contact-level models were obtained from a formulation based on a finite difference discretization and computed using fast multigrid algorithms. Comparison tests revealed that using multigrid-based methods offers a constant iteration count, which translates in linear time and space complexities. In the substrate problem, this leads to faster model extraction times, and enables extraction of large and accurate models resulting from fine discretizations.

Moreover, the multigrid method used to solve the large 3D system inherent to the finite difference discretization has been parallelized in a distributed environment. Two types of distributed computing paradigms have been implemented: a coarse-grain contact-level parallelization and a fine-grain full multigrid parallelization. The former one shows approximately perfect utilization of computing resources while, on the other hand, fine-grain multigrid parallelization is still too much penalized by communication. This problem can be smoothed by using the optimal geometric partition of the problem which minimizes communication. Moreover, using a load balancing scheme will be of importance if a heterogeneous computing environment like grid networks is used.

Table 5
Results from the parallel MG system solver on the distributed environment (time in s)

| # of CPUs | Solve | | | Communication time | Total | |
|---|---|---|---|---|---|---|
| | Time | Efficiency | Speedup | | Efficiency | Speedup |
| 1 | 19.21 | 1.00 | 1.00 | 0.00 | 1.00 | 1.00 |
| 2 | 10.95 | 0.88 | 1.75 | 1.13 | 0.80 | 1.59 |
| 4 | 6.14 | 0.78 | 3.13 | 2.60 | 0.55 | 2.20 |
| 8 | 3.86 | 0.62 | 4.98 | 2.91 | 0.35 | 2.84 |
| 12 | 3.53 | 0.45 | 5.44 | 3.38 | 0.23 | 2.78 |
| 16 | 2.95 | 0.41 | 6.51 | 3.16 | 0.20 | 3.14 |

Execution on single processor uses best algorithm (e.g. GS is the MG smoother).

Even though coarse-grain parallelization provides better results in comparison with the multigrid parallelization, these two approaches can be combined. In terms of memory resources, multigrid parallelization makes it possible to distribute the memory requirements by the available machines, which can be extremely useful when dealing with large industrial portions of layout. Multigrid parallelization will be the subject of future research in generic applications in circuit simulation.

## Acknowledgment

This work was partly supported by the Portuguese Foundation for Science and Technology, FCT, under the grant SFRH/BD/10586/2002.

## References

[1] D.K. Su, M.J. Loinaz, S. Masui, B.A. Wooley, Experimental results and modeling techniques for substrate noise in mixed-signal integrated circuits, IEEE J. Solid-State Circuits 28 (4) (1993) 420–430.
[2] R. Gharpurey, R.G. Meyer, Modeling and analysis of substrate coupling in integrated circuits, IEEE J. Solid-State Circuits 31 (3) (1996) 344–353.
[3] R. Gharpurey, Modeling and analysis of substrate coupling in integrated circuits, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA, June 1995.
[4] N. Verghese, Extraction and simulation techniques for substrate-coupled noise in mixed-signal integrated circuits, Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, August 1995.
[5] N.K. Verghese, D.J. Allstot, M.A. Wolfe, Verification techniques for substrate coupling and their application to mixed-signal IC design, IEEE J. Solid-State Circuits 31 (3) (1996) 354–365.
[6] B. Nauta, G. Hoogzaad, How to deal with substrate noise in analog CMOS circuits, in: European Conference on Circuit Theory and Design, vol. 12, Budapest, Hungary, 1997, pp. 1–6.
[7] J.M.S. Silva, L.M. Silveira, Dynamic models for substrate coupling in mixed-mode systems, in: VLSI-SOC'2003 XII IFIP International Conference on VLSI, Darmstadt, Germany, 2003, pp. 25–30.
[8] T.-H. Chen, C. Luk, C.C.-P. Chen, SuPREME: substrate and power-delivery reluctance-enhanced macromodel evaluation, in: International Conference on Computer Aided-Design, San Jose, CA, 2003, pp. 786–792.
[9] A. van Genderen, N. van der Meijs, E. Schrik, Modeling capacitive coupling effects via the substrate, in: ProRISC IEEE 12th Annual Workshop on Circuits, Systems and Signal Processing, Veldhoven, The Netherlands, 2001, pp. 366–370.
[10] A.J. van Genderen, N.P. van der Meijs, T. Smedes, Fast computation of substrate resistances in large circuit, in: European Design and Test Conference, Paris, 1996, pp. 560–565.
[11] S. Mitra, R.A. Rutenbar, L.R. Carley, D.J. Allstot, A methodology for rapid estimation of substrate-coupled switching noise, in: IEEE 1995 Custom Integrated Circuits Conference, 1995, pp. 129–132.
[12] J.R. Phillips, L.M. Silveira, Simulation approaches for strongly coupled interconnect systems, in: International Conference on Computer Aided-Design, 2001, pp. 430–437.
[13] T. Smedes, N.P. van der Meijs, A.J. van Genderen, Extraction of circuit models for substrate cross-talk, in: International Conference on Computer Aided-Design, San Jose, CA, 1995, pp. 199–206.
[14] J.P. Costa, M. Chou, L.M. Silveira, Efficient techniques for accurate modeling and simulation of substrate coupling in mixed-signal IC's, in: DATE'98—Design, Automation and Test in Europe, Exhibition and Conference, Paris, France, 1998, pp. 892–898.
[15] M. Chou, J. White, Multilevel integral equation methods for the extraction of substrate coupling parameters in mixed-signal ic's, in: 35th ACM/IEEE Design Automation Conference, 1998, pp. 20–25.
[16] F.J.R. Clement, E.Z.M. Kayal, M. Declercq, Layin: toward a global solution for parasitic coupling modeling and visualization, in: Proceedings of the IEEE Custom Integrated Circuit Conference, 1994, pp. 537–540.
[17] B. Stanisic, N.K. Verghese, R.A. Rutenbar, L.R. Carley, D.J. Allstot, Addressing substrate coupling in mixed-mode IC's: simulation and power distribution systems, IEEE J. Solid-State Circuits 29 (3) (1994) 226–237.
[18] J. Kanapka, J. Phillips, J. White, Fast methods for extraction and sparsification of substrate coupling, in: Proceedings of the 37th Design Automation Conference, 2000.
[19] G.V. der Plas, M. Badaroglu, G. Vandersteen, P. Dobrovolny, P. Wambacq, S. Donnay, G.G.E. Gielen, H.D. Man, High-level simulation of substrate noise in high-ohmic substrates with interconnect and supply effects, in: 41st ACM/IEEE Design Automation Conference, San Diego, CA, USA, 2004, pp. 854–859.
[20] M. van Heijningen, M. Badaroglu, S. Donnay, G.G.E. Gielen, H.D. Man, Substrate noise generation in complex digital systems: efficient modeling and simulation methodology and experimental verification, IEEE J. Solid-State Circuits 37 (8) (2002) 1065–1072.
[21] W.L. Briggs, A Multigrid Tutorial, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.
[22] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Co., Massachusetts, 1996.
[23] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[24] J. Zhang, Multi-level minimal residual smoothing: a family of general purpose multigrid acceleration techniques, J. Comput. Appl. Math. 100 (1998) 41–51.
[25] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, first ed., Springer, New York, NY, 1980.
[26] J.M.S. Silva, Modeling substrate coupling in mixed analog-digital circuits (in portuguese), Master's Thesis, Instituto Superior Técnico, Technical University of Lisbon, Lisboa, Portugal, May 2003.
[27] G.E. Keiser, Local Area Networks, McGraw-Hill, Inc., New York, NY, USA, 1989.

**João M.S. Silva** was born in Lisbon, Portugal. He received the Engineer's and Master's degrees in Electrical and Computer Engineering in 2000 and 2003, from *Instituto Superior Técnico* (IST) from the Technical University of Lisbon. He is currently a researcher at the Systems and Computers Engineering Institute R&D (INESC ID). His research interests are in the area of CAD algorithms for Electronics, physical design of VLSI circuits and computer architectures. He is currently working on simulation of power grids in integrated circuits. João Silva is also a member of the IEEE.

**L. Miguel Silveira** (S'85-M'95-SM'00) was born in Lisbon, Portugal. He received the Engineer's (summa cum laude) and Master's degrees in Electrical and Computer Engineering in 1986 and 1989, from *Instituto Superior Técnico* (IST) from the Technical University of Lisbon, and the M.S., E.E. and Ph.D. degrees in 1990, 1991 and 1994 from the Massachusetts Institute of Technology, Cambridge, MA. He is currently a full Professor of ECE at IST, a senior researcher

at the Systems and Computers Engineering Institute R&D (INESC ID), and a founding member of the Lisbon Center of the Cadence Laboratories. His research interests are in various aspects of computer-aided design of integrated circuits with emphasis on interconnect modeling and simulation, parallel computer algorithms and the theoretical and practical issues concerning numerical simulation methods for circuit design problems. Mr. Silveira is a Senior Member of the IEEE and a member of Sigma Xi.