Multigrid-like Technique for Power Grid Analysis

Joseph N. Kozhaya
University of Illinois, Urbana

Sani R. Nassif
IBM Austin Research Labs

Farid N. Najm
University of Toronto

Abstract

Modern sub-micron VLSI designs include huge power grids that are required to distribute large amounts of current, at increasingly lower voltages. The resulting voltage drop on the grid reduces noise margin and increases gate delay, resulting in a serious performance impact. Checking the integrity of the supply voltage using traditional circuit simulation is not practical, for reasons of time and memory complexity. We propose a novel multigrid-like technique for the analysis of power grids. The grid is reduced to a coarser structure, and the solution is mapped back to the original grid. Experimental results show that the proposed method is very *efficient* as well as suitable for both DC and transient analysis of power grids.

1 Introduction

In recent years, there has been an increased demand for *high performance* and *low power* VLSI designs. High performance is achieved by technology scaling, increased functionality and competitive designs. On the other hand, a common technique used to obtain low power designs is to scale down the supply voltage.

Increased chip functionality results in the need for *huge* power and ground distribution networks, here referred to simply as *power grids* since they typically have a grid structure. Lower supply voltages, on the other hand, make the voltage variation across the power grids very critical since it may lead to chip failures. Voltage drop on the power grid can reduce the supply voltage at logic gates and transistor cells to less than expected. This leads to reduced noise margins, higher logic gate delays, and overall slower circuits. Consequently, once voltage drop exceeds certain designer-specified thresholds, there is no guarantee that the circuit will operate properly [1, 2, 3, 4].

Thus, it is clear that in modern VLSI circuits, power grids are becoming performance limiting factors. Consequently, *efficient* analysis of the power grids [3, 4] of modern sub-micron VLSI designs is a must.

The first step in power grid analysis involves modeling the power grids [3, 5]. For purposes of our analysis, power grids are modeled as *linear* RC networks since on-chip inductance (in the power grid) in today's technology is too small to affect the analysis results. Power sources are modeled as simple constant voltage sources and power drains are modeled as independent time-varying current sources [3, 4, 5].

The behavior of such a system can be expressed following the modified nodal analysis (MNA) [6] formulation as the following ordinary differential equation:

$$Gx + C\dot{x} = u(t) \tag{1}$$

where x is a vector of node voltages, and source and inductor currents; G is the conductance matrix; C includes the capacitance and inductance terms, and $u(t)$ includes the contributions from the sources and the drains.

Applying Backward Euler (BE) numerical integration to (1) results in a set of linear equations:

$$(G + C/h)x(t+h) = u(t+h) + x(t)C/h \tag{2}$$

which can be readily simplified to $Ax(t+h) = b$, where $A = G + C/h$ and $b = u(t+h) + x(t)C/h$. It can be shown that the system of linear equations can be reformulated in such a manner to produce a system matrix, A, which can be shown to be a *nonsingular* $\mathcal{M}$-matrix [7].

Due to the large size of typical power grids, general circuit simulators such as SPICE are not adequate for power grid analysis because of CPU time and memory limitation. The inefficiency of standard simulators comes about because (a) they require a lumped element approximation of the circuit which requires the translation of a regular geometrical structure to a large set of equivalent circuit elements, and (b) they use general purpose solution methods meant to be robust in the face of stiff systems of equations. By contrast, power grids are well behaved spatially (nearly regular) and temporally (damped). This motivates a special-purpose simulator for power grids which can make use of these properties.

In this paper, we propose an *efficient* analysis technique that follows the lines of thought of multigrid methods which are commonly used for the solution of smooth partial differential equations (PDEs). Specifically, our method is inspired by the algebraic multigrid method (AMG) which is one variant of the multigrid approach. Thus, section 2 describes the multigrid approach with a specific emphasis on the algebraic multigrid variation. After discussing the multigrid technique, we present our proposed multigrid-like approach in section 3. The efficiency of our proposed technique is verified by the experimental results given in section 4. Finally, conclusions are provided in section 5.

2 Multigrid Method

Well-designed power grids are characterized by voltage distributions which are spatially *smooth* [3]. Furthermore, the analysis of power grids results in a system of linear equations structurally identical to that of a finite element discretization of a two-dimensional partial differential equation (PDE). Consequently, *efficient* methods for solving *smooth* PDEs are worth considering as potential competitive solvers for the power grid problem.

The multigrid method, MG, is an efficient technique widely used for solving smooth PDEs [8, 9, 10]. Initial interest in multigrid resulted from a detailed analysis of classical iterative methods and the reasons for their slow convergence. Let $e = x - \hat{x}$ be the error defined as the difference between the exact solution x and the approximate solution, $\hat{x}$. It can be shown that the error can be expressed as a linear combination of *low frequency* and *high frequency* Fourier modes [8]. Furthermore, the analysis of classical iterative methods leads to the following observation [8, 11]: *Classical iterative methods efficiently reduce the high frequency error components but are inefficient in reducing the low frequency error components.*

In order to avoid the limitations of classical methods, multigrid methods consist of two complementary components [8, 10]:

1. *Relaxation* (smoothing) which reduces the high frequency error components.

2. *Coarse grid correction* which reduces the low frequency error components.

Relaxation involves running a few iterations of a classical iterative solver. Coarse grid correction involves mapping the problem to a coarser grid, solving the mapped problem, and mapping the solution back to the fine grid. The mapping between the two grids (fine and coarse) requires the use of *intergrid transfer operators* which are also referred to as the restriction and prolongation operators. The restriction operator, $\mathcal{R}_h^{2h}$, is used to map the problem from the original fine grid, Ω^h, to the coarse grid, Ω^{2h}, $b^{2h} = \mathcal{R}_h^{2h} b^h$. On the other hand, the prolongation (also referred to as interpolation) operator, $\mathcal{P}_{2h}^h$, is used to map the solution back from the coarse grid Ω^{2h} to the original fine grid, Ω^h, $x^h = \mathcal{P}_{2h}^h x^{2h}$. One intuitive motivation for coarse grid correction is that the solution at a coarse grid typically provides a good initial guess for the iterative solver at the fine grid and thus results in rapid convergence. Another motivation for this approach is that the low frequency error components at the fine grid Ω^h appear as high frequency at the coarse grid Ω^{2h} [8]. Then, relaxation at the coarser grid reduces those components.

It is clear by now how the multigrid technique works [8]. Starting with a fine grid, a few relaxation steps (iterations) are applied to reduce the high frequency modes of the error. Then the low frequency (smooth) modes of the error are reduced by coarse grid correction.

2.1 Algebraic Multigrid

Two common variations of the multigrid method are: standard multigrid, SMG, and algebraic multigrid, AMG. Both involve relaxation and coarse grid correction. The efficiency of either method relies mostly on the choice of the multigrid components: the relaxation operator and the inter-grid transfer operators. In SMG methods, *uniform* coarsening and *linear* interpolation define the coarse grid and the grid transfer operators. Thus, the efficiency of SMG methods is determined by the choice of the relaxation operator, which is chosen to reduce those error components not well approximated by coarse grid correction [12]. AMG methods, on the other hand, work the opposite way. That is, the choice of the relaxation operator is first fixed and then, the coarsening procedure and interpolation technique are chosen to reduce those error components not well reduced by smoothing. AMG method is completely defined once the interpolation operator, $\mathcal{P}_{2h}^h$, is defined. Given $\mathcal{P}_{2h}^h$, AMG defines the restriction operator, $\mathcal{R}_h^{2h}$, and the reduced system matrix, A^{2h}, as follows:

$$\mathcal{R}_h^{2h} = (\mathcal{P}_{2h}^h)^T \quad \text{and} \quad A^{2h} = (\mathcal{P}_{2h}^h)^T A^h \mathcal{P}_{2h}^h \tag{3}$$

Then, the overall AMG process to solve the linear system $A^h x^h = b^h$ can be summarized as follows:

1. Reduce the original grid Ω^h to obtain a smaller grid Ω^{2h}.
2. Define an interpolation operator, $\mathcal{P}_{2h}^h$, to satisfy the AMG requirements pointed out above. Once $\mathcal{P}_{2h}^h$ is defined, $\mathcal{R}_h^{2h}$ and A^{2h} are also defined.
3. Map the problem to the coarser grid using the restriction operator, $b^{2h} = \mathcal{R}_h^{2h} b^h$.
4. Solve the reduced linear system $A^{2h} x^{2h} = b^{2h}$ for the vector, x^{2h}.
5. Map the solution back to the original fine grid using the prolongation operator, $x^h = \mathcal{P}_{2h}^h x^{2h}$.

Thus, it is clear that the efficiency of AMG methods is determined by the choice of the coarsening procedure and the interpolation method [12].

An error is defined to be algebraicly smooth if it is characterized by the fact that the residual, $r = Ae$, is small compared to the error, $r \ll e$ [12]. Furthermore, it is expected that on average $|r_i| \ll a_{ii}|e_i|$ [12]. This observation proves useful in providing a good approximation of the error in terms of its neighboring error values:

$$0 = a_{ii}e_i - r_i + \sum_{j \in N_i} a_{ij}e_j \approx a_{ii}e_i + \sum_{j \in N_i} a_{ij}e_j \tag{4}$$

where $N_i = \{j \neq i : a_{ij} \neq 0\}$ denotes the neighborhood of i. Geometrically, N_i denotes the grid nodes which are directly connected to node i.

Furthermore, since the A matrix is an $\mathcal{M}$-matrix, it can be shown that the error satisfies the following inequality [12]:

$$\sum_{j \neq i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} \ll 2 \tag{5}$$

It can be seen from (5) that the smooth error varies *slowly* in the direction of strong connections. That is, if $\frac{|a_{ij}|}{a_{ii}}$ is relatively large, then $(e_i - e_j)$ has to be small and thus, variation in the error values between nodes i and j is small. In AMG, (4) and (5) provide a mechanism for defining a *good* interpolation operator as well as guide most grid reduction algorithms [12].

3 Proposed Multigrid-like Analysis

Our technique is inspired by the algebraic multigrid method, AMG. Indeed, we follow the exact AMG steps to provide an *efficient* solution for the linear system $A^h x^h = b^h$ corresponding to the original fine grid, Ω^h. That is, the grid is first reduced and an interpolation operator is defined. Then, the problem is mapped to the coarser grid, solved at the coarser grid, and then, the solution is mapped back to the original fine grid. However, regular AMG, driven by the need for a good interpolation operator [12], imposes a grid reduction mechanism which may be inefficient for our needs, because it yields a grid which is not sufficiently coarse to yield the advantages required for fast power grid analysis.

Instead, we can do better than that by using a grid reduction algorithm which is similar to the SMG reduction methods (see section 3.1 below). Then, once the grid is reduced, our method defines an interpolation operator so as to maintain the error requirements of AMG (see section 3.2 below). Thus, the proposed technique combines the advantages of both SMG and AMG, while avoiding some of their limitations. Finally, our method completely ignores the *relaxation* step of multigrid which is applied to *smooth* the error. This is motivated by the fact that well-designed power grids are characterized by *smooth* voltage variation over the grid [3]. Consequently, the proposed approach promises significant speed-ups for transient analysis (see section 3.3 below). As a result *the general structure of our technique consists of repeatedly coarsening the grid until the problem is small enough to solve exactly using a direct approach, and then mapping the solution back to the original fine grid*. We note here that if the original grid is characterized by non-smooth voltage variation, then the relaxation step of the mutligrid must be included in order to maintain the accuracy of the solution.

An interesting question is how to handle the voltage sources and the current sources which are placed at nodes that may be removed to reduce the grid. Our technique handles voltage sources by always keeping the nodes where voltage sources are located at the reduced grid. This is guaranteed by passing to the reduction algorithm a list of nodes that should be kept (see section 3.1). Note that this doesn't severely affect the efficiency of the reduction because typical power grids have a small number of voltage sources (thousands in power grids consisting of millions of nodes). As for the current sources, a current source placed at a removed node, i, gets split into current sources at those nodes from which i will be interpolated. This is taken care of automatically by our technique because the problem is mapped to a coarser grid using the restriction operator, $b^{2h} = \mathcal{R}_h^{2h} b^h$ where the restriction operator is the transpose of the interpolation operator, $\mathcal{R}_h^{2h} = (\mathcal{P}_{2h}^h)^T$.

Table 1: Meaning of status flags.

Status Flag	Indication
N	No flag (default)
K	Kept
H	Visited Horizontally
V	Visited Vertically
R	Removed

3.1 Grid Reduction

A natural method for efficient grid reduction, inspired by SMG, is to skip every other wire, resulting in a situation as in Fig. 1. Such a reduction technique promises significant reduction of the grid because it reduces each dimension of the grid by half (roughly), thus reducing the whole grid by almost a factor of four. While it is straightforward to apply such a reduction technique to regular grids, it is not clear how it can be applied to *general* grids; specifically, *irregular* grids. Since typical power grids may be *irregular*, we need to define a reduction algorithm which systematically reduces a *general* grid. Furthermore, the algorithm should maintain the structure of the original grid (so that it includes only horizontal or vertical edges) so that it can be recursively applied until a coarse enough grid is obtained.

The major objective of the reduction algorithm is to remove as many nodes as possible while maintaining the ability to estimate voltages at the removed nodes by interpolation. The algorithm takes as input a fine grid Ω^h and a list of nodes to be *kept* and produces as output a reduced grid Ω^{2h} with a smaller number of nodes. The list of *kept* nodes should consist of specific nodes of interest to the user, but our technique automatically generates a default list containing the corner nodes and nodes where voltage sources are located. To summarize, our reduction algorithm follows the methods of grid reduction employed by SMG (skipping every other wire). However, it also supports general *irregular* grids as well as handles any user-specified requirement of *keeping* certain nodes at the coarser grid. We assume that the grid lines are either horizontal or vertical. By *neighbors* of a node we mean its immediate neighbors, those connected to it by a vertical or horizontal edge; there can be at most four of these. By *diagonal neighbors* of a node, we mean those nodes that can be reached from it by making two steps, first horizontally and then vertically or first vertically and then horizontally; a node can have more than four diagonal neighbors (as is the case for node r in Fig. 4, for example).

The algorithm makes use of certain status flags, which are shown in Table 1, to decide whether a node is to be kept or removed. Furthermore, these flags indicate how to interpolate the voltage at a removed node from its kept neighbors. The grid reduction algorithm makes repeated use of a so-called node *update* operation, which is defined as follows: *Starting from that node, go along a horizontal (vertical) direction and flag all visited nodes with H (V). Flag extremities as kept. A node*

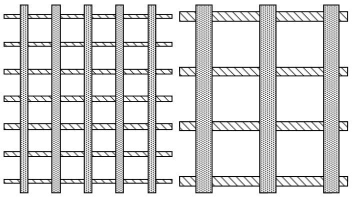

Figure 1: Multiple resolution power grids.

which is visited both horizontally and vertically (flagged with both H and V), is flagged as kept. The algorithm consists of four passes described as follows:

1. First Pass: *Update* every *kept* node. Note that if the *update* operation on a *kept* node i causes another node j to be flagged as K (*kept*), then the *update* operation is also performed on node j.
2. Second Pass: Go through the grid nodes in the given order. In our implementation, grid nodes are ordered from top to bottom, left to right. However, this is not a limitation of the algorithm. If a node i is flagged as H (V) after the first pass, flag that node i as R (*removed*), flag its neighbors along the same row (column) as K (*kept*) and *update* those neighbors. On the other hand, if a node i is not flagged (N) after the first pass, flag that node i as R (*removed*), flag its *diagonal* neighbors as K (*kept*), and *update* those nodes.
3. Third Pass (defines reduced grid): A node i which is flagged as *kept* is a node of the reduced grid; that is, $i \in \Omega^{2h}$. A node i which is flagged as H (V) after the first pass and then flagged as R after the second pass is removed and its horizontal (vertical) neighbors, j and k, are connected by an edge. The resistance of the new edge between j and k is the sum of the resistances of the edge between i and j and the edge between i and k. As for a node i which is flagged as N after the first pass and flagged as R after the second pass, that node i is removed together with its (horizontal and vertical) neighbors without affecting the reduced grid, Ω^{2h}. Note that defining the reduced grid this way maintains the structure of the original grid (so that it includes only horizontal or vertical edges) which allows for recursive applications of the reduction algorithm to produce coarser and coarser grids.
4. Fourth Pass (defines interpolation): Voltage of a *kept* node is the same as that computed at the coarser grid. Voltage of a node i which is flagged as H (V) after the first pass and flagged as R after the second pass is interpolated from its row (column) neighbors' voltages. Voltage of a node i which is flagged as N after the first pass and flagged as R after the second pass is interpolated from its *diagonal* neighbors which are *kept*.

We next describe how the algorithm works and how the flags associated with the different grid nodes change as we go through the different passes of the algorithm. Initially, all grid nodes are flagged as N except for those nodes required by the user to be *kept*. Nodes required to be *kept* are flagged as K. After the first pass of the algorithm, a node i will be flagged as K, H, V, or N. Node i is flagged as K if node i should be *kept* at the coarser grid. This may occur if one of the following conditions holds for node i:

1. Node i is a node passed by the user to be *kept*. In this case, the initial flag of node i is K and after the first pass, the flag of node i is still K (*kept*).
2. Node i is the extremity of a row or column that consists of some other node j which is flagged as K (*kept*).
3. Node i is the intersection of a row and a column that have been both *visited*. A row r (column c) is said to be *visited* if r (c) consists of at least one node j which is *updated*.

On the other hand, a node i is flagged as H (V) after the first pass if node i belongs to a row (column) that is *visited* during the first pass. That is, i belongs to a row r (column c) which consists of some other node j that is *updated* during the first pass. Finally, a node i is flagged as N after the first pass if it is the intersection of a row and a column that are both not visited during the first pass. Note that if a node i is flagged as N after the first pass, then no other node j which belongs to the same row or same column as i can be flagged as K (*kept*) after the first pass.

The second pass involves going through the nodes in the given order and checking their flags (obtained after first pass). If a node i is flagged as K (*kept*), nothing is done and the algorithm moves on to the next node. If, on the other hand, a node i is flagged as H (V), then the algorithm flags that node as R (*removed*), flags its horizontal (vertical) neighbors as *kept*, and *updates* those neighbors. Hence, because there is a call to an *update* operation at this stage, a node i with a certain flag after the first pass may be flagged with a different flag at this stage. However, this new flag may still be changed before a final flag is associated with i after the second pass. Finally, if a node i is flagged as N, then the algorithm flags that node as R (*removed*), flags its diagonal neighbors as K (*kept*), and *updates* those diagonal neighbors. Thus, it is guaranteed that after the second pass, every node of the grid is flagged as either K (*kept*) or R (*removed*). A node i flagged as K is *kept* at the coarser grid, Ω^{2h}, $i \in \Omega^{2h}$. On the other hand, a node i flagged as R is *removed*; that is, $i \notin \Omega^{2h}$.

The third pass simply involves going through the nodes and, based on their flags after first and second passes, decide how to build the reduced grid Ω^{2h}. As for the fourth pass, it decides which neighbors of a *removed* node i are used to interpolate the voltage at i.

Before describing how the algorithm works with an example, some observations may be helpful in explaining the algorithm. Following the algorithm, a node i which is marked as H (V) after the first pass and marked as R (*removed*) after the second pass is a node whose horizontal (vertical) neighbors are marked as K (*kept*). The other vertical (horizontal) neighbors of i would be marked as R (*removed*). On the other hand, a node i which is marked as N after the first pass and marked as R (*removed*) after the second pass is a node whose (horizontal and vertical) neighbors are marked as R (*removed*). The voltage at such a

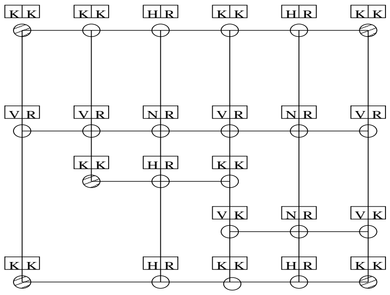

Figure 2: Reduction of an irregular grid.

node would be interpolated from its diagonal neighbors as explained in the algorithm. The only exception to this scenario is when a node i is marked as N after the first pass and then one of its neighbors, j, is marked as K (*kept*) during the second pass. Since j is marked as K during the second pass, this invokes an *update* operation on j which causes i to be flagged as H (V) assuming that i is a horizontal (vertical) neighbor of j. If the final flag of i after the second pass is R (*removed*), then i should be treated as a node flagged as H (V) after the first pass and flagged as R after the second pass. This is because the horizontal (vertical) neighbors of i are flagged as K (*kept*). The given example shows such a scenario at the node which is the intersection of the fourth row and the fifth column (the upper left corner is the intersection of the first row and first column). The tag associated with that node shows that this node is flagged as N after the first pass and as R after the second pass. However, there is an intermediate flag of H associated with that node after the *update* operation is applied to its right neighbor during the second pass of the algorithm.

Another point worth noting concerns building the reduced grid after the nodes are properly flagged as *kept* or *removed*. AMG doesn't worry about building the reduced grid because AMG handles grid reduction by considering the system matrix only. That is, AMG defines which nodes are to be removed and which are to be kept by going through the matrix entries and applying some reduction algorithm. Once that is defined, AMG then defines the interpolation operator which is then used to define the reduced system matrix A^{2h} as explained earlier. Then AMG reduction is simply applied to A^{2h} directly. However, one of the major reasons of the inefficiency of AMG reduction is because AMG doesn't build a representative grid of the problem but rather relies on the matrix for grid reduction.

In order to avoid that, we propose working with the actual grid to define an *efficient* reduction technique. Thus, we need to define a mechanism for building the reduced grid Ω^{2h}. One straightforward technique is to consider the reduced system matrix A^{2h} and accordingly build the reduced grid, Ω^{2h}. This is always possible because there is a one-to-one correlation between actual grids and matrices. However, building the reduced grid this way may produce a rather complex grid Ω^{2h} with diagonal (non-horizontal and non-vertical) edges. Given such a reduced grid Ω^{2h}, it becomes hard to efficiently reduce that grid. Furthermore, it becomes almost impossible to define a grid reduction algorithm than can be recursively applied to produce coarser and coarser grids. For these reasons, we propose a simpler method for building the reduced grid, Ω^{2h}, which maintains the grid structure (so that it includes only horizontal or vertical edges) thus allowing the recursive application of the reduction algorithm. As explained in the reduction algorithm, the method involves first deciding which nodes are *kept* and which nodes are *removed*, and then simply adding edges between *kept* horizontal and vertical neighbors.

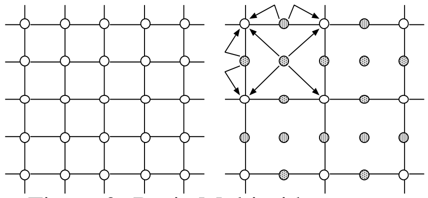
Figure 3: Basic Multigrid operator.

Next we illustrate how the grid reduction algorithm works on the example given in Fig. 2. Initially, all nodes have the default status of N except for the nodes which should be kept. In this example, these would be all the corner nodes of the grid (dashed nodes in Fig. 2). A tag consisting of two fields is associated with every node of the grid. The left field indicates the status of the node after the first pass and the right field indicates the status of the node after the second pass.

As shown in Fig. 2, after the first pass, an edge (row or column) consisting of at least one *kept* node, has its extremities flagged as *kept*. The remaining nodes on that edge are flagged with H or V based on whether the edge is horizontal or vertical. Note that some nodes still have a status flag of N which indicates that these nodes have not been visited during the first pass. Then after the second pass, nodes with a K flag are kept while those with an R flag are removed. Then, the third pass involves adding edges between the *kept* nodes which are neighbors thus resulting in the coarser grid Ω^{2h}.

Finally, we note that if our grid reduction algorithm is applied to a regular grid, then it produces optimal reduction. That is, it reduces the size of the grid by almost a factor of four as illustrated in Fig. 3.

3.2 Interpolation

AMG interpolation is guided by (4) and (5). Thus, the interpolation operator should be chosen to relate the voltage of a *removed* node, i, to the voltages of those *kept* nodes which are *strongly connected* to i. Typically, AMG considers a connection between two nodes to be strong when $|a_{ij}|/\max_{l\neq i}|a_{il}| \geq \theta$, where $0 \leq \theta \leq 1$ (θ is typically chosen to be 0.25 in practice [12]). With such a choice of the interpolation operator, the *coarse grid correction* would efficiently reduce the error.

In our reduction algorithm, the status flags indicate which neighbors of a removed node are to be used for interpolation, based on the fact that they are *kept* and *strongly connected* to a *removed* node m. As for the interpolation weights, these are obtained by considering the values of conductances between the nodes. Thus, if the voltage at a removed node m is interpolated from the voltages at nodes A and B, then the (linear) interpolation function $INT()$ is defined as:

$$V(m) = INT(V(A),V(B)) = a_0V(A) + a_1V(B) \qquad (6)$$

where $a_0 = \frac{g_{mA}}{g_{mA}+g_{mB}}$ and $a_1 = \frac{g_{mB}}{g_{mA}+g_{mB}}$. Here, g_{mA} it is the conductance between nodes m and A, and g_{mB} is the conductance between nodes m and B. Note that our technique for choosing the interpolation weights is similar to the technique used in AMG. To illustrate, consider a *removed* node m whose voltage will be interpolated from the voltages at the *kept* nodes A and B. AMG uses the following interpolation scheme [12]:

$$V(m) = \frac{|a_{mA}|}{a_{mm}}V(A) + \frac{|a_{mB}|}{a_{mm}}V(B) \qquad (7)$$

where a_{mA} is the entry of the A matrix relating nodes m and A, and a_{mB} is the entry of the A matrix relating nodes m and B. As for a_{mm}, one AMG approach is to define it as the diagonal entry of the A matrix corresponding to node m. Another common AMG method defines a_{mm} as: $a_{mm} = |a_{mA}| + |a_{mB}|$. For the power grid problem, $|a_{mA}| = g_{mA}$ and $|a_{mB}| = g_{mB}$, which shows that our interpolation technique is motivated by AMG.

However, this is not the full story. Recall that our grid reduction algorithm differs from AMG grid reduction methods; it is actually based on SMG reduction - it uses only geometric information and removes as many nodes as possible. Hence, it is possible to come across cases where a removed node i has all the nodes that are *strongly connected* to it removed as well. To illustrate this, consider Fig. 4, where a filled node indicates a *removed* node and a blank node indicates a node that is *kept*. In this example, we assume that every horizontal or vertical link represents a *strong connection* but two nodes that are separated by two or more links are not strongly connected. This situation is typical of power grids. Thus, r is strongly connected to m and m is strongly connected to B, but r and B are not strongly connected. Nodes such as B that are separated by two strong links from r, but which are themselves not strongly connected to r, are said to be two-level strongly connected to r. Our reduction would remove node r, as well as all the nodes that are strongly connected to it, m, n, p, and q. However, it can be shown that our algorithm guarantees that, if a node i is removed, either some nodes that are strongly connected to i are kept or some nodes that are two-level strongly connected to i are kept. Therefore, in our interpolation technique, if all strongly connected neighbors of a node have been removed along with it, we use its two-level strongly *kept* neighbors for interpolation. This is clearly illustrated in Fig. 4 where the voltage at node r is interpolated from those nodes which are two-level strongly connected to r; specifically, nodes A, B, C, D, and E. Note that this approach maintains the advantage of efficient grid reduction as well as meets the requirement of a *good* interpolation operator.

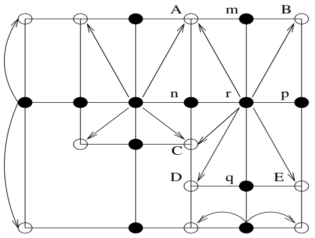

Figure 4: Interpolation from reduced grid nodes.

3.3 Time Domain Analysis

Well-designed power grids are characterized by *smooth* voltage variation. Thus, it is quite feasible to ignore the relaxation step of AMG without jeopardizing the accuracy of the solution. By adopting this approach, our technique actually becomes a *direct* solver as opposed to regular multigrid which is an *iterative* solver. Direct solvers offer significant speed-ups over iterative solvers when transient analysis is performed [1, 3, 4]. However, the major problem with direct solvers is their high memory requirement [1]. As a matter of fact, it may be impossible to solve a very large system using a direct solver. In such cases, an iterative solver has to be used and the problem is seriously aggravated when performing transient analysis because an iterative solver has to be used at every time step.

However, our approach offers an efficient solution to this problem because it uses a direct solver to solve a reduced system of much smaller dimension. Thus, our technique avoids the memory limitation of *direct* solvers while maintaining their speed-up advantage. Of course, the advantages of the proposed technique come at a slight cost in the accuracy of the solution since the relaxation step is ignored and interpolation is used. However, the results in the next section show that this error is small enough to maintain the efficiency and suitability of the proposed technique.

4 Experimental Results

The proposed multigrid method has been implemented and integrated into a linear simulator written in C++. All experimental results reported in this section were obtained by running the simulations on a 400MHz ULTRA 2 Sun workstation with 2GB of RAM and running the SunOS 5.7 operating system.

The practicality and efficiency of the proposed technique are illustrated by applying it for the analysis of the power grids of two *real* industrial ASIC designs. We will refer to these designs as C_1 and C_2. Both designs, C_1 and C_2 are 0.18μ CMOS designs and have a supply voltage of 1.8 V. Given the power grid, the technique requires as input the currents associated with the different power drains on the chip.

Different current measures can be used for the analysis depending on the application of interest. For instance, while peak current is a good representative measure for IR drop, average current is a better measure for electromigration analysis. On

Table 2: Grid reduction and CPU times (sec) using exact solve as well as our multigrid-like (MG) technique.

Design	Level	# of nodes	Exact time	MG time
C_1	0	318074	456.79	21.6
	1	187630		
	2	128864		
	3	101209		
	4	86883		
C_2	0	671088	1114.13	69.28
	1	421460		
	2	310143		
	3	258591		
	4	231982		

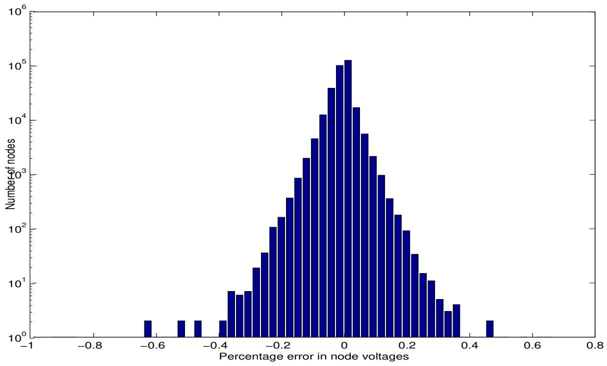

Figure 5: Error in nodes voltages for the C_1 design.

the other hand, a current waveform is the suitable current measure for transient analysis. A straight-forward technique for obtaining any current measure of interest is to simulate the power drains under *nominal* loads and *realistic* switching factors. This is how the current measures we used for our analysis were obtained. In all our experiments for DC analysis, we used the peak current drawn by the power drains as our current measure. As for transient analysis, the current measure used was the current waveform associated with the different power drains.

The irregular power grids of the two designs, C_1 and C_2, were simulated. The vias between consecutive layers are modeled as ideal shorts and the proposed approach is applied only to the two lowest metal layers (M_1, M_2). This is motivated by the fact that the lowest metal layers contribute the largest number of nodes. The nodes contributed by the higher metal layers are maintained as they are.

Several grid reductions are applied and the problem accordingly mapped to the coarser grids (as explained earlier, the reduction is repeated until the grid is coarse enough as specified by the user). Specifying four levels of reduction, Table 2 shows the number of nodes of the grid at every level. Table 2 also shows the CPU times for solving the given linear system using both a regular direct solver (shown in column 4) as well as the proposed multigrid-like technique (shown in column 5). It is clear that the proposed technique is almost 16× to 20× faster

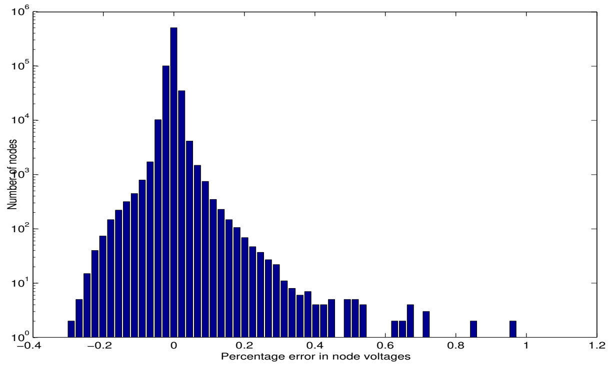

Figure 6: Error in nodes voltages for the C_2 design.

Table 3: Grid reduction and CPU times for transient analysis.

Design	Exact transient time	MG transient time
C_1	921.16 seconds	28.32 seconds
C_2	14.3 hours	86.36 seconds

than traditional simulation. Note that the same direct solver is used for solving both the original system as well as the reduced system which verifies that the speedup is not due to an advantage of one solver over another. In order to verify the accuracy of the results, the exact solution is compared to the estimated solution returned by our technique. The histograms of percentage errors in the voltages of the different nodes of the power grids corresponding to the two designs, C_1 and C_2, are shown in Figs. 5 and 6 respectively.

For the design C_1, the distribution of the node voltage errors has a mean of -0.0077% and a standard deviation of 0.0333%. For C_2, the error distribution has a mean of -0.0026% and a standard deviation of 0.0167%. Furthermore, Figs. 5 and 6 also show that the errors at all the power grid nodes of both designs lie in the -1.0% to 1.0% range. In fact, design C_1 has errors that range from -0.93% to 0.66% while design C_2 has errors that range from -0.30% to 1.0%. Thus, it is clear that the proposed technique provides an accurate solution to the power grid problem at a significant speed-up over regular solvers.

As explained earlier, the proposed multigrid-like technique is even more advantageous when applied for transient analysis. This is illustrated in Table 3, which shows the time required to run a transient simulation of the power grids of the two designs using a regular solver and the proposed technique. The power grids are simulated for a duration of 4 ns with 0.4 ns time steps. The speedup advantage is clear in both cases. However, it is more significant in the case of the C_2 design and the reason is that design C_2 is simulated using an iterative solver due to memory limitations, requiring a total of 14.3 hours. Our technique, on the other hand, uses a direct solver to solve the problem at the reduced grid. Thus, only one initial factorization is needed and only forward/backward solves are needed at the remaining time steps. The total time required for transient analysis using the proposed multigrid-like technique is 86.36 seconds, representing a speed-up of 600× for transient analysis.

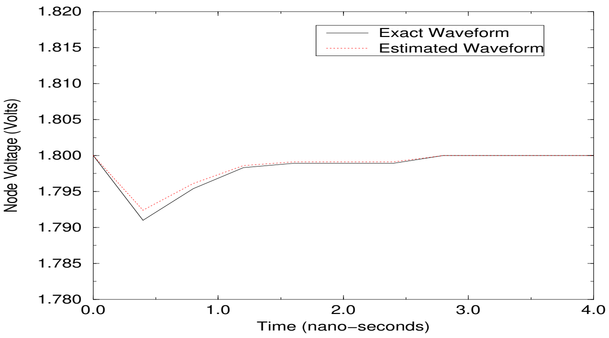

Figure 7: Error in the voltage waveform n the C_1 design.

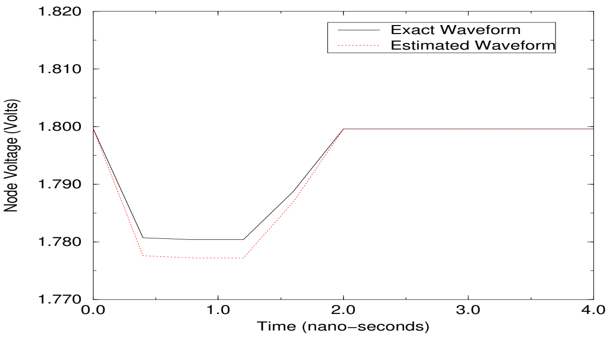

Figure 8: Error in the voltage waveform in the C_2 design.

Other methods to speed-up power grid analysis have been proposed [4]. In [4], a hierarchical power grid analysis technique is proposed which gives speed-ups between $2\times$ and $5\times$ for DC analysis. The authors also propose utilizing parallelism which increases the speed-ups to the range $10\times$ to $23\times$ [4]. However, their proposed method offers no speed-ups when transient analysis is applied in serial mode. Smaller speed-ups between $1.8\times$ and $5.1\times$ can still be observed when parallel execution is used for transient analysis. Note that the speed-up comparison is a function of the linear solvers being used as well as the size of the problems being solved. However, experimental results show that our method promises more significant speed-ups at a minimal cost in accuracy. Furthermore, these speed-ups are evident for both DC as well as transient analysis.

Finally, it remains to verify the accuracy of the resulting transient solution. This is illustrated by Figs. 7 and 8 which show the voltage waveform at one node of the power grids of designs C_1 and C_2 respectively. It is clear that the multigrid-like technique accurately tracks the exact voltage waveform at the given node. Comparisons at other nodes show similar results. Figs. 7 and 8 show an error in the node voltage of under 0.17% or about 3 mV (we find a voltage drop of 23 mV, while the exact waveform shows a drop of 20 mV).

Thus, the multigrid-like technique provides very accurate simulation results for both DC as well as transient analysis of the power grids with the added advantage of significant speed-up over regular analysis techniques.

5 Conclusion

An efficient PDE-like method for power grid analysis is presented. It follows the basic lines of thought of the multigrid technique which is widely used for the solution of smooth PDE problems. However, the proposed technique falls under the category of *direct* solvers and thus, significantly differs from the regular multigrid method which falls under the category of *iterative* solvers. Experimental results on *real* designs show speed-ups of one to two orders of magnitude over current methods for both DC and transient analysis.

References

[1] A. Dharchoudhury et. al. Design and analysis of power distribution networks in *PowerPC^{TM}* microprocessors. 35^{th} Design Automation Conference, San Francisco, 1998.

[2] G. Steele et. al. Full-chip verification methods for DSM power distribution systems. 35^{th} Design Automation Conference, San Francisco, 1998.

[3] S. R. Nassif and J. N. Kozhaya. Fast power grid simulation. 37^{th} Design Automation Conference, Los Angelos, CA, 2000.

[4] M. Zhao et. al. Hierarchical analysis of power distribution networks. 37^{th} Design Automation Conference, Los Angelos, CA, 2000.

[5] H. H. Chen and J. S. Neely. Interconnect and circuit modeling techniques for full-chip power noise analysis. IEEE Transactions on Components and Packaging II, 1998

[6] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic and System Simulation Methods*. McGraw-Hill, Inc. 1995.

[7] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Science*. Academic Press, New York 1979.

[8] W. L. Briggs. *A Multigrid Tutorial*. SIAM, 1987.

[9] A. Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. Proceedings Third International Conference on Numerical Methods in Fluid Mechanics, Paris 1972. Springer-Verlag Berlin, 1973.

[10] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag Berlin, 1985.

[11] K. St*ü*ben, and U. Trottenberg. *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*. Multigrid Methods Proceedings, edited by W. Hackbusch and U. Trottenberg K*ö*ln-Porz, 1981. Springer-Verlag Berlin, 1982.

[12] J. W. Ruge and K. St*ü*ben. *Algebraic Multigrid*. Multigrid Methods, edited by S. McCormick. SIAM 1987.