Hierarchical Analysis of Power Distribution Networks *

Min Zhao, Rajendran V. Panda, Sachin S. Sapatnekar*, Tim Edwards, Rajat Chaudhry, David Blaauw
Advanced Tools, Motorola Inc., Austin, TX
*Dept. of ECE, University of Minnesota, Minneapolis, MN

Abstract

Careful design and verification of the power distribution network of a chip are of critical importance to ensure its reliable performance. With the increasing number of transistors on a chip, the size of the power network has grown so large as to make the verification task very challenging. The available computational power and memory resources impose limitations on the size of networks that can be analyzed using currently known techniques. Many of today's designs have power networks that are too large to be analyzed in the traditional way as flat networks. In this paper, we propose a hierarchical analysis technique to overcome the aforesaid capacity limitation. We present a new technique for analyzing a power grid using macromodels that are created for a set of partitions of the grid. Efficient numerical techniques for the computation and sparsification of the port admittance matrices of the macromodels are presented. A novel sparsification technique using a 0-1 integer linear programming formulation is proposed to achieve superior sparsification for a specified error. The run-time and memory efficiency of the proposed method are illustrated through the analysis of case studies of several multi-million node power grids, extracted from real microprocessor and DSP designs.

1 Introduction

With the increase in the complexity of VLSI chips, designing and analyzing a power distribution network has become a challenging task. A robust power network design is essential to ensure that the circuits on a chip operate reliably at the guaranteed level of performance. A poorly designed power network can become the cause for a variety of problems such as loss of circuit performance, noise generation, and electro-migration failures. With the increased power level and device densities of microprocessors in sub-micron technologies, these problems are more likely unless serious attention is given to power network design. Critical to obtaining a robust design is the ability to analyze the network efficiently several times in the design cycle. Several previously published research works [1–4] discuss methodologies and techniques to accomplish this task efficiently.

The difficulty in power network analysis stems mainly from three sources: (i) the network is very large, typically 1 million to 100 million nodes, (ii) the network is nonlinear as it contains switching devices, and (iii) the voltage and current distribution in the network is dependent on the instruction executed on the processor. Our work, presented in this paper, addresses the first problem. The second problem is circumvented traditionally [1] by performing nonlinear simulation of individual circuit blocks without including the parasitics in the power interconnects, and then simulating the power interconnect as a whole using the time-variant current profiles, obtained in the nonlinear simulation as the excitation sources. The third problem is one of obtaining a good coverage of all possible worst case power demand situations. Manually generated "hot loops," an extensive set of input vectors, and statically generated worst case current profiles [5–7] are some of the alternatives that address the worst case coverage issue.

The size based complexity of the problem has been partly addressed earlier [1, 2] by using very efficient sparse linear system solution techniques. Cholesky factorization (direct method) [8] and conjugate gradient techniques with pre-conditioners (iterative method) [8] have been used to solve the linear system associated with the power grid. These specialized techniques operate very efficiently by exploiting the special structure and properties of the underlying linear system. However, the solutions proposed earlier have applied these techniques to a flat (nonhierarchical) model of the power network. As a result, there is a serious limitation on the size of the problem they can solve, the limitation being imposed by the amount of memory available for computation. At the current technological level, it is seen that the available computing resources are insufficient to simulate very large power grids of today's microprocessors using a flat model. The size of the power grid of a typical high performance microprocessor in 0.18 micron design, and using 6 layers of metal, is in the range of 10 million to 100 million nodes. Thus the power grid simulation would require solving a linear system of similar size at multiple time points. Clearly, the speed and memory capacity of a typical computing environment is insufficient to solve such a large system even with the most efficient linear system solution techniques.

In this work, we propose a hierarchical analysis technique to overcome the limitations of the traditional approach based on flat power grid model. Our approach comprises of the following steps: (1) Partitioning of the power grid into local and global grids, using the hierarchical structure in the design, (2) Generating macromodels for the local grids using efficient numerical methods, (3) Sparsifying the port admittance matrices of the macromodels, while maintaining the accuracy of the solution, (4) Simulating the global grid after augmenting it with the macromodels of the local grids, and finally, (5) Simulating the local grids where desired.

The basic strength of the proposed approach is derived from the well-known strategy of "divide and conquer," which is realized through partitioning. However, the efficiency and usefulness of the hierarchical approach is sensitive to several factors, such as the partitioning technique and the memory and runtime costs involved in generating the macromodels. Our work in this research addresses these problems in order to realize a practical and efficient implementation of the hierarchical analysis strategy. We propose a partitioning strategy that reduces the memory required for storage in our hierarchical simulation approach. Moreover, a novel matrix sparsification technique based on 0-1 integer linear programming is proposed to further reduce the memory requirements. Additionally, an efficient numerical procedure for calculating the macromodels is given. The computation takes advantage of the fact that the underlying linear system is symmetric and positive definite. The proposed approach has been applied for analyzing the power grid of a number of high performance microprocessors and DSP chips, obtaining significant memory and runtime advantages over the flat model analysis approach. To our knowledge, no work has been reported so far to address this critical issue of limitation on the size of power grids analyzable using current approaches.

The remainder of the paper is organized as follows. In section

*This work was supported in part by the Semiconductor Research Corporation under contract number 99-TJ-714 and by the National Science Foundation under contract number CCR-9800992

2, we present the concept of macromodeling and the partitioning strategy. Also presented in that section are the computational techniques for generating the macromodels. In section 3, the matrix sparsification technique is explained. Section 4 reports the performance results of the proposed approach for a set of benchmark designs, followed by conclusions in section 5.

2 Macromodeling Approach

2.1 Overview of Power Grid Simulation

Before presenting the macromodeling approach, let us present an overview of power grid simulation in general. A chip's power distribution system is modeled as a linear RLC network with independent time-varying current sources modeling the switching currents of the transistors. Simulating the network requires solving the following system of differential equations, which are formed in a typical approach such as the Modified Nodal Analysis (MNA) [9] approach:

$$\mathbf{G}\cdot\mathbf{x}(t)+\mathbf{C}\cdot\mathbf{x}'(t)=\mathbf{b}(t), \tag{1}$$

where **G** is a conductance matrix, **C** is the admittance matrix resulting from capacitive and inductive elements, **x**(t) is the time-varying vector of voltages at the nodes, and currents through inductors and voltage sources, and **b**(t) is the vector of independent time-varying current sources. This differential system is very efficiently solved by reducing it to a linear algebraic system

$$(\mathbf{G}+\mathbf{C}/h)\cdot\mathbf{x}(t)=\mathbf{b}(t)+\mathbf{C}/h\cdot\mathbf{x}(t-h), \tag{2}$$

using Backward Euler (BE) technique with a fixed time step, *h*. The BE reduction with fixed time stepping is advantageous for transient simulation since the left hand side (LHS) matrix $(\mathbf{G}+\mathbf{C}/h)$, referred to as the coefficient matrix, is rendered stationary, allowing either pre-processing or factoring of the matrix for a one-time cost and reusing it efficiently to solve the system at successive time points.

When **x** consists only of node voltages, as in the case of a modified nodal formulation of a network with R's, C's, and current sources only, the coefficient matrix can be shown to be symmetric and positive definite. A symmetric positive definite formulation is feasible even when inductive elements are included in the analysis, although this would involve an additional reduction step from the modified nodal formulation. The symmetric positive definiteness of the coefficient matrix, which is also very sparse, is especially attractive as the system can now be solved very efficiently using specialized linear system solution techniques, such as Cholesky factorization (direct method) and Conjugate Gradient (iterative method) techniques. The direct method through Cholesky factors is very cost-effective for simulations at multiple time points, as the expensive step of factoring is performed only once initially and its cost is amortized over multiple time point solutions. Successive solutions would involve only inexpensive forward and backward substitution procedures. Although the macromodeling techniques presented in this paper are suitable for use with either type of solution approach, direct or indirect, we will assume, for simplicity of presentation, that the underlying linear solver is direct.

2.2 Basic Idea

The run time and memory requirement for solving a linear system is determined primarily by the size, sparsity, and structure of the coefficient matrix. If the network is very large (10^7 - 10^8 nodes), the available physical and virtual memory of the system is insufficient even for loading in the data associated with the network. Even when the base memory requirement is met, memory demand quickly grows during the matrix factorization process, due to new fills being created. Given a reordering scheme, the number of fills created is determined by the initial sparsity and structure of the matrix. The sparsity is given by the ratio of the number of elements in the network to the number of nodes. While tree-like network structures have low fills, mesh structures generally tend to have large fills during factorization. The amount of matrix computation being very sensitive to the sparsity and fill pattern, it is very desirable to have the initial matrix as sparse as possible. The objective of the proposed approach is, hence, twofold - (i) to reduce the size of the problem, and (ii) to maintain a high degree of sparsity in the reduced problem.

The first objective is met by partitioning the given network into subnetworks of manageable size, and solving the network by solving the sub-pieces individually. Since the entire network is tightly connected, we cannot ignore the interaction between the various partitions without incurring significant error. So, in order to account for the interactions between partitions, while at the same time not enlarging the size of the problem at hand, we use models for the partitions that capture their behavior as observed at their interface nodes (also called ports). We refer to these models as macromodels. A macromodel is a multi-port linear circuit element that has the same linear relation between the voltages and currents through its ports as the partition itself. With macromodels for partitions available, the original (unpartitioned) network is efficiently solved after replacing the partitions by the respective macromodels, as the macromodels are much smaller in size than the partitions themselves.

The gains made through partitioning can be quickly lost if the partitions generate very dense macromodels, and thus increase the effective size of the problem. Our approach addresses this issue in two ways. First, the partitioning is performed strategically as explained in section 2.4. Then, an optional step of sparsification can be applied to the generated models. The key issue in sparsification is not to compromise accuracy of the final solution. The sparsification technique is covered in section 3.

Besides the memory advantage, the macromodeling approach provides a significant speedup as the creation of macromodels for the partitions can be performed in parallel.

2.3 Hierarchical Modeling

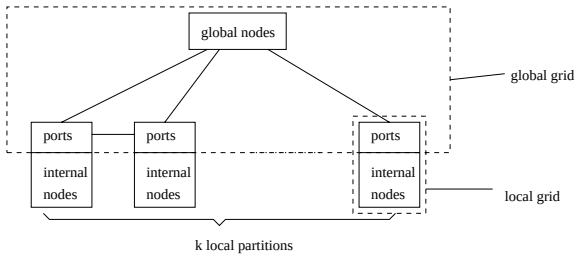

Figure 1: Hierarchical power network analysis

The macromodel approach to power grid analysis is illustrated in Figure 1. Let us consider a division of the entire power network into one global partition and *k* local partitions. A node in a local partition having links only to other nodes in the same partition is called an *internal node*, a node in the global partition is called a *global node*, and a node in a local partition that is connected to some node outside the local partition (i.e., in the global partition or in another local partition) is called a *port*. The *global grid* is then defined to include the set of nodes that lie in the global partition and the port nodes, while the grid in a local partition constitutes a *local grid*.

Now each of the *k* local grids can be modeled as a multi-port linear element with transfer characteristics given by

$$\mathbf{I}=A\cdot\mathbf{V}+\mathbf{S},\quad \mathbf{I}\in\mathbf{R}^m, A\in\mathbf{R}^{m\times m}, \mathbf{V}\in\mathbf{R}^m, \mathbf{S}\in\mathbf{R}^m \tag{3}$$

where *m* is number of ports in the local grid, *A* is the port admittance matrix, **V** is the vector of voltages at the ports, **I** is the current through the interface between the local and global grids, and **S** is a vector of current sources connected between each port and the reference node. **S** essentially has the effect of moving all the current sources internal to a local grid to the ports of the multi-port model. We refer to the set (*A*,**S**) as the macromodel of the respective local grid. The macromodel (*A*,**S**) in equation (3) is obtained through a reduction procedure starting from the modified nodal equations of the local grid expressed in the form:

$$G \cdot \mathbf{U} = \mathbf{J}, \quad G \in \mathbf{R}^{n \times n}, \mathbf{U} \in \mathbf{R}^{n}, \mathbf{J} \in \mathbf{R}^{n} \tag{4}$$

where n is number of nodes in the local grid, G is the coefficient matrix, **U** is the voltage vector of the nodes of the local grid, and **J** is vector of currents that flow out of each node in the local grid. For the port nodes, **J** would also include the currents through the interface between the local and global grids. The procedure of deriving the transfer characteristic in Equation (3) from the modified nodal equation of (4) is referred to as macromodeling, and will be addressed in detail in section 2.5.

Once the macromodels for all the local grids are generated, the entire network can be abstracted simply as the global grid, with the macromodel elements connected to it at the port nodes. This is achieved by combining the coefficient matrix and the RHS current vector of the global grid with the macromodels, (A,**S**); Equations (3) of each local grid may be stamped into the modified nodal equations of the global grid as follows.

$$\begin{bmatrix} G_{00} & G_{01} & G_{02} & \dots & G_{0k} \\ G_{01}^T & A_1 & G_{12} & \dots & G_{1k} \\ G_{02}^T & G_{12}^T & A_2 & \dots & G_{2k} \\ \vdots & & & & \vdots \\ G_{0k}^T & G_{1k}^T & G_{2k}^T & \dots & A_k \end{bmatrix} \begin{bmatrix} \mathbf{V_0} \\ \mathbf{V_1} \\ \mathbf{V_2} \\ \vdots \\ \mathbf{V_k} \end{bmatrix} = \begin{bmatrix} \mathbf{I_0} \\ \mathbf{-S_1} \\ \mathbf{-S_2} \\ \vdots \\ \mathbf{-S_k} \end{bmatrix} \tag{5}$$

In the above equation,

- global nodes are labeled as partition 0
- G_{ij} represents the conductance links between partition i and partition j.
- $\mathbf{I_0}$ is the vector of currents that flow out of the global nodes.
- $\mathbf{S_i}$ is the constant vector of partition i.
- $\mathbf{V_i}$ is voltage vector of partition i.
- A_i is the port admittance matrix of partition i, where $i \in [1,k]$.

This is a system of $(n_0 + m_1 + m_2 + \ldots + m_k)$ linear equations, where n_0 is the number of global nodes and m_i is the number of ports in each partition.

From the above reduction scheme, the voltages and currents in the entire power grid can be solved in the following steps:

- Obtain global grid voltages by solving equation (5).
- For each partition, obtain **I** from equation (3) using the port voltages
- Solve equation (4) *for each partition* using **I** on the right hand side, to obtain voltages at the internal nodes of partitions.

The flow of the macromodel approach is illustrated in Figure 2.

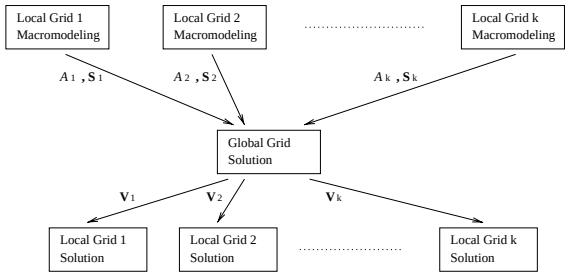

Figure 2: Flow of the macromodel

2.4 Partitioning Strategy

The main difficulty in macromodeling is that the model is often fully dense even though the partition from which it is created itself may be very sparse. Note that the entries of matrix A in equation (8) are admittances of paths between pairs of ports. Thus, a nonzero entry at position (i, j) results if there is a conducting path in the partition between these ports, even though there may not be a direct link between these ports. As a result, the number of nonzero entries in A is $O(m^2)$, where m is the number of ports, unless the grid inside the partition is heavily fragmented. Nevertheless, there is a substantial win if m^2 is much smaller than the number of nodes in the partition that are abstracted away by this model. Thus, the key idea in the partitioning strategy is to identify a subnetwork and a interface boundary such that the number of internal nodes is much larger than the square of the number of nodes at the interface.

Fortunately, the natural hierarchical boundaries of circuit blocks often meet the above criteria. For instance, a large memory array with 3 local metal layers may have several millions of internal nodes, but it may have very few (hundreds of) nodes interfacing with the upper layer of the global grid, and almost none with other circuit blocks. Although one can have a sophisticated partitioning strategy, we have found in practice that a simple inspection procedure of checking every circuit block or a group of adjacently placed blocks for the above criteria works very well.

2.5 Macromodeling

Macromodeling is the procedure of deriving Equation (3) from the modified nodal equations of the partition. The modified nodal equations for a partition can be written as

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} \begin{bmatrix} \mathbf{U_1} \\ \mathbf{V} \end{bmatrix} = \begin{bmatrix} \mathbf{J_1} \\ \mathbf{J_2} + \mathbf{I} \end{bmatrix} \tag{6}$$

where

- $\mathbf{U_1}$ and $\mathbf{V}$ are vectors of voltages at the internal nodes and ports respectively
- $\mathbf{J_1}$ and $\mathbf{J_2}$ are vectors of current sources connected at the internal nodes and ports respectively
- $\mathbf{I}$ is the vector of currents through the interface
- G_{12} is the admittance of links between the internal nodes and the ports
- G_{11} is the admittance matrix of internal nodes
- G_{22} is the admittance matrix of ports.

From (6), we may rewrite the first set of equations as

$$\mathbf{U_1} = G_{11}^{-1}(\mathbf{J_1} - G_{12}\mathbf{V}) \tag{7}$$

Substituting this value of $\mathbf{U_1}$ into the second equation of (6), we get

$$\mathbf{I} = (G_{22} - G_{12}^T G_{11}^{-1} G_{12})\mathbf{V} + (G_{12}^T G_{11}^{-1}\mathbf{J_1} - \mathbf{J_2}) \tag{8}$$

Here, $G_{12}^T G_{11}^{-1}\mathbf{J_1} - \mathbf{J_2}$ is the constant vector **S** in Equation (3) and $G_{22} - G_{12}^T G_{11}^{-1} G_{12}$ is the port admittance matrix A in Equation (3).

It may be noted that the pre-multiplication and post-multiplication operations with G_{11}^{-1} can be carried out without explicitly inverting G_{11}, but through multiple invocation of the direct or iterative solver.

The above calculation can be made very efficient using the fact that the coefficient matrix, G is symmetric and positive definite. We show below how A and **S** can be computed efficiently from the submatrices of the Cholesky factors, rather than the Cholesky factors themselves. Relating G_{11}, G_{12}, and G_{22} to the submatrices of the Cholesky factors of G, we have

$$\begin{aligned} \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} \\ &= \begin{bmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{bmatrix} \end{aligned}$$

Now computing A in terms of submatrices of factors, we get

$$\begin{aligned} A &= G_{22} - G_{12}^T G_{11}^{-1} G_{12} \\ &= L_{21}L_{21}^T + L_{22}L_{22}^T - L_{21}L_{11}^T (L_{11}L_{11}^T)^{-1} L_{11}L_{21}^T \\ &= L_{22}L_{22}^T \end{aligned}$$

Similarly, vector **S** is given by

$$\begin{aligned}\mathbf{S} &= G_{12}^T G_{11}^{-1}\mathbf{J_1} - \mathbf{J_2} \\ &= L_{21}L_{11}^T(L_{11}^T)^{-1}L_{11}^{-1}\mathbf{J_1} - \mathbf{J_2} \\ &= L_{21}L_{11}^{-1}\mathbf{J_1} - \mathbf{J_2} \qquad (9)\end{aligned}$$

The above simplified technique reduces computation dramatically over the direct computation based on equation (6), as L_{22} and L_{11} used in equation (9) are already triangular.

2.6 Analysis of the Computation Cost

In this section we present the computational advantage of macromodeling over the flat model analysis approach.

Suppose the cost of factorizing a matrix is $C_1(l)$, and the cost of one forward and one backward substitution is $C_2(l)$, where l is the size of the matrix, and $C_2(l) << C_1(l)$. Let N be the number of nodes in the entire power network.

If no macromodels are used for the power analysis, the computation cost of the first run is $C_1(N)$ and the computation cost of a subsequent run is $C_2(N)$. In the macromodeling approach, the computation cost of the first run can be expressed as

$$\begin{aligned}C_1(n_1) + C_1(n_2) + \ldots + C_1(n_k) + C_1(n_0 + m_0 + m_1 + \ldots + m_k) \\ + C_2(n_1) + C_2(n_2) + \ldots + C_2(n_k) \qquad (10)\end{aligned}$$

Here, $n_i, i \in [1,k]$ is number of nodes in each partition, n_0 and m_i are defined in Section 2.3, and $n_0 + n_1 + n_2 + \ldots + n_k = N$. The computation cost from macromodeling is given by $C_1(n_1) + C_1(n_2) + \ldots + C_1(n_k)$ by using the simplified macromodeling method described in section 2.5. The cost of finding the solution to the global network is $C_1(n_0 + m_0 + m_1 + \ldots + m_k)$ and the cost of solving the local grids is $C_2(n_1) + C_2(n_2) + \ldots + C_2(n_k)$, since the factors obtained from macromodeling can be used for solving the local grid.

The computation cost of each subsequent run can therefore be approximated as

$$C_2(n_0 + m_0 + m_1 + \ldots + m_k) + 2C_2(n_1) + 2C_2(n_2) + \ldots + 2C_2(n_k) \qquad (11)$$

where the computation cost of macromodeling is $C_2(n_1) + C_2(n_2) + \ldots + C_2(n_k)$ since the A_i's are unchanged and only the $\mathbf{S_i}$'s must be recalculated during the subsequent run in macromodeling.

Expressions (10) and (11) provide a rough estimate of computation costs based on the size of the network and its partitions. In reality, the density of a matrix is an important factor that influences the solution speed. Generally, the conductance matrices for partitions are denser than the conductance matrix of the entire network, and thus the conductance matrix in equation (8) used for the global solution is a dense matrix.

Typically, Cholesky factorization requires $n^3/6$ multiplications and substitution requires $n^2/2$ multiplications. However the sparsity of the conductance matrix, combined with efficient reordering, enables the observed computation cost to be near-linear with the dimension of the matrix. However, with an increase in the size of the conductance matrix, the computation cost will approach the $n^3/6$ or $n^2/2$ curve gradually. In such cases, the computation cost for the macromodel approach will be lower than that for the flat analysis even with the overheads associated with partitioning.

Most important of all, the *divide and conquer* procedure applied to the power network makes parallel execution of power network simulation possible. During parallel execution, the execution time of the first run is given by

$$\begin{aligned}\max(C_1(n_1), C_1(n_2), \ldots, C_1(n_k)) + C_1(n_0 + m_0 + m_1 + \ldots + m_k) \\ + \max(C_2(n_1), C_2(n_2), \ldots, C_2(n_k))\end{aligned}$$

where $C_1(n_0 + m_0 + m_1 + \ldots + m_k)$ is the global solution time, $\max(C_1(n_1), C_1(n_2), \ldots, C_1(n_k))$ represents the maximum execution time among macromodeling of partitions and $\max(C_2(n_1), C_2(n_2), \ldots, C_2(n_k))$ represents the maximal execution time out of partition solutions. Similarly, the execution time of the subsequent run is given by

$$C_2(n_0 + m_0 + m_1 + \ldots + m_k) + 2 \times \max(C_2(n_1), C_2(n_2), \ldots, C_2(n_k))$$

Moreover, the memory requirement with macromodels is the maximum memory required for solving any partition, rather than the sum of memory requirement of each partition.

Besides run time and memory advantage, macromodeling provides a certain flexibility to a design/analysis situation so that significant analysis effort can be saved. Given below are few examples of design/analysis situations when such flexibility is useful.

Example-1: When a designer is interested in the detailed analysis only of a specific circuit block, then significant design time is saved by not simulating the other partitions, but while accounting accurately the effect of switching of these other blocks on the block s/he is interested in.

Example-2: A designer knows *a priori* in which circuit block or blocks the worst drop is to be expected, and the objective of the analysis is to only to find the worst IR drop estimate for the design. Then, it will be necessary to simulate only few blocks (partitions) in the last step of the macromodel approach.

Example-3: The process of fixing problems in a power grid is usually an iterative one. The process consists of detecting an error, making local changes to the grid to correct the problem, and rerunning the analysis. In this case, only the macromodeling of the partition whose grid was changed needs to be recalculated. The speed-up in analysis due to this makes it possible for the designer to fix the problems interactively with the analysis tool.

3 Sparsification of Macromodels

In section 2.4, we pointed out that the number of entries in the macromodel has $O(m^2)$ complexity for model size m. Although the macromodels reduce the size of the system to the smaller system described in equation (5), the sparsity of the coefficient matrix of equation (5) decreases considerably due to the density of the A_i submatrices. For an iterative solver, this is undesirable as the number of floating point operations (FLOPs) to solve the system increases. For a direct solver, this affects both the FLOPs, as well as the memory required to factorize. The additional memory demand is caused by excessive fills created by the dense parts during factoring. So, to derive the most benefit out of the macromodeling approach, it is important that the coefficient matrix in equation (5) is kept sparse. While the partitioning strategy explained in section 2.4 is a natural way of achieving this, other sparsification techniques in conjunction with good partitioning schemes are very useful for making the macromodeling approach practical. In this section, we present a novel technique to sparsify the port admittance matrices of the macromodels.

Our sparsification method is motivated by the observation that although the matrix A is dense, it consists of a large number of values that are numerically small and will have little influence on the results if approximated to zero. We provide an algorithm to sparsify the coefficient matrix A by dropping some of its entries, while keeping the error introduced by this process below a specified value. The proposed sparsification technique also preserves the symmetry and the positive definite property of the matrix. Note that the sparsification procedure needs to be performed only once (during the first run).

3.1 Problem Definition

The problem is stated as follows: Given the transfer characteristic equation of each partition

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ \vdots \\ i_m \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,m} \\ a_{3,1} & a_{3,2} & a_{3,3} & \ldots & a_{3,m} \\ \vdots & & & & \\ a_{m,1} & a_{m,2} & a_{m,3} & \ldots & a_{m,m} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_m \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_m \end{bmatrix}, \qquad (12)$$

the nominal value of v_j, $j \in [1,m]$: B, $B > 0$, and

the error in i_j, $j \in [1,m]$: e_j

Table 1: Run-time and memory comparison for the first simulation

Chip	#nodes (millions)	Without macromodel		With macromodel				
		Run-time (minutes)	Peak Memory (GB)	# part	#nodes(max) (millions)	Total Run-time Serial(min)	Total Run-time Parallel(min)	Peak Memory (GB)
Chip-1	3.9	93	1.5	12	0.40	43	7	0.2
Chip-2	2.7	57	1.2	9	0.58	25	6	0.3
Chip-3	7.5	629	2.6	11	0.79	136	26	0.4
Chip-4	20.0	-	-	7	3.5	444	152	1.3

transform equation (12) into

$$\begin{bmatrix} i'_1 \\ i'_2 \\ i'_3 \\ \vdots \\ i'_m \end{bmatrix} = \begin{bmatrix} a'_{1,1} & a'_{1,2} & a'_{1,3} & \dots & a'_{1,m} \\ a'_{2,1} & a'_{2,2} & a'_{2,3} & \dots & a'_{2,m} \\ a'_{3,1} & a'_{3,2} & a'_{3,3} & \dots & a'_{3,m} \\ \vdots & & & & \\ a'_{m,1} & a'_{m,2} & a'_{m,3} & \dots & a'_{m,m} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_m \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_m \end{bmatrix} \quad (13)$$

to maximize

$$\text{the number of } a'_{j,k}, j \neq k, \text{ such that } a'_{j,k} = 0,$$

subject to

$$| i_{j\prime} - i_j | \leq e_j, \ j \in [1, m]$$
$$a'_{j,k} = a'_{k,j} \text{ (maintaining matrix symmetry).}$$

3.2 Problem Formulation

This problem can be formulated into a 0-1 multidimensional knapsack problem [10, 11]. In this section, we describe the transformation from the above problem to the knapsack problem.

The task here involves zeroing out off-diagonal elements of the matrix A. It is easy to show that these sparsification operations maintain the positive definite property of the matrix. To see this, we note first that the partition can be thought of as being purely resistive (for example, any capacitors are linearized). Given this "resistive" network, one may build an equivalent network of a set of equivalent resistances R_{jk} between each pair of ports j and k. The matrix A is then simply the conductance matrix for this network of R_{jk}'s, and is therefore diagonally dominant. This leads to two conclusions: (1) all off-diagonal elements must be nonpositive, and (2) zeroing out off-diagonal elements of A maintains the diagonal dominance of the matrix, and therefore its positive definite property.

The problem formulation is described as follows. First consider the maximum error that an element of matrix $a_{j,k}$ can cause if it is rounded off to 0. Since B is positive and $a_{j,k} \leq 0, j \neq k$, the maximum negative error caused by rounding off $a_{j,k}$, $en_{j,k}$, is given by

$$en_{j,k} = a_{j,k} * B, j \neq k$$

Let $X_{j,k}$ represent a Boolean value, 1 when element $a_{j,k}$ is rounded to zero, and 0 otherwise. The matrix sparsification problem can be formulated as 0-1 knapsack problem as follows.

$$\begin{aligned} \text{Maximize} \quad & z(x) = \textstyle\sum_{k=1}^{m} \sum_{j=1}^{k} X_{j,k} \\ \text{subject to} \quad & -\textstyle\sum_{k=1}^{j-1} en_{j,k} * X_{j,i} - \sum_{k=j+1}^{m} en_{j,k} * X_{j,k} \leq e_j, j \in [1, m] \\ & X_{j,k} \in \{0, 1\} \text{ for all } X_{j,k}, \ j < k \end{aligned} \quad (14)$$

In (14), the indices of the variables $X_{j,k}$ are required to satisfy the relation, $j < k$, so that $X_{j,k} = 1$ indicates the rounding-off of both $a_{j,k}$ and $a_{k,j}$ to maintain the symmetricity of A. Therefore, the resulting sparsified matrix is symmetric and positive definite.

The 0-1 knapsack problem can be solved optimally either by dynamic programming or using an ILP solver. In our implementation, we use the latter, but with some modifications for speed considerations. First, we relax the integer requirement and solve the fractional knapsack problem using a linear programming solver [12]. Next, the fractional x_{jk}'s are sorted, and applied successively until the maximum error in $|i_j|$ reaches the specified limit, e_j.

4 Experimental Results

The hierarchical analysis method using macromodels was implemented using C and embedded in an existing in-house power analysis tool [1]. An efficient direct linear solver based on Cholesky factors was used in all the experiments. The extracted power grids of four high performance general purpose/DSP microprocessor chips were used to benchmark the performance of macromodeling (Tables 1 and 2) and sparsification (Table 3) techniques. Chips 1, 2 and 4 are DSP and communication chips whose power grids are implemented in 3 layers of metal. Chips 3 and 5 are high performance microprocessor chips using 5 and 6 metal layers respectively. The analyses were run on Sun workstations. The run time measures used for comparison are based on the actual time required to complete the task.

4.1 Performance of Macromodeling Technique

Table 1 compares the performance of the proposed hierarchical approach using macromodels with that of the nonhierarchical approach. Two metrics are compared - the peak memory demand and the total run-time. The number of nodes, in millions, for the entire power network is given in column 2, the number of partitions used by the macromodeling algorithm are listed in Column 5, and the number of nodes in the largest partition is given in column 5. Column 3 shows the total time, in minutes, taken for completing the analysis on the flat model, while columns 7 and 8 show the total time required by the hierarchical approach. The run-time in column 7 corresponds to the cases when the macromodels for the various partitions were generated serially on a single computer, whereas column 8 is for the cases when these computations are performed in parallel. The run-time reported in this table is the time taken for analyzing the power network at the first time point in a sequence of simulations. Columns 4 and 9 show the peak memory demand, in Gigabytes, during the analysis without macromodels and with macromodels, respectively. Chip-4 can not be solved without macromodeling due to its large size.

It is evident from the above table that the problem size tackled with the proposed approach is substantially reduced from the original problem. This is the primary goal of the proposed approach so that a chip-level analysis of very large designs is made possible. Based on the benchmarks, it can be seen that the size of the linear system that needs to solved with the new approach is about 10X smaller than the traditional approach.

The effect of problem size reduction is clearly reflected in the peak memory requirements of the different approaches shown in the table. Again, a 10X to 20X reduction in memory requirement is seen possible with the hierarchical approach. This implies that with the available computing resources (memory and speed), the new method enables the analysis of much larger designs that will become common in the near future. From the results, we can see that without macromodels the run time can be several hours (e.g., 10.5 hours for Chip 3) for a supply network with millions of nodes. As a result of reducing the size complexity, the run-time is reduced by a factor of 2X to 5X even when the macromodels are computed one after another on a single computer. The run-time is dramatically reduced by 10X to 23X, if the parallelism created by the macromodel is utilized. It is noteworthy that the speedups improve with the size of the circuit under consideration.

It should be noted that the performance of hierarchical approach reported in Table 1 does not consider the additional performance gain resulting from the proposed sparsification technique.

Table 2 compares the performance of the two approaches based

Table 2: Comparison of run time for 1000 subsequent simulations

Chip	Without macromodel run-time (hours)	With macromodel run-time	
		Total(hours)	Parallel(hours)
Chip-1	8.4	28.0	4.0
Chip-2	8.0	22.5	4.4
Chip-3	33.7	43.5	6.6

on the time required to perform simulations at 1000 successive time points, after the first one. Thus, the run-times here are independent of the time taken to generate the macromodels. Column 2 shows the run time without macromodels. For the hierarchical approach, run-times for both serial (column 3) and parallel (column 4) execution are shown. Since the memory requirement of these runs is less than that of the first run, these figures are omitted in Table 2.

The hierarchical approach executed in serial mode recorded unfavorable run-times for the benchmarks. However, the disparity in run-times between the nonhierarchical and hierarchical approach (in serial mode) diminishes as the size of the original network becomes larger, as evidenced from the results for Chip-3, which has 7.5 million nodes. This behavior is not unexpected, and can be explained by the fact that the overhead associated with computing the **S** vector for each partition at every time step, and back-solving each partition again in the final step of the solution, is a dominant factor. This behavior is exhibited for networks up-to a certain size, where the original matrix and the reduced matrix do not differ greatly in terms of the time required for a back-solve. However, as the network becomes larger, the difference in problem sizes with and without macromodels are significantly different, and the overhead cost of handling the partitions becomes negligible in the overall cost. As a result, the hierarchical approach becomes favorable for very large networks even in the serial execution mode.

The run-time advantage of parallel execution mode is very clear from Table 2. Results show that the parallel execution utilizing hierarchy is $1.8-5.1$ times faster than the nonhierarchical approach. As designers would like to simulate the power grid with long traces of current signatures in order to obtain good coverage of the IR-drop situations, efficiency of simulation in this phase is crucial. The parallel execution mode, as well as the flexibility in the hierarchical analysis discussed in section 2.6, make the hierarchical analysis approach extremely attractive.

4.2 Performance of the Sparsification Technique

The sparsification procedure described in section 3 reduces the number of nonzero elements while maintaining an acceptable level of accuracy. In our implementation, the specified error e_j is defined as $e_j = \max(const, |\, s_j \times x\% \,|)$, where *const* is a small positive constant, $s_j, 1 \leq j \leq m$, is as defined in equation (12), and $x\%$ is the user-defined error limit, which is typically $0\% - 10\%$. The sparsification technique was implemented using a linear programming solver *lp_solve_2.3* [12].

Table 3 reports the sparsity and run-time improvements achieved for two benchmark examples, analyzed at different levels of accuracy. The second column in the table shows the voltage value of the clean power supply and the value of the maximum voltage drop observed in the circuit. Columns 3 and 4 report the number of nodes and the total number of ports respectively in the global grid. The number of nonzero elements in the coefficient matrix of equation (5) are shown in Column 5. Column 6 shows the maximum voltage error caused by the sparsification procedure. The ratio of maximum observed error in voltage to the maximum voltage drop is shown in column 7. Finally, column 8 reports the time required to solving equation (5).

For each benchmark, the proposed sparsification technique was tested at four levels of accuracy. The benchmark Chip-5 is a 6-layer, mesh type, power grid. Its power grid is much denser than the other examples, and this example also has some partitions with large number of ports. As a result, the coefficient matrix obtained for this example could not be solved with the available computing resources without sparsification.

The results clearly show that the sparsity of the coefficient matrix is improved by as much as 11X, incurring only 2.6% error in the final results. The improved sparsity improved the run-time for the dense example, Chip-5 significantly, besides greatly reducing the memory requirement.

Table 3: The effect of sparsification

Chip	Clean/ Max-drop(v)	# nodes	# ports	#non-zeroes	Max-err (v)	Error %	Run-time (secs)
Chip -3	2.0/0.12	23261	379	106909	0.0	0.0%	38.3
				98505	0.000043	0.04%	36.0
				98253	0.00058	0.5%	32.4
				98005	0.0013	1%	30.6
Chip -5	1.8/0.02	2932	2849	2067572	0.0	0.0%	-
				366612	0.000045	0.2%	36.5
				249588	0.00021	1.0%	24.4
				197868	0.00051	2.6%	20.1

5 Conclusions and Future Work

In this paper, we have presented a hierarchical power network analysis method using novel macromodeling and matrix sparsification techniques. The proposed techniques were shown to gain significant memory and run-time advantages over the traditional approach of analyzing the power network without using the hierarchy. The experimental results based on analyzing the entire power network of four high performance microprocessor designs confirmed these claims. The hierarchical analysis approach shows excellent promise as a viable alternative to the traditional nonhierarchical analysis method, capable of handling the increasing size of power grids in modern microprocessors.

It was shown that the method of partitioning has significant influence on the performance of this approach. One of our future directions is to explore optimal partitioning techniques that can be applied with minimal user-intervention. Another research direction is to develop more efficient sparsification techniques.

References

[1] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, "Design and analysis of power distribution networks in PowerPC microprocessors," in *DAC*, pp. 738–743, 1998.

[2] G. Steele, D. Overhauser, S. Rochel, and Z. Hussain, "Full-chip verification methods for DSM power distribution systems," in *DAC*, pp. 744–749, 1998.

[3] H. Chen and D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *DAC*, pp. 638–643, 1997.

[4] S. Taylor, "The challenge of designing global signals in UDSM CMOS," in *CICC*, pp. 429–435, 1999.

[5] H. Kriplani, F. Najm, and I. Hajj, "Pattern independent minimum current estimation in power and ground buses of CMOS VLSI circuits," *TCAD*, vol. 14, no. 8, pp. 998–1012, 1995.

[6] A. Krstic and K. Cheng, "Vector generation for maximum instantaneous current through supply lines for CMOS circuits," in *DAC*, pp. 383–388, 1997.

[7] Y.-M. Jiang, T. Young, and K. Cheng, "VIP – an input pattern generator for identifying critical voltage drop for deep submicron designs," in *ISLPED*, pp. 156–161, 1999.

[8] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore: The Johns Hopkins University Press, 1984.

[9] C. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, no. 6, pp. 504–509, 1975.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.

[11] K. G. Murty, *Operations Research Deterministic Optimization Models*. New Jersey: Prentice Hall, Englewood Cliffs, 1995.

[12] M. R. C. M. Berkelaar, "LP SOLVE 2.3 Users' Manual," 1998.