# Power Grid Analysis Using Random Walks

Haifeng Qian, Sani R. Nassif, *Senior Member, IEEE*, and Sachin S. Sapatnekar, *Fellow, IEEE*

***Abstract*—This paper presents a class of power grid analyzers based on a random-walk technique. A generic algorithm is first demonstrated for dc analysis, with linear runtime and the desirable property of localizing computation. Next, by combining this generic analyzer with a divide-and-conquer strategy, a single-level hierarchical method is built and extended to multilevel and "virtual-layer" hierarchy. Experimental results show that these algorithms not only achieve speedups over the generic random-walk method, but also are more robust in solving various types of industrial circuits. Finally, capacitors and inductors are incorporated into the framework, and it is shown that transient analysis can be carried out efficiently. For example, dc analysis of a 71 K-node power grid with C4 pads takes 4.16 s; a 348 K-node wire-bond dc power grid is solved in 93.64 s; transient analysis of a 642 K-node power grid takes 2.1 s per timestep.**

***Index Terms*—Capacitance, inductance, physical design, power grid, random walk, simulation, supply network.**

## I. Introduction

POWER grid analysis is an indispensable step in high-performance very large scale integrated circuit (VLSI) design. In successive technology generations, the $V_{DD}$ voltage decreases, resulting in narrower noise margins. Meanwhile, IR drops on power grids become worse as wire resistances increase due to the reduced interconnect wire widths, and the currents through the grid increase. Since power grids play an important role in determining circuit performance, it is critical to analyze them accurately and efficiently to check for signal integrity.

A typical power grid may be represented by the model in Fig. 1, consisting of wire resistances, wire inductances, wire capacitances, decoupling capacitors, $V_{DD}$ pads, and current sources that correspond to the currents drawn by logic gates or functional blocks. There are two subproblems to power grid analysis: *dc analysis* to find steady-state node voltages, and *transient analysis*, which is concerned with finding voltage waveforms considering effects of capacitors, inductors, and time-varying current waveform patterns.

The dc analysis problem is formulated as

$$G\mathbf{V} = \mathbf{E} \tag{1}$$

where $G$ is the conductance matrix for the interconnected resistors, $\mathbf{V}$ is the vector of node voltages, and $\mathbf{E}$ is a vector of independent sources. Traditional approaches exploit the sparse and positive definite nature of $G$ to solve this system of linear equations for $\mathbf{V}$. However, the cost of doing so can become prohibitive for a modern-day power grid with hundreds of millions of nodes, and this will only become worse as the circuit size is ever growing from one technology generation to the next. Furthermore, in recent technologies, inductive effects in the top few metal layers can no longer be ignored. The transient analysis problem involves the solution of an equation similar to (1) at each time point in the analysis. Especially when mutual inductances are taken into consideration, the left-hand-side matrix, which contains the contribution of capacitors and inductors, is significantly denser than that for dc analysis, making it even more expensive even at a single time point.

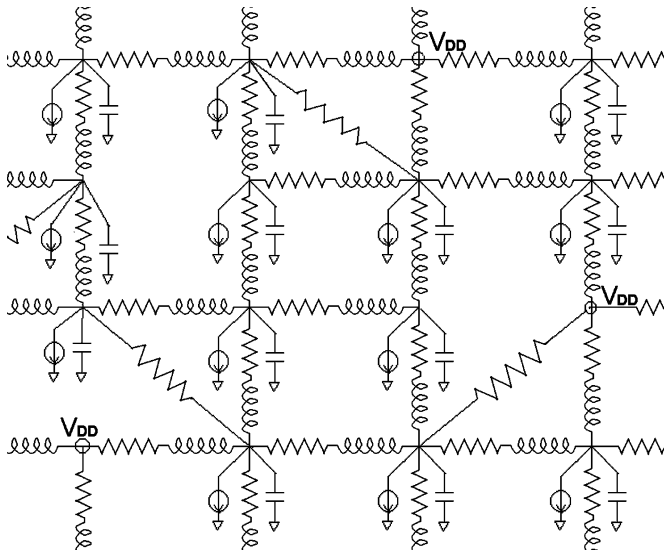



Fig. 1. Part of a typical power grid model.

Different circuit models and simulation techniques have been developed for power grid analysis, to handle large problem size, and to incorporate capacitances and inductances efficiently [2], [4], [5], [8], [16], [19], [23], [29], [30], [33], [34], [39]. Among them, several methods are proposed to achieve a lower time and space computational complexity by sacrificing a certain degree of accuracy. For example, [19] proposes a grid-reduction scheme to coarsen the circuit recursively, solves a coarsened circuit, and then maps back to find the solution to the original circuit. The approach in [39] utilizes the hierarchical structure of a power grid, divides it into a global grid and multiple local grids, and solves them separately.

In this paper, we apply a statistical approach based on random walks to solve the problem of power grid analysis. Random walks correspond to a classical problem in statistics, and their use in solving linear equations dates back to as early as [10] and [37]. Subsequently, several other solvers have been developed

Manuscript received November 29, 2003; revised October 8, 2004. This work was supported in part by the Semiconductor Research Corporation (SRC) under Contract 2003-TJ-1092, by the National Science Foundation (NSF) under award CCR-0205227, and by an IBM Faculty Award.

H. Qian and S. S. Sapatnekar are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: qianhf@ece.umn.edu; sachin@ece.umn.edu).

S. R. Nassif is with the IBM Austin Research Laboratory, Austin, TX 78758 USA (e-mail: nassif@us.ibm.com).

Digital Object Identifier 10.1109/TCAD.2005.850863

[12], [32], [35]. The work in this paper is inspired by [9]. A brief overview of these works is provided in Section II-A.

This paper is organized as follows. Based on the above mathematical foundation, we apply random walks for power grid analysis and develop a basic dc analysis algorithm in Section II that we call the generic method. Next, in Section III, we combine the divide-and-conquer idea of [39] with this generic random-walk method and present a hierarchical random-walk method. These algorithms are extended to handle RKC transient analysis (where $K$ is the inverse inductance, or susceptance, matrix) with capacitors and inductors in Section IV. We use test results to show that the proposed algorithms provide good accuracy–runtime tradeoffs and are faster than traditional methods with acceptable error levels in Section V. We demonstrate that the proposed algorithms have the feature of localizing computation, which makes them especially useful when only part of the grid is to be analyzed. Finally, we present concluding remarks in Section VI. This paper is an extended description of the work in [25] and [26].

## II. The Generic Random-Walk Method

This section focuses on dc analysis only and is organized as follows. Section II-A provides a summary of previous works that use random walks to solve linear equations, Section II-B presents the theoretical basis of the generic algorithm, followed by a simple illustrative example in Section II-C. The time complexity and limitations of the generic method are discussed in Section II-D and Section II-E, respectively.

### A. A Brief History

A random walk, also viewed as a discrete abstraction of the physical phenomenon of Brownian motion, is one category of the general Monte Carlo methods of numerical computation. In this paper, we employ this method to solve systems of linear equations that are diagonally dominant. Historically, the theory that underlies this work was developed on two seemingly independent tracks, related to the analysis of potential theory [6], [12], [13], [17], [18], [22] and to the solution of systems of linear equations [10], [12], [32], [35], [37]. However, the two applications are closely related, and research on these tracks has resulted in the development of analogous algorithms, some of which are equivalent. These mathematical works have found meaningful applications in electrical engineering [1], [3], [20], [28].

Along the first track, the goal has been to solve Laplace's equation in a closed region with given boundary values (i.e., under Dirichlet conditions), and it was proven that the value at a location can be estimated by observing a number of Brownian particles that start from this location and travel until they hit the boundary, and taking the average of the boundary values at the end points [6], [13], [18]. An important improvement was proposed in [22], which proved that, instead of simulating tiny movements of a Brownian particle, the particle can leap from a location to a random point on a sphere that is centered at this location, and that shapes other than a sphere can be used, given the corresponding Green function.[1] Another important development [17] extended the theory to solving Poisson's equation under Dirichlet conditions and more general elliptic differential equations (under certain restrictions).

The second parallel track, which considered the solution of systems of linear equations, will be discussed in greater detail here, since it is directly related to our algorithm. The first work that proposed a random-walk-based linear solver is [10], although it was presented as a solitaire game of drawing balls from urns. It was proven in [10] that, for any matrix $G$ such that $\max_r |\lambda_r(I - G)| < 1$, where $\lambda_r$ denotes the $r^{\text{th}}$ eigenvalue of a matrix, a game can be constructed and a random variable[2] $X$ can be defined such that $E[X] = (G^{-1})_{ij}$, where $(G^{-1})_{ij}$ is an entry of the inverse matrix of $G$. In [10], the variable $X$ is a "payment" when exiting the game. Under certain settings, the algorithm of [10] is equivalent to the "home" and "award" concepts in our theory, which is presented in the next section.

Two years later, the work in [37] continued this discussion in the formulation of random walks and proposed the use of another random variable[3] $Y$ to replace $X$. A "mass" value was defined for every step in a walk, and $Y$ was defined as the total amount of "mass" carried through a walk. It was proven in [37] that $E[Y] = E[X]$, and it was argued that, in certain special cases, $Y$ has a lower variance than $X$, and hence is likely to converge faster. Under certain settings, the algorithm of [37] is equivalent to the "motel" concept in our method.

Both [10] and [37] have the advantage of being able to compute part of an inverse matrix without solving the whole system, in other words, localizing computation. Over the years, various descendant stochastic solvers have been developed [12], [32], [35]. Some of them, e.g., [32] and [35], do not have the property of localizing computation.

From a different perspective, the work in [9] aimed at investigating random walks by using electrics. It drew a parallel between resistive networks and random walks and interpreted the relationship between conductances and probabilities. With underlying rules similar to [10], [9] proved many insightful conclusions linking statistics and electrics.

In summary, the theory of our proposed generic random-walk algorithm is directly inspired by [9] but is mathematically a combination of [10] and [37], and it inherits the property of localizing computation. Not surprisingly, in potential theory, there is a method that can be viewed as roughly parallel to our basic framework: the counterpart is [17]. Besides these legacies, our algorithm also includes efficiency-improving techniques, which are not seen in previous works, and which play a crucial role in obtaining a performance that is practically useful.

### B. Principles

We will focus our discussion on the analysis of a $V_{DD}$ grid, pointing out the difference for a ground grid where applicable. For the dc analysis of a power grid, let us look at a single node $x$ in the circuit, as illustrated in Fig. 2. Applying Kirchoff's

[1]Many years later, this evolved to [20], a successful Monte Carlo algorithm in VLSI design automation.

[2]The notation that is used in [10] for $X$ is $G$, but we have changed the notation since we use $G$ to signify another quantity in our discussion.

[3]Again, the notation is changed for clarity: [37] referred to this as $M$.

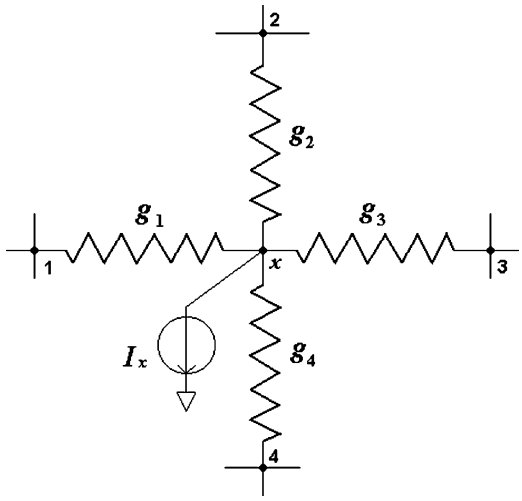



Fig. 2. Representative node in the power grid.

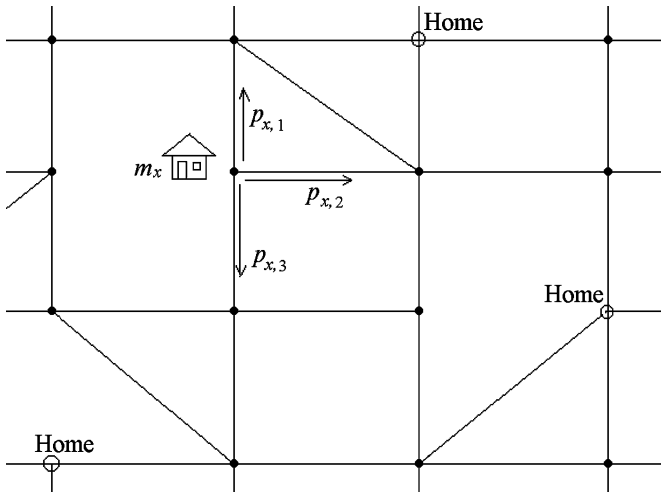



Fig. 3. Instance of a random-walk "game."

Current Law, Kirchoff's Voltage Law, and the device equations for the conductances, we have

$$\sum_{i=1}^{\text{degree}(x)} g_i(V_i - V_x) = I_x \tag{2}$$

where the nodes adjacent to $x$ are labeled $1, 2, \cdots, \text{degree}(x)$, $V_x$ is the voltage at node $x$, $V_i$ is the voltage at node $i$, $g_i$ is the conductance between node $i$ and node $x$, and $I_x$ is the current load connected to node $x$. Equation (2) can be rewritten as

$$V_x = \sum_{i=1}^{\text{degree}(x)} \frac{g_i}{\sum_{j=1}^{\text{degree}(x)} g_j} V_i - \frac{I_x}{\sum_{j=1}^{\text{degree}(x)} g_j}. \tag{3}$$

We can see that this implies that the voltage at any node is a linear function of the voltages at its neighbors. We also observe that the sum of the linear coefficients associated with the $V_i$'s is 1. For a power grid problem with $N$ non-$V_{DD}$ nodes, we have $N$ linear equations similar to the one above, one for each node. Solving this set of equations gives the exact solution.

Now let us look at a random walk "game," given a finite undirected connected graph (for example, Fig. 3) representing a street map. A walker starts from one of the nodes and goes to an adjacent node $i$ every day with probability $p_{x,i}$ for $i = 1, 2, \cdots, \text{degree}(x)$, where $x$ is the current node, and $\text{degree}(x)$ is the number of edges connected to node $x$. These probabilities satisfy the following relationship:

$$\sum_{i=1}^{\text{degree}(x)} p_{x,i} = 1 \tag{4}$$

The walker pays an amount $m_x$ to a motel for lodging everyday until he/she reaches one of the homes, which are a subset of the nodes. If the walker reaches home, he/she will stay there and be awarded a certain amount of money $m_0$. We will consider the problem of calculating the expected amount of money that the walker has accumulated at the end of the walk, as a function of the starting node, assuming he/she starts with nothing. This gain function is therefore defined as

$$f(x) = E[\text{total money earned} \,|\text{walk starts at node } x]. \tag{5}$$

It is obvious that

$$f(\text{one of the homes}) = m_0. \tag{6}$$

For a nonhome node $x$, assuming that the nodes adjacent to $x$ are labeled $1, 2, \cdots, \text{degree}(x)$, the $f$ variables satisfy

$$f(x) = \sum_{i=1}^{\text{degree}(x)} p_{x,i} f(i) - m_x. \tag{7}$$

For a random-walk problem with $N$ nonhome nodes, there are $N$ linear equations similar to the one above, and solving this set of equations gives the exact values of $f$ at all nodes.

It is easy to draw a parallel between this problem and power grid analysis. Equation (7) becomes identical to (3), and (6) reduces to the condition of perfect $V_{DD}$ nodes if

$$p_{x,i} = \frac{g_i}{\sum_{j=1}^{\text{degree}(x)} g_j} \quad i = 1, 2, \cdots, \text{degree}(x)$$
$$m_x = \frac{I_x}{\sum_{j=1}^{\text{degree}(x)} g_j} \quad m_0 = V_{DD} \quad f(x) = V_x. \tag{8}$$

The formulation for ground net analysis is analogous; the major differences are that 1) the $I_x$'s have negative values and 2) $V_{DD}$ is replaced by zero. As a result, the walker earns money in each step but gets no award at home.

In other words, for any power grid problem, we can construct a random-walk problem that is mathematically equivalent, i.e., characterized by the same set of equations. It can be proven that such an equation set has, and only has, one unique solution [9]. It is both the solution to the random-walk problem and the solution to the power grid problem. Therefore, if we find an approximated solution for the random walk, it is also an approximated solution for the power grid.

A natural way to approach the random-walk problem is to perform a certain number of experiments and use the average money left in those experiments as the approximated solution. If this amount is averaged over a sufficiently large number of walks by playing the "game" a sufficient number of times, then by the law of large numbers [38], an acceptably accurate solution can be obtained. This is the idea behind the proposed *generic algorithm* that forms the most basic implementation.

According to the Central Limit Theorem [38], the error is a 0-mean Gaussian variable with variance inversely proportional to $M$, where $M$ is the number of experiments. Thus, we have an accuracy–runtime tradeoff. Instead of fixing $M$, we use a stopping criterion driven by a user-specified error margin $\Delta$:

$$P[-\Delta < V_e - V < \Delta] > 99\% \tag{9}$$

where $V_e$ is the estimated voltage from $M$ experiments. If the variance of these results is $Var$, the above criterion becomes

$$\begin{aligned} Q\left(\frac{\Delta}{\sqrt{\frac{Var}{M}}}\right) &< 0.005 \\ \frac{Var}{M} &< \left(\frac{\Delta}{Q^{-1}(0.005)}\right)^2 \end{aligned} \tag{10}$$

where $Q$ is the standard normal complementary cumulative distribution function, defined as

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{u^2}{2}} du.$$

According to condition (10), $M$ is decided on the run and has different values for different nodes. It is worth noting that for each node, for a fixed confidence level, $M \propto 1/\Delta^2$.

In (9), (10), and in our implementation, the 99% confidence level is used for illustrative purpose. In practice, the stopping criterion (9) can be adaptive to different node voltages: for a node with high estimated voltage drop, i.e., a dangerous node, we can switch the criterion to a higher confidence level, a lower $\Delta$, or both; for a node with low estimated voltage drop, i.e., a safe node, the computation stops after satisfying a relaxed criterion with lower confidence level or larger $\Delta$. In other words, the computation for a node voltage starts with a low-accuracy criterion; when this accuracy level is met, a decision is made based on the estimated voltage drop at this time: if this value is below a certain threshold, the computation stops; otherwise, the algorithm switches to a higher accuracy criterion and continues; when the new accuracy level is met, another decision is made based on the new estimated voltage drop, and even higher accuracy can be used if necessary, and so on. Using this adaptive strategy, more runtime would be spent on potential failure nodes, to get more accurate voltages, while safe nodes only get coarse estimation.

A desirable feature of the proposed algorithm is that it localizes the computation, i.e., it can calculate a single node voltage without having to solve the whole circuit. This is especially meaningful when the designer knows which part of the power grid is problematic, or when the designer makes a minor change in the design and wishes to see the impact. For example, if the objective of the analysis is to find the voltage at a single node, then this approach can perform a number of random walks starting from that node. In a typical power grid that has a sufficient number of pads that are reasonably close to any node, such a walk is likely to terminate soon at a home. As compared with a conventional approach that must solve the full set of matrix equations to find the voltage at any one node, the computational advantage of this method could be tremendous, and we validate this in Section V.

When solving for multiple node voltages, an efficiency-enhancing technique can be used. Since the voltage at each already calculated node is known, it becomes a new home in the game with an award amount equal to its calculated voltage. In other words, any later random walk that reaches such a node terminates and is rewarded a money amount equal to the calculated voltage. This operation speeds up the algorithm dramatically, as there are more terminals to end a walk, and therefore the average number of steps in each walk is reduced. At the same time, this operation improves accuracy without increasing $M$, because each experiment that ends at such a node is equivalent to multiple experiments. A cost of this speedup is that the error at a calculated node also affects later computation, in other words, this speedup technique is not 100% positive, but another accuracy-runtime tradeoff. Practically, it is such a good deal that we can almost ignore the cost: errors tend to cancel each other, and the impact on accuracy is minor, while the speedup is dramatic.

Due to this speedup technique, the nodes computed early in the algorithm and those computed late are treated differently. For the first node, random walks are carried out in the original game where home nodes correspond to voltage sources only. As more and more nodes are calculated, they all become new homes in the game, and random walks from later nodes are carried out in a game with a larger and larger number of homes. Therefore, the ordering of nodes could potentially affect the performance. In the implementation, we use the read-in ordering without any processing, which is close to a random ordering; a truly random ordering can be easily obtained by permuting the read-in ordering if necessary. We prefer a random ordering, because as computation proceeds, the density of home nodes is increased evenly throughout the whole circuit, and the performance of the algorithm is stable. Finally, we want to point out that, stopping criterion with the same error margin $\Delta$ is applied to all nodes, regardless of their positions in the ordering, and that we only need to compute nodes that are of interest, which, in the context of the generic algorithm, refer to the bottom-metal-layer nodes only.

### C. A Simple Example

In order to show how the proposed algorithm works, let us look at a simple circuit, as shown in Fig. 4. The true voltages at node A, B, C, and D are 0.6, 0.8, 0.7, and 0.9, respectively. Applying (8) to this circuit, we construct an equivalent random-walk game, as shown in Fig. 5, where numbers inside circles represent motel prices and home awards, and numbers beside the arrows represent the transition probabilities from each node to a neighboring node.

To find out the voltage of node A, we start the walker at node A with zero balance. He/she pays the motel price of \$0.2, then

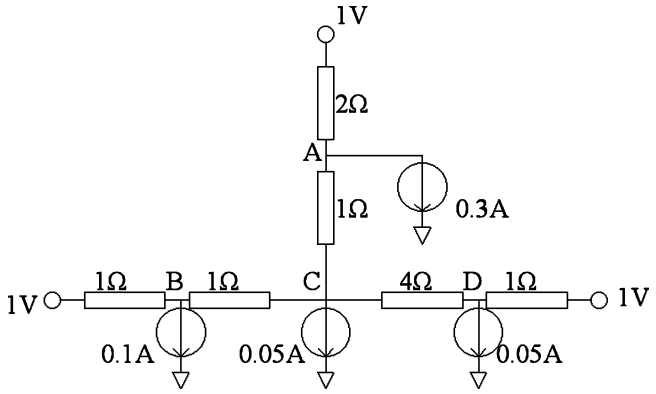



Fig. 4. Simple circuit example.

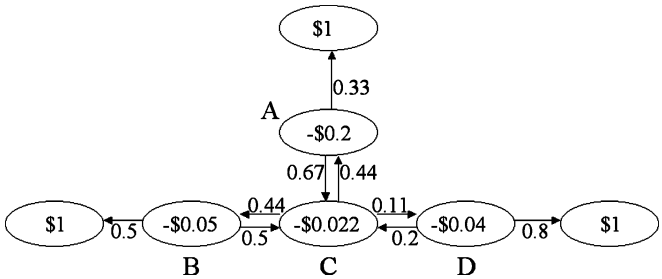



Fig. 5. Random-walk game corresponding to the circuit in Fig. 4.

TABLE I
CONVERGENCE OF THE SIMPLE EXAMPLE. $\Delta$ IS THE ERROR MARGIN IN (9), $V_A$ IS THE ESTIMATED VOLTAGE AT NODE A, $M$ IS THE ACTUAL NUMBER OF WALKS USED

| $\Delta$ | Exp #1 | | Exp #2 | | Exp #3 | | Exp #4 | | Exp #5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $V_A$ | $M$ | $V_A$ | $M$ | $V_A$ | $M$ | $V_A$ | $M$ | $V_A$ | $M$ |
| 0.05 | 0.6097 | 174 | 0.6067 | 156 | 0.5803 | 184 | 0.6418 | 103 | 0.6241 | 117 |
| 0.02 | 0.6087 | 1150 | 0.6075 | 946 | 0.5979 | 1140 | 0.5837 | 1254 | 0.6084 | 1232 |
| 0.01 | 0.6034 | 4562 | 0.6013 | 4664 | 0.6043 | 4315 | 0.5982 | 4441 | 0.6016 | 4619 |

either goes up with probability 0.33 to the terminal and ends this walk, or goes down with probability 0.67 to node C, then pays \$0.022, and continues from there. Such a walk could be very short: for example, the walker may directly go up and end up with \$0.8. Alternatively, the walk could be very long if it keeps going back and forth between A, B, C, and D, and the walker could end up with very little money; however, the probability of such a walk is low. We perform $M$ such experiments and take the average of the $M$ results as the expected gain during one walk and change the units from dollars to volts to obtain the estimated voltage of node A.

Table I shows how the estimated voltage converges to the true value of 0.6 V. The five columns in the table represent five different runs of the proposed algorithm, corresponding to different seeds for the random number generator.

Finally, as discussed in Section II-B, when we move on to other nodes after computing node A, node A can be used as a new home node, with an award equal to its estimated voltage.

### D. Runtime Trends

Due to the speedup technique at the end of Section II-B, the number of walks for a node $M$ decreases as computation proceeds. However, for flip-chip power grids (designs with C4 packaging) with similar structures, it is practically upper-bounded by certain constant $M'$, which corresponds to the number of walks needed to compute the node with the maximum $Var$, according to (10). Such a "difficult" node is likely to be the one that is the most faraway from C4 pads, for example, at the center of a square outlined by four adjacent C4 pads. If we consider two imaginary flip-chip circuits with the same structure around each C4 pad, one having a certain size and the other being infinitely large (infinitely many C4 pads), and if we consider the center node of a square outlined by four adjacent C4 pads in each of the two circuits, the two $Var$ values would be roughly the same. Therefore, the maximum $Var$ is independent of the circuit size, and consequently $M'$ is independent of the circuit size.

In the implementation, we will impose a constant limit $L$ on the number of steps in a walk; details are provided in Section V. Thus, for a power grid with $N$ non-$V_{DD}$ nodes, we can estimate worst-case time complexity as $O(LM'N)$, where each unit corresponds to one random-number generation, a few logic operations, and one addition. Therefore, the worst-case runtimes are linear in the circuit size $N$ for flip-chip designs.

Because of the fact that $M$ decreases as computation proceeds, the above worst-case discussion is an overestimate, and we will now look at the actual runtime, which can be viewed as the average-case runtime or the typical runtime. In order to argue that the average runtime per node is independent of the circuit size, let us consider two circuits with sizes $N_1$ and $N_2$, which are each solved with random ordering of nodes. At the same stage of the computation, for example, when $5\%N_1$ nodes are solved in the first circuit and $5\%N_2$ nodes are solved in the second circuit, because the densities of "homes" are the same (roughly 5%) in both circuits, the average lengths of walks are the same in both circuits, and the typical $Var$ values are also the same in both circuits, which means that the typical $M$ values are the same in both circuits at this time. Therefore, the CPU time for the $(5\%N_1+1)^{\text{th}}$ node in the first circuit is the same as the CPU time for the $(5\%N_2+1)^{\text{th}}$ node in the second circuit, in the average sense, and this is true for other percentage values as well. Therefore, the overall average runtime per node, which is equal to the average over all percentages, should be the same value for both circuits and independent of circuit sizes. Therefore, for flip-chip designs with similar structures, the overall runtime is linear in circuit size. This point is validated by the following simulations.

Four artificial circuits are constructed to verify the linearity of the average runtime in the case of flip-chip designs, and the results are listed in Table II. They are made with the same node-count to pad-count ratio. They all have a regular two-dimensional (2-D) grid structure, with the nodes forming a 2-D array of size $50 \times 50$, $100 \times 100$, $500 \times 500$, and $1000 \times 1000$, respectively, each edge of the grid being a 1 $\Omega$ resistor, and each node having a current load of 0.05 mA. The perfect voltage sources are at the intersections of the $(25+50i)^{\text{th}}$ horizontal

TABLE II
TIME COMPLEXITY OF THE GENERIC ALGORITHM FOR ARTIFICIAL FLIP-CHIP DESIGNS

| Circuit | Node number | Pad number | Total step number |
|---|---|---|---|
| #1 | 2.5K | 1 | 2.4e7 |
| #2 | 10K | 4 | 7.5e7 |
| #3 | 250K | 100 | 1.7e9 |
| #4 | 1M | 400 | 6.4e9 |

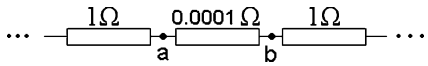



Fig. 6. Isolated low resistance forms a "trap."

wires and the $(25+50j)^{\text{th}}$ vertical wires, where $i$ and $j$ are nonnegative integers. (This structure is not realistic, but serves the purpose of time-complexity study.) The range of voltage drop is roughly the same 0–0.1 V, for all four circuits, and they are solved with the same accuracy $P[-5\text{ mV} < \text{ error} < 5\text{ mV}] > 99\%$, and with random ordering of nodes. The metric of complexity is the total step number of all the walks, where each step corresponds to one random-number generation, a few logic operations, and one addition. Table II shows a sublinear complexity for small sizes and more strictly linear complexity for larger sizes. These results agree with the earlier discussion of linear runtime.

For wire-bond power grids, however, the time complexity of the generic algorithm is superlinear.

### *E. Limitations of the Generic Method*

In Section V, we will use simulation results to show that the generic method has good performance for some industrial circuits. However, it also has been found that the generic method requires large runtimes for certain types of circuits.

One issue is shown in Fig. 6. If a single low resistance is isolated by other high resistances, because the random walker is more likely to choose the direction with lower resistance, he/she could spend many steps oscillating between nodes $a$ and $b$. Our algorithm employs a preprocessing step to detect such isolated low resistances and uses the $Y-\Delta$ transformation [27] to remove them without losing accuracy. In general, a subgraph formed by several low resistances can be isolated by other high resistances and form a "trap," and the corresponding discussion is provided in Appendix III.

Although the above problem can be taken care of inside the generic random-walk framework, the following are two other major problems that cannot.

1) In wire-bond power grids, a small number of perfect voltage sources are located on four sides of the top metal layer, and a walk from a central node takes a very large number of steps to terminate. Fig. 7 illustrates this power grid structure. In general, for a large graph with very few homes, the runtime is high.
2) In certain power grids, wire resistances in a metal layer are significantly larger than the resistances of the vias connecting the layer to the next metal layer beneath it. Because the random walk is more likely to choose a direction with lower resistance, this structure forms a barrier that makes it difficult for the walker to go up to the top layer and reach a perfect voltage source. Fig. 8 illustrates this effect. Note that this is different from the isolated-resistor problem in Fig. 6. These vias are not isolated by high resistances but are located between relatively well-connected lower layers and relatively bad-connected upper layers. Even if the vias are shorted, or removed by the $Y-\Delta$ transformation, random walks are still likely to stay in the lower layers. In real-life circuits, this is the situation where the pitch of power wires in one metal layer is much larger than the pitch in the next layer beneath it.

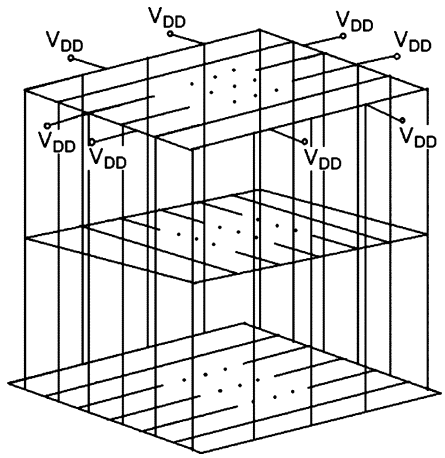



Fig. 7. Schematic of a wire-bond power grid structure with peripheral $V_{DD}$ pins.

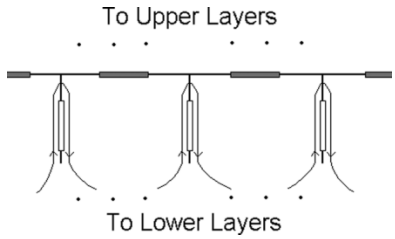



Fig. 8. Barrier effect. Gray rectangles represent high wire resistances, white rectangles represent low via resistances, and curves show the route of the random walker when he/she attempts to approach the top metal layer.

It is worth noting the fundamental reason for the first problem. The way that random walks estimate a node voltage drop is to capture the significant paths of current supply. Therefore, the detailed structure and current demand distribution of a power grid affect the runtime of random walks. For most flip-chip designs, the current load at a node is mainly supplied by several nearby voltage pads; therefore, random walks only account for these relatively short paths, and random walks are relatively short. For chips where there exists long-distance current delivery, which is true for wire-bond power grids, random walks try to capture all devices that significantly affect the target node, and hence can be very long.

In the next section, we will introduce the hierarchical random-walk method that overcomes these problems naturally and speeds up solutions to industrial circuits.

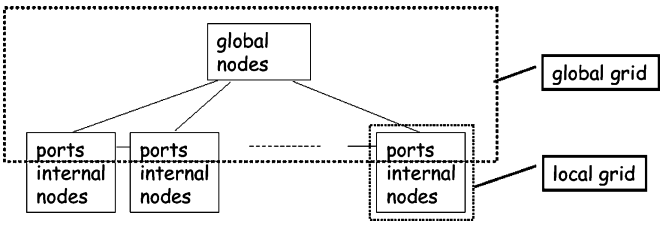



Fig. 9. Hierarchical strategy in [39].

## III. Hierarchical Random-Walk Methods

This section begins with a mathematical derivation of the single-level hierarchical method in Section III-A; the advantages of this method are discussed in Section III-B. Section III-C applies this concept to develop two variations, the multilevel method and the virtual-layer method, which are proven to be effective on industrial circuits. The discussion will be focused on dc analysis only.

### A. Principles

The hierarchical strategy in [39] is illustrated in Fig. 9. The whole power grid is divided into a global grid and multiple local grids, and interfacing nodes are defined as ports. From the global perspective, the behavior of a local grid is completely described by the following equation:

$$\mathbf{I_{ports}} = A\mathbf{V_{ports}} + \mathbf{S} \tag{11}$$

where $\mathbf{I_{ports}}$ is the vector of currents flowing from the global grid into this local grid, $\mathbf{V_{ports}}$ is the vector of port voltages, $A$ is a square matrix, and $\mathbf{S}$ is a constant vector. In dc analysis, matrix $A$ represents the effective conductances between the ports, and vector $\mathbf{S}$ represents local current sources.

The algorithmic flow of [39] is shown in Fig. 10. First, macromodels, i.e., the $A$ matrices and $\mathbf{S}$ vectors, are extracted from local grids. Next, the set of linear equations for the global grid is solved and port voltages obtained. Finally, local grids are solved individually. For realistic power grids, compared with a direct solver that solves (1), this algorithm solves a smaller equation set in each step. When the number of ports is reasonably small, and the $A$ matrices are reasonably sparse, a speedup is achieved.

*1) Constructing $A$ and $\mathbf{S}$:* An exact method for calculating $A$ and $\mathbf{S}$ is provided in [39], and 0–1 integer linear programming (ILP) is used to make $A$ sparse, at the expense of a bounded loss in accuracy. We will now demonstrate a random-walk approach to build $A$ and $\mathbf{S}$ and to achieve sparsity naturally as a part of this procedure.

The proposed macromodeling approach is based on the following lemma, which allows us to symbolically estimate one node voltage as a linear function of voltages at an arbitrary set of nodes. Its proof is provided in Appendix I.

*Lemma 1:* If we define $k$ nodes $h_1, h_2, \cdots, h_k$ to be the home nodes, i.e., terminals where random walks end, then for any node $i$, the estimated $V_i$ using $M$ random walks is in the form

$$V_i = c_1 V_{h_1} + c_2 V_{h_2} + \cdots + c_k V_{h_k} - \xi \tag{12}$$

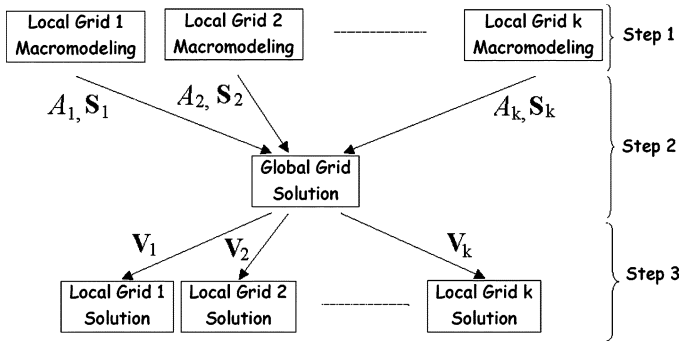



Fig. 10. Algorithm flow in [39].

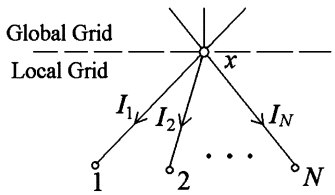



Fig. 11. Branches of $I_x$.

where $\xi$ is a constant, $V_{h_j}$ is the voltage at terminal node $h_j$, and coefficients have the format

$$c_j = \frac{\text{number of walks ending at } h_j}{M} \tag{13}$$

satisfying the relationship

$$c_1 + c_2 + \cdots + c_k = 1. \tag{14}$$

This lemma states that, given any set of nodes $h_1, h_2, \cdots, h_k$, and for any node $i$, we can find the values $c_1, c_2, \cdots, c_k$ and $\xi$ by running random walks such that the symbolical equation (12) is approximately true for any possible voltages at those nodes. The meaning of $c_j$ is given by (13), while $\xi$ is the average motel expense in one walk.

To construct $A$ and $\mathbf{S}$, we look at the ports one by one. Fig. 11 illustrates an example of a port $x$ that has several connections to the global grid and to a local grid. The current flowing from the global grid through $x$ into the local grid is

$$I_x = I_1 + I_2 + \cdots + I_N \tag{15}$$

where the neighbors of $x$ inside the local grid are labeled $1, 2, \cdots, N$, and $I_1, I_2, \cdots, I_N$ are the currents flowing from port $x$ to each of them.

Now consider node $i$, one of the neighbors of $x$: this could be either an internal node or a port node. If node $i$ is a port

$$I_i = g_i V_{\text{port } x} - g_i V_i \tag{16}$$

where $V_{\text{port } x}$ and $V_i$ are voltages at node $x$ and node $i$, respectively, and $g_i$ is the conductance between $x$ and $i$. On the other hand, if node $i$ is an internal node, we may run $M$ random walks from node $i$, with the ports being the terminals at which walks end, and estimate $V_i$ symbolically. Since there is usually no internal $V_{DD}/\text{GND}$ pads in a local grid, the set of all possible

terminals is $\{h_1, h_2, \cdots, h_k\} = \{\text{port } 1, \text{port } 2, \cdots, \text{port } k\}$. By Lemma 1, the following equation can be constructed.

$$V_i = c_1 V_{\text{port } 1} + c_2 V_{\text{port } 2} + \cdots + c_k V_{\text{port } k} - \xi. \quad (17)$$

Returning to the scenario in Fig. 11, we see that for any node $i$, one of the $k$ ports in (17) is $x$ itself. The current $I_i$ from port $x$ to an internal node $i$ is

$$\begin{aligned} I_i &= g_i V_{\text{port } x} - g_i V_i \\ &= g_i(1 - c_{\text{port } x}) V_{\text{port } x} - g_i \sum_{\text{port } j \neq x} c_j V_{\text{port } j} + g_i \xi \\ &\quad \forall i = 1, 2, \cdots, N. \end{aligned} \quad (18)$$

Equation (16) can be viewed as a special case of (18). Substituting (16) and (18) into (15), we obtain

$$I_x = \alpha_1 V_{\text{port } 1} + \alpha_2 V_{\text{port } 2} + \cdots + \alpha_k V_{\text{port } k} + \gamma$$

where

$$\begin{aligned} \alpha_j &= -\sum_{i=1}^{N} g_i c_j(\text{node } i) \quad \text{for } j \neq x \\ \alpha_x &= \sum_{i=1}^{N} g_i \left(1 - c_{\text{port } x}(\text{node } i)\right) \\ \gamma &= \sum_{i=1}^{N} g_i \xi(\text{node } i). \end{aligned} \quad (19)$$

This is what we need: $\alpha_1, \alpha_2, \cdots, \alpha_k$ is a row in matrix $A$, and the $\gamma$ term is an element in vector $\mathbf{S}$. So far, we have found the entries in $A$ and $\mathbf{S}$ that correspond to port $x$. For every port node of a local grid, we do the same thing, i.e., we construct (17) and (18) by running random walks from every internal neighbor of the port node and then obtain (19) by adding up (18)'s. This way, we can construct matrix $A$ row by row and vector $\mathbf{S}$ entry by entry.

The sparsity of the $A$ matrix is controlled by the number of random-walk experiments $M$. When (17) is constructed, it is an approximation to the real relation between $V_i$ and port voltages, which would be a full equation; in other words, $c_1, \cdots, c_k$ would be all nonzero values in the exact equation. However, due to the relationship shown in (13), any coefficient below $1/M$ cannot appear in the constructed equation (17). Therefore, effectively, insignificant terms are automatically dropped in the process of Lemma 1. This eventually leads to the sparsity of $A$. $M$ can be viewed as the resolution of the estimation. The larger $M$ is, the more entries in (17) are nonzero, and the denser $A$ is.

Although the number $M$ provides a control over the sparsity–accuracy tradeoff, and $M$ can be dynamically determined by a stopping criterion similar to (9) defined on $\gamma$, there is not an analytical formulation of the relation between $M$ and the accuracy of matrix $A$. In our implementation of the hierarchical method reported in Section V, $M$ is predetermined and fixed in each run, and different tradeoff points are found by changing $M$. It is worth noting that our approach is compatible with [39], i.e., our three steps and the three steps of [39] are interchangeable and can form a hybrid scheme; the ILP method in [39] can also be used as a postprocessing step to make our constructed matrix $A$ sparser. There is, however, a difference between the two

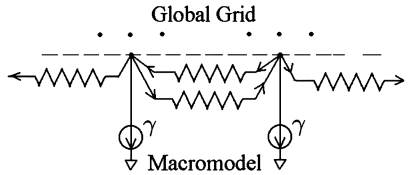



Fig. 12. Imaginary circuit interpretation of a macromodel in dc analysis.

that gives our method an advantage over [39] for not-easily-partitionable designs, and this will be discussed in Section III-B.

*Lemma 2:* The sum of each row of the estimated $A$ matrix is zero.

From (19), all weights of the port voltages are negative except for $\alpha_x$. For any row in matrix $A$, from (14) and (19), we get

$$\begin{aligned} \sum_{\text{port } j=1}^{k} \alpha_j &= \alpha_x + \sum_{\text{port } j \neq x} \alpha_j \\ &= \sum_{i=1}^{N} g_i \left(1 - c_{\text{port } x}(\text{node } i)\right) \\ &\quad - \sum_{\text{port } j \neq x} \sum_{i=1}^{N} g_i c_j(\text{node } i) \\ &= \sum_{i=1}^{N} g_i \left(1 - \sum_{\text{port } j=1}^{k} c_j(\text{node } i)\right) \\ &= 0 \quad (\text{applying equation (14)}). \end{aligned} \quad (20)$$

Thus, we have proven Lemma 2. In general, $A$ will not be symmetric even though the exact $A$ matrix has this property; however, it is preferable to leave it this way in order to preserve accuracy. As we will show in the next section, this is not a barrier to solving the global grid.

*2) Solving the Global Grid:* Now we move on to Step 2 in Fig. 10, solving the global grid based on the extracted macromodels. In order to do so using random walks, we interpret each macromodel as an imaginary circuit. Due to Lemma 2, (19) can be converted to

$$I_x = \sum_{\text{port } j \neq x} (-\alpha_j)(V_{\text{port } x} - V_{\text{port } j}) + \gamma. \quad (21)$$

Equation (21) can be viewed as a circuit, in which $(-\alpha_j)$ conductance connects node $x$ and node $j$, and an independent current source $\gamma$ flows out of node $x$. This is an imaginary circuit, because each resistor only exists for one direction (corresponding to the asymmetry of the computed $A$ matrix), i.e., the conductance from node $x$ to node $j$ could be different from the conductance from node $j$ to node $x$. Fig. 12 illustrates this imaginary circuit composed of directed resistors.

Based on this imaginary circuit interpretation, the global grid can be solved by running random walks from each port nodes, and the port voltages may be obtained.

*3) Solving the Local Grids:* Next, we move on to Step 3 in Fig. 10, solving the bottom-metal-layer nodes in each local grid, based on the port voltages computed in Step 2. The ports correspond to "homes" in this random-walk game, and each walk

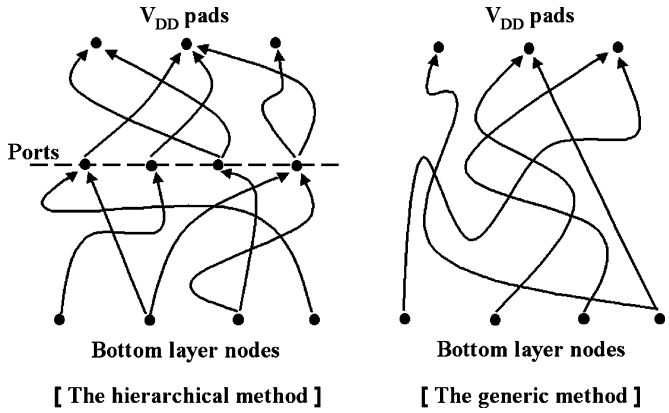



Fig. 13. Random walks in the hierarchical algorithm are shorter than those in the generic algorithm.

from the bottom layer typically ends within a reasonably small number of steps.

### B. Benefits of Hierarchy

The approach in [39] requires the demarcation of partitions corresponding to small cuts between partitions, since this would lead to small port matrices. In our approach, such partitioning is not necessary, and the algorithm only needs to distinguish local nodes, global nodes, and ports. Therefore, multiple local grids are not needed, and only the boundary between the global grid and the local grid needs to be decided. This can be done in various ways, and we recommend the following natural approach: given a power grid, we choose a layer of vias as the border between the global grid and the local grid and define the upper ends of these vias as ports and the lower ends as the internal neighbors of the ports.

Choosing this layer of vias is a new degree of tradeoff: if a lower layer is chosen, the global grid size is larger, the number of ports is larger, and consequently solving the global grid takes more runtime; on the other hand, the local grid is smaller and there are more terminals, i.e., solved ports, and therefore solving the local grid takes less time. In practice, we choose a layer of vias such that the global grid is roughly 10% of the entire circuit size.

Compared with the generic random-walk algorithm, the hierarchical algorithm has two major advantages.

1) The hierarchical method is faster. The reason is illustrated in Fig. 13. When solving the global grid, each random walk starts from a port and ends at a perfect voltage source; when solving the local grid, each random walk starts from a bottom-layer node and ends at a port. In either case, a walk has fewer steps than a walk in the generic method that starts from a bottom-layer node and has to reach a perfect voltage source at the top metal layer. Also, when random walks are shorter, the variance of the results of walks tends to be lower, and consequently, a higher accuracy can be achieved with the same number of walks, or fewer walks are needed to achieve the same accuracy level. Although we pay the overhead of building macromodels, the overall savings typically dominate this cost. We will validate this by our experimental results in Section V.
2) The hierarchical method is more robust. As illustrated in Fig. 8, in certain power grids, a highly resistive metal layer forms a barrier that makes it difficult for the walker to go up to the top layer, and the runtime of the generic method is therefore very long. The hierarchical method solves these circuits simply by defining ports right on this barrier. In other words, instead of relying on the random walker to pass this barrier, we cut a walk into two segments and preserve the barrier nature in the macromodel. This can also be viewed as an extreme case of the speedup shown in Fig. 13. Corresponding results can be found in Section V. If there exist more than one such barrier structure in a power grid, they can all be handled by multilevel hierarchy, which is presented in the next section.

As mentioned in Section II-E, a more general problem of the generic random-walk method is that in a large graph with very few homes, the runtime is high. One example is wire-bond power grids with pads at the periphery, shown in Fig. 7. In the next section, we will show how such graphs can be handled when the idea of hierarchy evolves to the virtual-layer concept.

Finally, we want to point out a defect of hierarchy. In the hierarchical algorithm, we can no longer solve a single node only: the overhead of building and solving the hierarchy has to be paid first. In other words, the algorithm does not have the *complete* locality anymore. One way to maintain a *partial* locality is to use multiple local grids: when a change is made in the design, only the macromodel of the local grid containing the change needs to be rebuilt and re-solved. Note that this is not included in our implementation, which uses a single local grid, as discussed at the beginning of this section.

### C. Variations of Hierarchy

A natural extension of the algorithm in Section III-A is to use multilevel hierarchy. Making use of all available vias, we can build macromodel on top of macromodel. After this bottom-up traversal, the circuit is reduced to a global grid, then port voltages are solved in a top-down order, and the bottom-layer voltages are obtained in the end. Compared with the single-level method, the extra cost of the multilevel method is building multiple macromodels, while the benefit is shorter walks in each level. Hence, there is a tradeoff in choosing the number of levels. Since the single-level hierarchical method is better than the generic method, we expect the multilevel method to be even faster and more robust. Test results in Section V show that the multilevel method has a similar accuracy–runtime tradeoff as the single-level method.

Another extension leads to the concept of a "virtual-layer," when we choose ports such that the global grid physically does not exist. In other words, there are no direct connections between these ports in the original circuit: this can be considered to be similar in flavor to grid coarsening in [19]. When we abstract all connections of these ports into a macromodel, this macromodel provides imaginary connections between ports, and the global grid is totally composed of such virtual connections, as shown in Fig. 14.

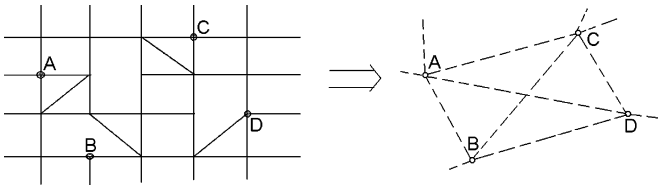



Fig. 14. Original graph with ports marked, and the extracted virtual layer.

For example, in a large graph where the number of homes is very limited and all homes are located at periphery, a random walk from a center node typically needs a very large number of steps. We may traverse the graph (for example, a breadth-first search in our implementation), and mark one port in every $p$ nodes. For $p = 10$, the sampling rate is 1/10; this number cannot be too low, because we have to guarantee that the virtual layer will be a connected graph. Special arrangements must be made such that each home is surrounded by ports, because edges leading to a home should not be abstracted into the macromodel. Then all connections of these ports are abstracted into a macromodel, except for those leading to a home. Thus, the virtual layer is constructed, and the size is 10% of the original graph. After solving it, we go back to the local grid, i.e., the original graph, and because there are solved ports all over the graph, it can be solved efficiently.

This virtual-layer method will be shown useful when solving a wire-bond power grid in Section V

## IV. RKC Transient Analysis

In this section, we extend the random-walk algorithms to transient analysis, where voltage waveforms are to be found while considering the effects of capacitances, inductances, and time-varying current waveforms. Throughout this section, and in the implementation, the backward Euler approximation with a constant timestep $h$ is used to convert differential equations to linear equations. We assume that the timestep size $h$ is kept constant in a transient analysis.

### A. Capacitors

Let us first incorporate capacitors into the proposed framework. The equations to be solved are as follows [14]:

$$G\mathbf{V}(t) + C\mathbf{V}'(t) = \mathbf{E}(t) \tag{22}$$

where $G$ is a conductance matrix, $C$ is the matrix introduced by capacitors, $\mathbf{V}(t)$ is the vector of node voltages, and $\mathbf{E}(t)$ is the vector of independent sources. Applying the backward Euler formula with a timestep of $h$, the equations become

$$\left(G + \frac{C}{h}\right)\mathbf{V}(t) = \mathbf{E}(t) + \frac{C}{h}\mathbf{V}(t-h). \tag{23}$$

This transformation translates the problem to solving a linear equation set. As before, we consider one single node $x$, at one timestep at time $t$, and we have

$$\sum_{i=1}^{\text{degree}(x)} g_i\left(V_i(t) - V_x(t)\right) = \frac{C_x}{h}\left(V_x(t) - V_x(t-h)\right) + I_x(t) \tag{24}$$

where $V_x$, $V_i$, $I_x$, and $g_i$ are as defined in (2), and $C_x$ is the capacitance between node $x$ and ground.

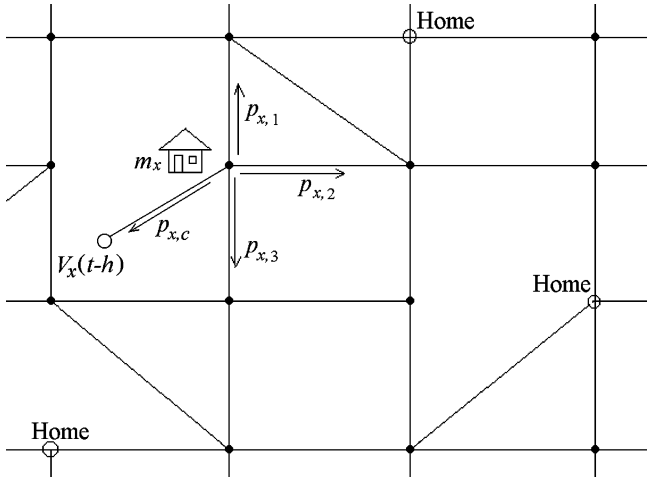



Fig. 15. Rules for the transient analysis "game."

For an *RC* network with capacitors between two nonground nodes, those capacitors can be replaced by resistors and current sources, while a current source between two nodes can be replaced by two current sources between the two nodes and ground. Then, the following algorithm is applicable. Here, we only discuss the case described in (24).

Equation (24) can be converted to the following form:

$$V_x(t) = \sum_{i=1}^{\text{degree}(x)} \frac{g_i}{\sum_{j=1}^{\text{degree}(x)} g_j + \frac{C_x}{h}} V_i(t) + \frac{\frac{C_x}{h}}{\sum_{j=1}^{\text{degree}(x)} g_j + \frac{C_x}{h}} V_x(t-h) - \frac{I_x(t)}{\sum_{j=1}^{\text{degree}(x)} g_j + \frac{C_x}{h}}. \tag{25}$$

The rules of the random-walk game are changed to accommodate the changes in the above equation. As shown in Fig. 15, each node $x$ has an additional connection, and the walker could end the walk and be awarded the amount $V_x(t-h)$ with probability

$$\frac{\frac{C_x}{h}}{\sum_{j=1}^{\text{degree}(x)} g_j + \frac{C_x}{h}}.$$

Intuitively, this rule is equivalent to replacing each capacitor by a resistor and a voltage source.

Under this new rule, the random-walk game is mathematically equivalent to the equation set (23), and both the generic method and the hierarchical method can perform transient analysis of a *RC* network, timestep by timestep. In each timestep, the $V(t-h)$ values are updated with the node voltage values solved from the previous timestep.

The complete locality of the dc generic algorithm is still valid in the *RC* generic algorithm: we can compute a single node voltage at a single time point without solving any other nodes or any other timesteps. If we want to compute the voltage at node $x$ at time $t$, the walks start at node $x$ in the random walk game for time $t$; some walks may reach $V(t-h)$ terminals, and then they continue in the random walk game for time $(t-h)$; some of these may reach $V(t-2h)$ terminals, and then they continue in the random walk game for time $(t-2h)$, and so on. The

real terminals where walks end are those from physical voltage sources, which are present at all times. The farthest a walk can go in time is the time point zero, which is a dc analysis game. In short, "traveling back time" makes the complete locality feasible, and this is inspired by [1].

The hierarchical method is affected by the additional $V(t-h)$ terminals in various ways, and we will now take a close look. When we build macromodels as described by Lemma 1, the set of possible terminals not only contains ports, but also the $V(t-h)$ terminals inside the local grid. Consequently, instead of (17), we get

$$V_i = c_1 V_{\text{port } 1} + c_2 V_{\text{port } 2} + \cdots + c_k V_{\text{port } k} + c_{k+1} V_{h_{k+1}} + \cdots + c_{k'} V_{h_{k'}} - \xi \quad (26)$$

where $h_{k+1}, \cdots, h_{k'}$ are the $V(t-h)$ terminals inside the local grid. Since the $V(t-h)$'s are known values, they can be lumped into a constant

$$\xi' = \xi - \left(c_{k+1} V_{h_{k+1}} + \cdots + c_{k'} V_{h_{k'}}\right) \quad (27)$$

and we get

$$V_i = c_1 V_{\text{port } 1} + c_2 V_{\text{port } 2} + \cdots + c_k V_{\text{port } k} - \xi'. \quad (28)$$

With (28), we can continue the macromodeling and construct equations (18) and (19). However, the sum of $c_1, \cdots, c_k$ is no longer one, and that relation is replaced by

$$c_1 + c_2 + \cdots + c_k \le 1. \quad (29)$$

As a consequence of (29), Lemma 2 is replaced by the following lemma in transient analysis.

*Lemma 3:* The estimated $A$ matrix in transient analysis is diagonally dominant.

To prove Lemma 3, we rewrite (29) as

$$1 - c_{\text{port } x} \ge \sum_{\text{port } j \ne x} c_j. \quad (30)$$

Applying this inequality and (19), we get

$$\begin{aligned} \alpha_x &\ge \sum_{i=1}^{N} g_i \sum_{\text{port } j \ne x} c_j(\text{node } i) \\ &= \sum_{\text{port } j \ne x} \sum_{i=1}^{N} g_i c_j(\text{node } i) \\ &= \sum_{\text{port } j \ne x} (-\alpha_j) = \sum_{\text{port } j \ne x} |\alpha_j|. \end{aligned} \quad (31)$$

This inequality holds for any row of matrix $A$, and thus we have proven Lemma 3.

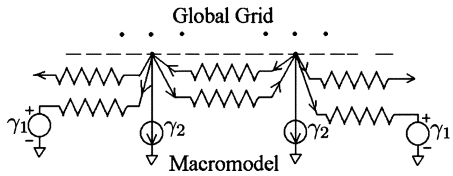



Fig. 16. Imaginary circuit interpretation of a macromodel in transient analysis.

Also because of inequality (29), as well as Lemma 3 replacing Lemma 2, the formulation of the imaginary circuit (21) becomes the following in transient analysis:

$$\begin{aligned} I_x &= \sum_{\text{port } j \ne x} (-\alpha_j)(V_x - V_{\text{port } j}) \\ &\quad + \left(\alpha_x + \sum_{\text{port } j \ne x} \alpha_j\right) V_x + \gamma \\ &= \sum_{\text{port } j \ne x} (-\alpha_j)(V_x - V_{\text{port } j}) \\ &\quad + \left(\alpha_x + \sum_{\text{port } j \ne x} \alpha_j\right)(V_x - \gamma_1) + \gamma_2. \end{aligned} \quad (32)$$

There are many ways to split $\gamma$ into $\gamma_1$ and $\gamma_2$. One of them is very meaningful: $\gamma_1$ is a weighted average of some $V(t-h)$'s inside this local grid, and $\gamma_2$ is a weighted sum of some current loads inside this local grid, as follows:

$$\gamma_1 = \frac{\frac{C_x}{h} V_x(t-h) + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i} V_{h_j}}{\frac{C_x}{h} + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i}} \quad (33)$$

$$\gamma_2 = \sum_{i=1}^{N} \frac{g_i}{M} \sum_{q=1}^{M} \sum_{r} \frac{I_{q,i,r}}{s_{q,i,r}} \quad (34)$$

where the neighbors of $x$ inside the local grid are labeled $1, 2, \cdots, N$; $C_x$ is capacitance between port $x$ and ground; $g_i$ is conductance between $x$ and $i$; $(k' - k)$ is the number of $V(t-h)$ terminals inside the local grid, and they are $h_{k+1}, \cdots, h_{k'}$; $N_{h_j,i}$ is the number of walks from $i$ that end at terminal $h_j$; $I_{q,i,r}$ is the current load flowing out of the node at the $r^{\text{th}}$ step of the $q^{\text{th}}$ walk from node $i$; and $s_{q,i,r}$ is the sum of the conductances connected to the node at the $r^{\text{th}}$ step of the $q^{\text{th}}$ walk from node $i$. The proof of (33) and (34) is provided in Appendix II.

Due to Lemma 3, the term $(\alpha_x + \sum_{\text{port } j \ne x} \alpha_j)$ is nonnegative, and (32) can still be interpreted as a circuit. Fig. 16 illustrates this new imaginary circuit: $(-\alpha_j)$ conductance connects node $x$ and node $j$; $(\alpha_x + \sum_{\text{port } j \ne x} \alpha_j)$ conductance connects $x$ to a voltage source with voltage equal to $\gamma_1$; and an independent current source $\gamma_2$ flows out of node $x$.

### B. Inductors

Inductances include self-inductances and mutual inductances. Under the backward Euler approximation, a self-inductance becomes a resistor and a current source in parallel and can be easily handled by the random-walk algorithms. However,

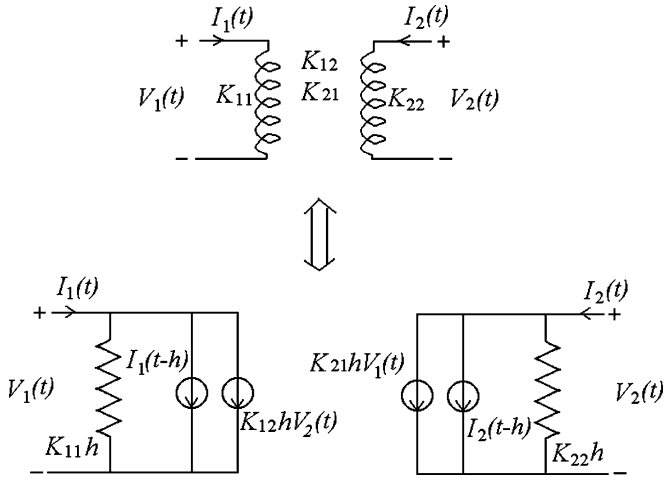



Fig. 17. Companion model of a pair of inductors, adapted from [15].

mutual inductances are difficult to incorporate into the proposed framework, because of their induced extra unknown variables: the currents through the inductors.

Therefore, we use the inverse inductance, or susceptance, matrix $K$ [7], [15] to model inductors. It has also been shown that the $K$ matrix has better locality than the partial inductance matrix $L$, and hence reduces the problem size of circuit simulation [7], [15]. The $K$ matrix is defined as the inverse of the $L$ matrix, and the device equations under the backward Euler approximation are

$$K\mathbf{V}(t) = \frac{\mathbf{I}(t) - \mathbf{I}(t-h)}{h} \tag{35}$$

where $\mathbf{V}(t)$ is the vector of voltage drops over the inductors, $\mathbf{I}(t-h)$ is the vector of known currents through the inductors from the previous timestep, $\mathbf{I}(t)$ is the vector of unknown currents through the inductors in the present timestep, and $h$ is the timestep size. Equation (35) can be written as

$$\mathbf{I}(t) = hK\mathbf{V}(t) + \mathbf{I}(t-h) \tag{36}$$

and the corresponding companion model is illustrated in Fig. 17, where only a pair of coupled inductors are shown.

In our circuit model, each wire segment is a $\pi$ model, which is composed of a resistor and an inductor in series, and capacitors at two ends. Fig. 18(a) shows this model, where capacitors are not drawn. By substituting the companion model of the inductor, we obtain the circuit in Fig. 18(b), where the inductor is replaced by a resistor and two current sources in parallel. One of the two current sources is equal to the current from the last timestep, which is a known constant; the other source is a voltage-controlled current source that corresponds to the current induced by other inductors, i.e., a function of $V_B$'s and $V_C$'s from a number of other wire segments. The model in (b) can be further converted to (c) and then to (d), which is a circuit form that can be handled by our random-walk algorithms.

One complication caused by mutual inductances is that the current sources in Fig. 18(d) is a function of not only $V_A$'s and $V_B$'s, but also $V_C$'s, while $V_C$'s are not among the system variables when we solve the circuit in form (d). In other words,

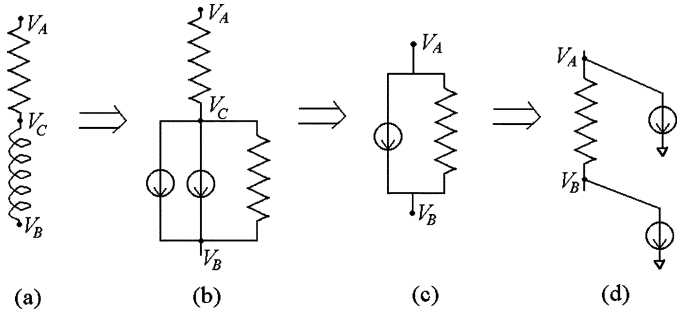



Fig. 18. Wire segment model.

the voltage-controlled current sources cannot be expressed as a linear function of node voltages.

To resolve the above problem, we propose an iterative approach to compute node voltages in each timestep, and in each iteration, we assume the voltage-controlled current sources to have constant values. First, we use $V_B$'s and $V_C$'s of all wire segments from the previous timestep as the initial guess and compute the values of the current sources in Fig. 18(b)–(d). Next, by assuming these current sources to be constant, we use random walks to solve the circuit in form (d) and obtain new $V_A$ and $V_B$ values. Then, we update $V_C$'s and hence current sources in (b)–(d) and solve in form (d) again. This process iterates until voltages converge.

In the hierarchical algorithm, if we only consider mutual inductances inside the global grid, i.e., the top few metal layers, then the above iteration is only done in the stage of solving the global grid. However, if we consider the general case, where mutual inductances exist in the local grid and between the local and global grids as well, then one iteration in the above iterative process includes all three stages: macromodeling, solving the global grid, and solving the local grid. Note that with the bookkeeping technique presented in the next section, these computations can be done efficiently without running random walks.

The above iterative approach is guaranteed to converge, and the proof is provided in Appendix IV. In our simulations, the convergence criterion is maximum voltage difference being less than $10^{-5}$ V, and the iterative process always converges within three iterations. The results are reported in Section V.

### C. Bookkeeping

For transient analysis, traditional direct linear equation solvers are efficient in computing solutions for succeeding timesteps after initial matrix factorization since only a forward/backward substitution step is required for each additional timestep. Analogously, our random-walk algorithm employs a speedup mechanism.

In the generic method, we first perform a dc analysis that is used as the initial condition; next, when computing the first transient timestep, we keep a record for each node. This record keeps a count of, in these $M$ walks, how many times the walker ends at $V_{DD}$, how many times the walker ends at some $V(t-h)$, how many times the walker pays for a motel at some node, and so on. Then, in the follow-up timesteps, we do not perform random walks any more, and simply use these records recursively and assume that the walker gets awards at same locations and pays

for same motels, and only the award amounts and motel prices have changed. Thus, with bookkeeping, new voltages can be computed by some multiplications and additions efficiently.

This bookkeeping technique is based on the observation that the routes taken by the walks are decided solely by the resistor values corresponding to the resistances in the original circuit and the resistances that arise from the companion models for the capacitors and inductors. Under a constant timestep size, these are all unchanged from one timestep to the next, and therefore, using the same routes is justifiable. The task of bookkeeping maintains the set of nodes visited in these walks and the frequency with which they are visited. From one timestep to the next, however, the current source values in the circuit will change, and the voltage/current sources from companion models will change, since $V(t-h)$ at a capacitor will be different from $V(t-2h)$, and $I(t-h)$ through a inductor will be different from $I(t-2h)$, etc. This implies that the motel prices and reward values will change. With the help of the bookkeeping information, the voltage at each node can simply be found as a weighted sum of values at nodes visited, which are maintained in the bookkeeping record.

In the hierarchical methods, we also keep a record for solving the global grid, as well as building the vector $\mathbf{S}$, because $\mathbf{S}$ needs to be updated whenever the current sources or $V(t-h)$ sources in the local grid change.

The space complexity demanded by this bookkeeping is approximately linear in the number of nodes and is not worse than the space complexity of a traditional direct solver. This will be shown in Section V. Another concern is whether using the same record repeatedly could cause error accumulation. Our simulation results show that the error level is acceptable even after 1000 timesteps.

Finally, the values that need to be repeatedly updated in transient analysis are listed in the following.

1) $V(t-h)$ values in Fig. 15, for every timestep.
2) Current source values in Fig. 18(b)–(d), for every iteration in every timestep.
3) Hierarchical methods update vector $\mathbf{S}$, which is a function of current sources and $V(t-h)$ sources in the local grid, for every timestep;
4) In the case of the hierarchical method where there are mutual inductances in the local grid and between the local and global grids, vector $\mathbf{S}$ needs to be updated, for every iteration in every timestep, as described in the previous section.

## V. Results

In this section, we use three industrial benchmarks to evaluate the proposed algorithms for dc analysis. Then, *RC* and RKC circuits generated based on structures of real-life circuits are used to test the performance of transient analysis. Computations are carried out on a Linux workstation with 2.8 GHz CPU frequency.

The three industrial power grids are listed hereafter.

1) Industry1 is a 70 729-node circuit, and we solve for the 15,876 bottom-metal-layer $V_{DD}$ nodes and 15 625 bottom-metal-layer ground nodes. The voltage range of $V_{DD}$ bottom layer is 1.1324–1.1917 V.
2) Industry2 has 218 947 nodes, in which 25 137 bottom-metal-layer $V_{DD}$ nodes and 18 803 bottom-metal-layer ground nodes are to be solved. The voltage range of the $V_{DD}$ bottom layer is 1.612 48–1.798 22 V, that of ground bottom layer being 0.000 334–0.066 505 V.
3) Industry3 is a wire-bond ground net with 347 566 nodes, and the bottom layer has a voltage range of 0.024 347–0.110 860.

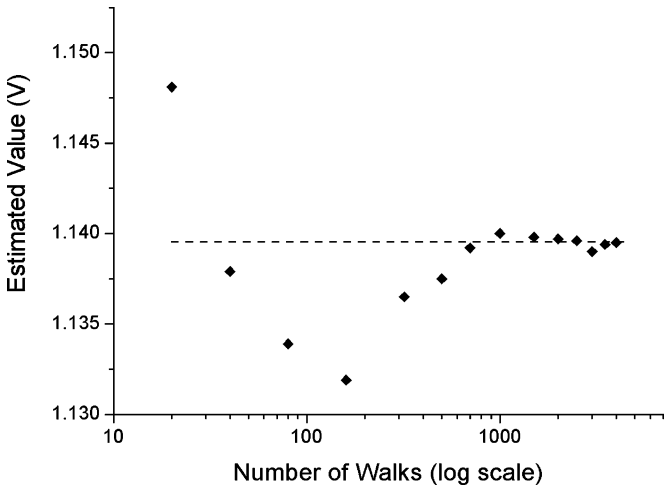



Fig. 19. Estimated voltages at a single node for various values of $M$.

One implementation issue is the choice of random number generator. A random number uniformly distributed between 0 and 1 is needed for making a decision at each step in a random walk. The higher the required quality is for random numbers, the longer the runtime is. In all our implementations, we use the generator of [24, p. 279]. The simulation results show that it provides sufficient quality for power grid analysis application.

Fig. 19 shows the results of computing the solution for only one node in Industry1, using the generic algorithm. The markers are estimated voltage values for different $M$'s, and the dashed line is the true voltage. The ultra-accurate right-most point, for which $M$ is 4000, only takes 0.42-s runtime, and thus shows the efficiency of using our algorithm to solve individual nodes without solving the whole circuit. This is ideal in the scenario of incremental design, where the designer makes a change and wants to see the effect on a node voltage. Note that in the hierarchical algorithms, we can no longer solve a single bottom-metal-layer node only: the overhead of building and solving the hierarchy has to be paid first. For example, if we use the single-level hierarchical method to solve one bottom-metal-layer node in Industry2, this overhead is 15 s. In the scenario of incremental design, this is still better than solving the whole linear equation set.

As indicated in Section II-B, one implementation issue is that, in order to avoid any possible deadlock, we need to set a limit $L$ on the number of steps in a walk. Any walk that fails to end within $L$ steps will be forced to end, and be awarded $V_{DD}$ if inside the $V_{DD}$ net, and be awarded 0 if inside the ground net. This operation is optimistic and will result in a bias in the estimated voltage; however, if the limit is chosen appropriately, the error

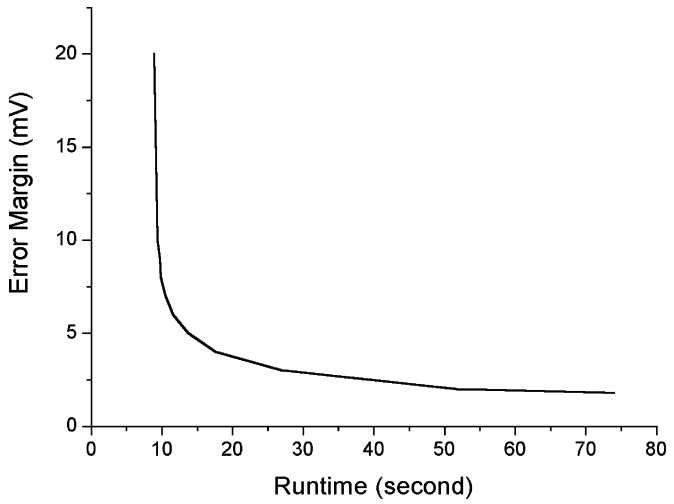



Fig. 20. Runtime–$\Delta$ tradeoff for the computation of all bottom-metal-layer nodes in Industry1.

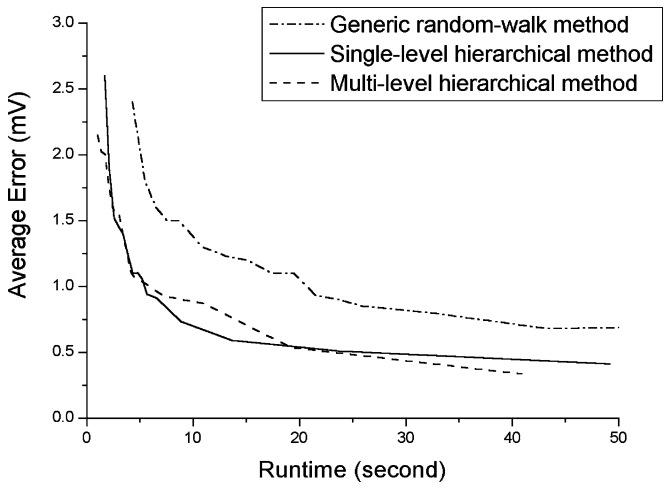



Fig. 21. Accuracy–runtime tradeoff curves for solving Industry1 using the generic random-walk method, the single-level hierarchical method, and a two-level hierarchical method.

will be very small as the probability of a walk of this length is minute. Thus, a new degree of accuracy–runtime tradeoff is introduced, and we empirically set this limit to be 10 000 steps as a good tradeoff point, where the bias error is acceptable and not much runtime is wasted. For example, when the generic method solves Industry1 with $\Delta$ being 4 mV, there are 2.5 K walks that violate this step limit and are forced to end, and the starting nodes of these 2.5 K walks are 1.6 K nodes; for these 1.6 K nodes, the average error is 1.86 mV, and the maximum error is 5.97 mV. (For reference, the overall average error is 1.64 mV, and the overall maximum error is 8.86 mV. Thus, the impact of the step limit is minor.) For hierarchical methods, there are typically no or only a few violations.

The above tradeoff only affects runtime indirectly, while the error margin $\Delta$ in (9) decides $M$, which is directly proportional to runtime and needs careful investigation. Fig. 20 plots the relation between $\Delta$ and runtime for solving the complete Industry1, i.e., finding all bottom-metal-layer voltages, using the generic random-walk method. The runtime is always larger than 8 s because the minimum value of $M$ is set to be 40. The lower part of this curve shows the quadratic relation between $M$ and $\Delta$: $M \propto 1/\Delta^2$. For example, the runtime is around 15 s when $\Delta$ is 4 mV, and roughly 60 s when $\Delta$ is 2 mV.

Fig. 21 plots the tradeoff between average error and runtime solving Industry1, where the three curves are for the generic random-walk method, the single-level hierarchical method, and a two-level hierarchical method, respectively. All hierarchies are divided at vias. All three methods use predetermined and fixed $M$ in each run, and points on the curves correspond to different $M$ values. Both hierarchical methods achieve roughly three to four times speedup over the generic method, with the same average error.

In practice, the user decides the tradeoff point by choosing $M$ values according to the needs of the analysis. Here, for runtime comparison purpose, we choose a reasonable tradeoff point on each of the three curves, and list them in Table III.

Fig. 22 plots the tradeoff between average error and runtime solving Industry2, using the single-level hierarchical method and a three-level hierarchical method. All hierarchies are divided at vias. Both methods use predetermined and fixed $M$ in each run, and points on the curves correspond to different $M$ values. The curve for the generic random-walk method is omitted because its runtime is unacceptably high for this circuit. The reason has been discussed in Section II: a highly resistive metal layer on top of low-resistance vias forms a barrier structure. This circuit shows an example of the robustness introduced by hierarchy. Again, the tradeoff point should be decided by the designer. Here, we choose a reasonable tradeoff point on each of the curves for Table III. One tradeoff point of the generic method is also listed.

TABLE III
dc ANALYSIS COMPARISON. N IS CIRCUIT SIZE, E1 IS AVERAGE ERROR, E2 IS MAX ERROR, T IS RUNTIME, NT IS NORMALIZED RUNTIME, DEFINED AS RUNTIME PER THOUSAND NODES, P IS PEAK MEMORY, AND NP IS NORMALIZED PEAK MEMORY, DEFINED AS PEAK MEMORY PER THOUSAND NODES. G DENOTES THE GENERIC RANDOM-WALK METHOD, S DENOTES THE SINGLE-LEVEL HIERARCHICAL METHOD, AND M DENOTES THE MULTILEVEL HIERARCHICAL METHOD

| Benchmark | | N | E1(mV) | E2(mV) | T | NT(sec) | P(MB) | NP(MB) |
|---|---|---|---|---|---|---|---|---|
| Industry1 | G | 71K | **1.1** | **9.8** | **17.40 sec** | **0.245** | **10.7** | **0.15** |
| | S | | **1.1** | **6.6** | **4.34 sec** | **0.061** | **11.4** | **0.16** |
| | M | | **1.1** | **9.4** | **4.16 sec** | **0.059** | **16.8** | **0.24** |
| Industry2 | G | 219K | 10.9 | 142.2 | 329.57 sec | 1.50 | 27.3 | 0.12 |
| | S | | **1.4** | **30.7** | **20.82 sec** | **0.095** | **37.0** | **0.17** |
| | M | | **1.4** | **35.3** | **30.12 sec** | **0.138** | **41.4** | **0.19** |
| Industry3 | G | 348K | 4.3 | 7.6 | 71 min | 12.2 | 57.7 | 0.17 |
| | S | | 4.4 | 18.8 | 498.02 sec | 1.43 | 72.4 | 0.21 |
| | M | | **3.6** | **17.0** | **93.64 sec** | **0.27** | **84.6** | **0.24** |
| *Chip2 by the method of [39]* | | *2.7M* | *N/A* | *N/A* | *25 min* | *0.56* | 300 | 0.11 |

The runtime comparison is shown in Table III. The three rows Industry2-G, Industry3-G, and Industry3-S are results with robustness problems, as discussed in Section II-E, while the boldface rows are results without the problems, or with them overcome. The numbers for chip2 in [39] are listed as a baseline. In

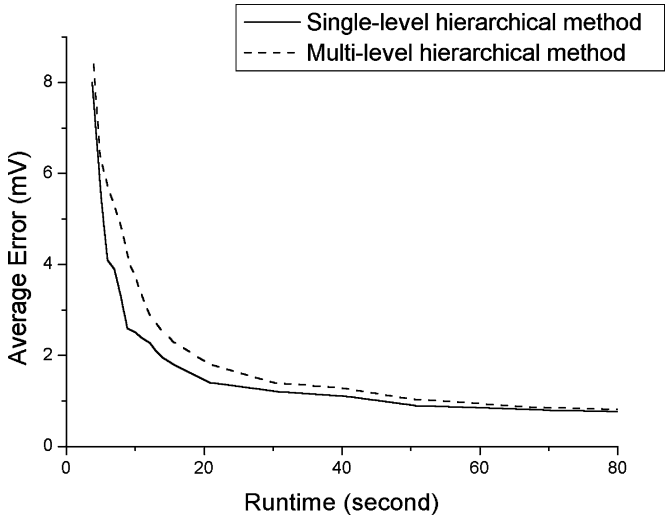



Fig. 22. Accuracy–runtime tradeoff curves for solving Industry2, using the single-level hierarchical method and a three-level hierarchical method.

viewing the numbers, it is important to note that our computer is approximately three times faster than those used by [39], according to SPEC benchmarks [31]. Runtimes reported by [39] show superlinear time complexity; chip2 is their smallest circuit and therefore has the smallest normalized runtime. Since the time complexity of random-walk algorithms is linear in circuit size (for circuits with similar structure), as power grid size increases, they will outperform [39] more. Note that due to factors such as benchmark structure, coding, compiling, and platform difference, this is only an approximate comparison, even after considering the $3\times$ factor.

For Industry1, both hierarchical methods show a $4\times$ speedup over the generic method. For Industry2, the speedup is dramatic and shows the robustness introduced by hierarchy.

The multilevel hierarchical method does not show a runtime advantage over the single-level method for Industry2. The reason is that the benefit of the multilevel hierarchy, which is easier access to home nodes, is not worth the cost of building multiple macromodels, for the Industry2 case with C4 packaging. However, it is worthwhile for Industry3, a similar-sized circuit with wire-bond packaging.

Industry3 is a wire-bond power grid, a difficult circuit type to solve. Even after it is reduced to its top metal layer only, there are still 80 K nodes, yet there are only 20 perfect voltage sources distributed on four sides of the top metal layer. Thus, it requires high runtimes if using the generic method or the single-level method, as listed in Table III. We employ a two-level hierarchical method, the top level being a virtual layer, as discussed in Section III-C. This scheme solves this benchmark in a reasonable amount of time, with acceptable error. The results are listed in Table III, and the normalized runtime is seen to be higher than solving other circuit types.

In order to evaluate the transient analysis, since we were unable to obtain real-life *RC*/*RLC* power grid circuits, we generated four circuits with realistic parameters. RC1 and RC2 listed in Table IV are *RC* networks based on the structure of Industry1. RKC1 and RKC2 listed in Table V are RKC networks based on the structure of Industry2. Inductances are assumed to be only in the top two metal layers and are estimated using formulas provided by [11]. Then $K$ matrices are constructed by the method proposed by [7], using $7\times7$ and $7\times5$ window sizes for the two metal layers. Current-load waveforms are designed such that inductive effect is visible: the simulation using a direct solver shows that if inductors in circuit RKC1 are ignored, the induced error is up to 21 mV.

TABLE IV
*RC* TRANSIENT ANALYSIS RESULTS. N IS THE CIRCUIT SIZE, TS IS THE NUMBER OF TIMESTEPS, T IS CPU TIME PER TIMESTEP FOR SUBSEQUENT TIMESTEPS, E1 IS THE AVERAGE ERROR, E2 IS THE MAX ERROR, AND P IS THE PEAK MEMORY. G DENOTES THE GENERIC RANDOM-WALK METHOD, AND S DENOTES THE SINGLE-LEVEL HIERARCHICAL METHOD

| Ckt | | N | TS | T(sec) | E1(mV) | E2(mV) | P(MB) |
|---|---|---|---|---|---|---|---|
| RC1 | G | 3.7K | 500 | 0.0026 | 1.6 | 11.9 | – |
| | S | | | 0.0014 | 2.0 | 13.7 | – |
| RC2 | G | 2.3M | 1000 | 0.65 | N/A | N/A | 680 |
| | S | | | 0.64 | N/A | N/A | 854 |

TABLE V
RKC TRANSIENT ANALYSIS RESULTS. N IS THE CIRCUIT SIZE, TS IS THE NUMBER OF TIMESTEPS, T IS CPU TIME PER TIMESTEP FOR SUBSEQUENT TIMESTEPS, E1 IS THE AVERAGE ERROR, E2 IS THE MAX ERROR, AND P IS THE PEAK MEMORY

| Ckt | N | TS | T(sec) | E1(mV) | E2(mV) | P(MB) |
|---|---|---|---|---|---|---|
| RKC1 | 6.4K | 1000 | 0.0165 | 0.8 | 13.9 | – |
| RKC2 | 642K | 1000 | 2.1 | N/A | N/A | 837 |

The results of *RC* analysis using both the generic method and the hierarchical method are shown in Table IV. CPU times are measured for the timesteps that follow the initial dc analysis and the first transient step. The solution for circuit RC1 is compared with HSPICE, while circuit RC2 is too large to be simulated in HSPICE. Note that E1 is the average over all nodes at all timesteps, and E2 is the maximum over all nodes at all timesteps. The peak memory numbers are small for RC1 and are omitted. The runtimes are several times faster than traditional direct solver runtimes reported in [39], even after normalization by the speed factor of 3. The space complexity is higher for the hierarchical method, because bookkeeping is needed not only for the bottom-metal-layer nodes, but also for building and solving the global grid. However, the peak memory of the hierarchical method is still lower than that of traditional methods reported in [39], in terms of memory consumption per million nodes.

The results of RKC analysis are shown in Table V. The single-layer hierarchical method is used, and the algorithm discussed in Section IV-B is used when solving the global grid with inductors. Note that inductances are assumed to be only in the top two metal layers, and hence only in the global grid. CPU times are measured for the timesteps that follow the initial dc analysis and the first transient step. The solution for circuit RKC1 is compared with a traditional direct solver, while circuit RKC2 is too large to be simulated by a direct solver. Note that E1 is the average over all nodes at all timesteps, and E2 is the maximum over all nodes at all timesteps. The peak memory is small for RKC1 and is omitted. Comparing with Table IV, we can see that RKC analysis has higher time and space complexity than

*RC* analysis. This is due to the extra storage for mutual inductances and the extra iterations of computation.

When viewing Tables IV and V, one common concern is error accumulation: although the error of one timestep is low, it could add up to large error over many timesteps. This concern drives us to measure E1 and E2. Note that E1 is the average over all timesteps, and E2 is the maximum over all timesteps. They suggest that the errors are acceptable after 500/1000 timesteps. Practically, they suggest that errors tend to cancel each other and that the accumulation has a very slow rate.

## VI. Conclusion and Extension

This paper presents a random-walk based power grid analyzer. A generic algorithm is first developed, and then several hierarchical methods are built to make the algorithm faster and more robust in solving various types of circuits. Capacitors and inductors are incorporated, and an RKC transient analysis algorithm is proposed. Experimental results show that these algorithms reach good runtime–accuracy tradeoffs.

### A. Potential Applications

Possible scenarios that the proposed algorithms offer advantages over traditional methods are summarized as follows.

1) *dc analysis of a whole circuit*: Random-walk algorithms provide a solution with runtime linear in circuit size. Although existing iterative solvers also have linear runtime, the random-walk solvers provide an alternative that allow users to tradeoff between speed and accuracy, and this tradeoff can be naturally tuned by changing the number of walks or the error margin. This is useful when ultra-accurate solution is not necessary, for example, early-stage performance analysis.
2) *Transient analysis of a whole circuit*: Iterative solvers are inefficient for transient analysis. Compared with direct-solver-based techniques, simulation results show that, with acceptable errors, random-walk algorithms have the following advantages: linear runtime for the initial timestep, as opposed to superlinear runtime of a direct solver, and hence they are better suited for large designs; lower memory consumption; and lower runtime for the follow-up timesteps. Again, the tradeoff between accuracy and runtime/memory consumption can be easily tuned.
3) *Solving a small number of nodes*: The generic algorithm can compute any single node voltage without solving the whole circuit and can be very useful in incremental design. This advantage holds for all chips with C4 packaging. For wire-bond packaging, it is partially compromised: a hierarchical method needs to be used, and there is an overhead of building and solving hierarchy.
4) *Parallel computing*: Random-walk based algorithms are inherently compatible with parallel computing. The computations for different nodes, and even random walks for the same node, can be carried out independently on different processors. The only communication needed between parallel processors is to share the voltages of already computed nodes, to take advantage of the speedup technique at the end of Section III-B.

### B. Potential Extension to AWE/PRIMA

The proposed algorithm can also be used to perform moment generation for power grid transient analysis in the frequency domain using asymptotic waveform evaluation (AWE) or passive reduced-order interconnect macromodeling algorithm (PRIMA). An existing stochastic moment generation approach is [21], which, in each step, randomly samples one or two capacitors and removes all others, in order to facilitate computation for large *RC* networks. Our method is different.

For transient analysis formulated by (22) (note that in *RLC* analysis, $\mathbf{y}(t)$ includes entries of inductor currents, and $C$ includes inductance entries), the moment generation is solving the following equation sets one by one [30]:

$$\begin{aligned} G\mathbf{m_0} &= [1,0,0,\cdots,0]^T \\ G\mathbf{m_i} &= -C\mathbf{m_{i-1}} \quad \text{for } i = 1,2,\cdots,k \end{aligned} \tag{37}$$

where $\mathbf{m_0}, \mathbf{m_1}, \cdots, \mathbf{m_k}$ are the moment vectors to be solved. In each iteration, the moment generation is equivalent to replacing each capacitor with a constant current source, replacing each inductor with a constant voltage source, and performing dc analysis.

The above computation can be carried out using our random-walk algorithms, with one extension to handle the voltage sources induced by inductors. In dc analysis of a circuit with a voltage source $V_0$ between two nonground non-$V_{DD}$ nodes $x$ and $y$, such that $V_x - V_y = V_0$, we combine $x$ and $y$ into one single supernode $xy$ in the random-walk game, and use $V_y$ as its nominal voltage. By Kirchoff's Current Law

$$\begin{aligned} &\sum_{i\in N_x} g_{x,i}(V_i - V_0 - V_y) + \sum_{i\in N_y} g_{y,i}(V_i - V_y) = I_x + I_y \\ V_y = &\sum_{i\in N_x} \frac{g_{x,i}}{\sum_{j\in N_x} g_{x,j} + \sum_{j\in N_y} g_{y,j}}(V_i - V_0) \\ &+ \sum_{i\in N_y} \frac{g_{y,i}}{\sum_{j\in N_x} g_{x,j} + \sum_{j\in N_y} g_{y,j}} V_i \\ &- \frac{I_x + I_y}{\sum_{j\in N_x} g_{x,j} + \sum_{j\in N_y} g_{y,j}} \end{aligned} \tag{38}$$

where $N_x$ is the set of nodes adjacent to $x$ excluding $y$, $N_y$ is the set of nodes adjacent to $y$ excluding $x$, $g_{i,j}$ is the conductance between node $i$ and node $j$, and $I_x$ and $I_y$ are the current loads connected to node $x$ and $y$. The new rule is that if the random walker goes from node $xy$ to a neighbor of $x$, he/she has to pay extra money $V_0$; correspondingly, if he/she walks from a neighbor of $x$ to node $xy$, he/she gains $V_0$. Note that if $x$ and $y$ has a common neighbor, it is considered as two different directions from node $xy$'s point of view. Under this new rule, we can perform moment computation in AWE/PRIMA using random walks. The bookkeeping technique in Section IV-C is applicable, and after solving $\mathbf{m_0}$ and $\mathbf{m_1}$, the follow-up computations can be done efficiently.

## APPENDIX I
## PROOF OF LEMMA 1

Consider the random-walk game that has $k$ terminals $h_1, h_2, \cdots, h_k$. For each individual walk in the game, the money earned at the end of the walk is composed of an award, which is a terminal voltage, minus a sequence of motel expenses. The result of the $q^{\text{th}}$ walk from node $i$ is

$$W_q = V_{\text{end } q} - u_q \tag{39}$$

where $q \in \{1, 2, \cdots, M\}$ is the index of the walk, $V_{\text{end } q} \in \{V_{h_1}, V_{h_2}, \cdots, V_{h_k}\}$ is the voltage at the terminal where the random walk ends, and $u_q$ is the sum of all motel expenses.

When we take the average of the results from the $M$ random walks, we obtain an estimated $V_i$ in the form

$$V_i = \frac{\sum_{q=1}^{M} W_q}{M} = c_1 V_{h_1} + c_2 V_{h_2} + \cdots + c_k V_{h_k} - \xi \tag{40}$$

where

$$c_j = \frac{N_j}{M}, \quad j \in \{1, \cdots, k\} \quad \text{and} \quad \xi = \frac{\sum_{q=1}^{M} u_q}{M} \tag{41}$$

where $N_j$ is the number of walks that end at terminal $h_j$. Because every random walk stops at a terminal, $N_j$'s must satisfy the following condition:

$$\sum_{j=1}^{k} N_j = M. \tag{42}$$

Therefore

$$c_1 + c_2 + \cdots + c_k = \sum_{j=1}^{k} \frac{N_j}{M} = \frac{M}{M} = 1. \tag{43}$$

## APPENDIX II
## PROOF OF EQUATIONS (33) AND (34)

In *RC* transient analysis, considering the general case where there exists capacitance $C_x$ between port $x$ and ground, then (15) becomes

$$I_x = I_1 + I_2 + \cdots + I_N + \frac{C_x}{h}\left(V_x - V_x(t-h)\right) \tag{44}$$

where the neighbors of $x$ inside the local grid are labeled $1, 2, \cdots, N$, and $I_1, I_2, \cdots, I_N$ are the currents flowing from port $x$ to each of them. By (19) and (27), and considering the extra term introduced by $C_x$, the $\gamma$ term in (32) can be expanded as follows:

$$\begin{aligned}\gamma &= -\frac{C_x}{h}V_x(t-h) + \sum_{i=1}^{N} g_i \xi_i' \\ &= -\frac{C_x}{h}V_x(t-h) \\ &\quad + \sum_{i=1}^{N} g_i \left(\xi_i - \left(c_{k+1,i}V_{h_{k+1}} + \cdots + c_{k',i}V_{h_{k'}}\right)\right)\end{aligned} \tag{45}$$

where $g_i$ is the conductance between $x$ and $i$, and $\xi_i', \xi_i, c_{k+1,i}, \cdots, c_{k',i}$ are the corresponding coefficients defined in (26) and (27) constructed for $i \in \{1, \cdots, N\}$. From (26), we know that $(k' - k)$ is the number of $V(t-h)$ terminals inside the local grid and that they are $h_{k+1}, \cdots, h_{k'}$. From (41), we have

$$c_{j,i} = \frac{N_{h_j,i}}{M}, \quad j \in \{k+1, \cdots, k'\} \tag{46}$$

$$\xi_i = \frac{\sum_{q=1}^{M} u_{q,i}}{M} \tag{47}$$

where $N_{h_j,i}$ is the number of walks from $i$ that end at terminal $h_j$, and $u_{q,i}$ is the sum of expenses paid at motels during the $q^{\text{th}}$ walk from node $i$. Substituting (46) and (47) into (45), we get

$$\begin{aligned}\gamma &= -\frac{C_x}{h}V_x(t-h) \\ &\quad + \sum_{i=1}^{N} g_i \left(\frac{\sum_{q=1}^{M} u_{q,i}}{M} - \sum_{j=k+1}^{k'} \frac{N_{h_j,i}}{M} V_{h_j}\right) \\ &= -\frac{C_x}{h}V_x(t-h) + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{q=1}^{M} u_{q,i} \\ &\quad - \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i} V_{h_j}.\end{aligned} \tag{48}$$

Define

$$\gamma_1 = \frac{\frac{C_x}{h}V_x(t-h) + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i} V_{h_j}}{\frac{C_x}{h} + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i}} \tag{49}$$

$$\gamma_2 = \sum_{i=1}^{N} \frac{g_i}{M} \sum_{q=1}^{M} u_{q,i}. \tag{50}$$

Then

$$\gamma = \gamma_2 - \gamma_1 \left(\frac{C_x}{h} + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i}\right). \tag{51}$$

By (8), $u_{q,i}$ can be further expanded, and $\gamma_2$ becomes

$$\gamma_2 = \sum_{i=1}^{N} \frac{g_i}{M} \sum_{q=1}^{M} \sum_{r} \frac{I_{q,i,r}}{s_{q,i,r}} \tag{52}$$

where $I_{q,i,r}$ is the current load flowing out of the node at the $r^{\text{th}}$ step of the $q^{\text{th}}$ walk from node $i$, and $s_{q,i,r}$ is the sum of the conductances connected to the node at the $r^{\text{th}}$ step of the $q^{\text{th}}$ walk from node $i$.

So far, we have derived (49) and (52), which are (33) and (34), respectively. These two equations have their physical meanings: $\gamma_1$ is a weighted average of some $V(t-h)$'s inside this local grid, and $\gamma_2$ is a weighted sum of some current loads inside this local grid. Now we need to show that (32) is true; in other words

$$\gamma = -\gamma_1 \left(\alpha_x + \sum_{\text{port } j \neq x} \alpha_j\right) + \gamma_2. \tag{53}$$

To do so, we look at (26) constructed for $i \in \{1, \cdots, N\}$. By Lemma 1, we have

$$c_{1,i} + \cdots + c_{k,i} + c_{k+1,i} + \cdots + c_{k',i} = 1. \tag{54}$$

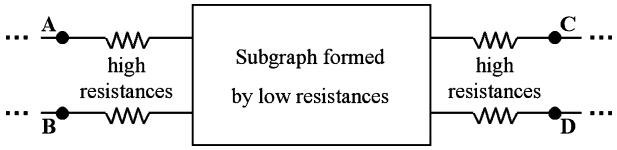



Fig. 23. Isolated subgraph formed by low resistances and isolated by high resistances.

Substituting (46) into the above equation, we get

$$1 - \sum_{j=1}^{k} c_{j,i} = \sum_{j=k+1}^{k'} \frac{N_{h_j,i}}{M}. \tag{55}$$

By (19), and considering the extra coefficient $C_x/h$ introduced by $C_x$ as shown in (44),

$$\begin{aligned}\alpha_x + \sum_{\text{port } j \neq x} \alpha_j &= \frac{C_x}{h} + \sum_{i=1}^{N} g_i (1 - c_{\text{port } x,i}) \\ &\quad + \sum_{\text{port } j \neq x} \left( - \sum_{i=1}^{N} g_i c_{j,i} \right) \\ &= \frac{C_x}{h} + \sum_{i=1}^{N} g_i \left( 1 - \sum_{j=1}^{k} c_{j,i} \right) \\ &= \frac{C_x}{h} + \sum_{i=1}^{N} g_i \sum_{j=k+1}^{k'} \frac{N_{h_j,i}}{M} \quad \text{(by (55))} \\ &= \frac{C_x}{h} + \sum_{i=1}^{N} \frac{g_i}{M} \sum_{j=k+1}^{k'} N_{h_j,i}. \end{aligned} \tag{56}$$

By (51) and (56), (53) must be true, and our proof is complete.

## Appendix III
## Remedies for Isolated Low-Resistance Subgraphs

As the general case of Fig. 6, a subgraph formed by several low resistances can be isolated by other high resistances and form a "trap," and random walks may spend many steps inside this subgraph. Fig. 23 shows an example.

It is worth pointing out that a preprocessing step to remove such a subgraph is only necessary when the ratio between the surrounding high resistances and the inside low resistances is extremely high. For example, if at every node in the subgraph, $\sum_{\text{inside}} g / \sum_{\text{outside}} g = 19$, then according to (8), at every step, the random walk stays in the subgraph with probability $\sum_{\text{inside}} g / (\sum_{\text{inside}} g + \sum_{\text{outside}} g) = 0.95$. However, $0.95^{20} < 0.36$; in other words, the probability that a random walk stays in the subgraph for more than 20 steps is less than 0.36. Therefore, in this example, removing this subgraph may not be necessary.

None of the three industrial benchmarks contains a subgraph that needs to be removed, and our implementation only removes single isolated low resistances as shown in Fig. 6, by the $Y - \Delta$ transformation. Therefore, the following discussion has not been implemented or tested. If removing a subgraph is indeed necessary, the following techniques may be employed.

1) *If this subgraph is a tree*: The $Y - \Delta$ transformation can be iteratively applied on leaf nodes. In each transformation, an edge of the subgraph, i.e., a low resistance, is removed. In the end, the subgraph disappears without any loss of accuracy.
2) *If this subgraph is not a tree, or if the previous technique introduces overly complex connectivity*: Then, using Fig. 23 as an example, we can define the subgraph as a local grid, and nodes A, B, C, and D as ports. Then, the subgraph can be replaced by a macromodel, which provides connections between A, B, C, and D without trapping random walks. This macromodeling can be carried out either by the algorithm from [39] or by our approach in Section III-A-1, without excessive loss of accuracy.

## Appendix IV
## Proof of Convergence for the Iterative Approach in Section IV-B

The modified nodal equation set for the circuit in the form of Fig. 18(b) can be written as

$$(F + H)\mathbf{X} = \mathbf{E} \tag{57}$$

where matrix $F$ contains the contributions of resistors and companion models for capacitors and self-inductances, matrix $H$ contains the contributions of the companion models (voltage-controlled current sources) for mutual inductances, $\mathbf{X}$ is the vector of node voltages, and $\mathbf{E}$ is the vector of independent sources, which include original current/voltage sources, the voltage sources from the companion models for capacitors, and the current sources from the companion models for self-inductances [14]. Note that, because the modified nodal equations are constructed for the circuit form of Fig. 18(b), $\mathbf{X}$ includes both the end nodes of wire segments (nodes $A$'s and $B$'s in Fig. 18), and the middle nodes ($C$'s in Fig. 18), which do not exist physically.

The iterative algorithm in Section IV-B can be written as

$$\begin{aligned} F\mathbf{X^{k+1}} &= -H\mathbf{X^{k}} + \mathbf{E} \\ \mathbf{X^{k+1}} &= -F^{-1}H\mathbf{X^{k}} + F^{-1}\mathbf{E} \end{aligned} \tag{58}$$

where $\mathbf{X^k}$ is the solution vector from the previous iteration, and $\mathbf{X^{k+1}}$ is the updated solution vector. Note that our algorithm does not perform the matrix computation of (58), and instead, it converts the circuit to the form of Fig. 18(d) and uses random walks to carry out the computation. However, our underlying iteration is (58).

Therefore, the necessary and sufficient condition for our iterative algorithm to converge is

$$\max_r \left| \lambda_r (F^{-1}H) \right| < 1 \tag{59}$$

where $\lambda_r$ denotes the $r^{\text{th}}$ eigenvalue of a matrix [36]. In order to prove condition (59), the following lemma is needed.

*Lemma 4:* Matrices $F$, $(F + H)$, and $(F - H)$ are positive definite.

Matrix $F$ is an irreducibly diagonally dominant matrix with positive diagonal entries, for any connected power grid, and therefore is positive definite [36].

Let matrix $F_1$ be the component of $F$ that corresponds to the contributions of resistors and companion models for capacitors, and let matrix $F_2$ be the contributions of self-inductances. Then $F = F_1 + F_2$. Let $\mathbf{y}$ be any real-valued nonzero vector. We have

$$\mathbf{y^T}(F+H)\mathbf{y} = \mathbf{y^T}F_1\mathbf{y} + \mathbf{y^T}F_2\mathbf{y} + \mathbf{y^T}H\mathbf{y}. \tag{60}$$

Because $F_1$ is a diagonally dominant matrix with positive diagonal entries (maybe reducible, i.e., representing an unconnected network), and hence must be nonnegative definite

$$\mathbf{y^T}F_1\mathbf{y} \geq 0. \tag{61}$$

Let $N$ be the number of inductors, and they are labeled $1, 2, \cdots, N$. Let $e_{i,1}$ and $e_{i,2}$ be the nodes at the two ends of inductor $i$; in other words, they are the nodes $B$ and $C$ in Fig. 18; let them be defined with consistent direction; in other words, for parallel wire segments, $e_{i,1}$'s always point to the same direction. In the $K$ matrix, $K_{i,i}$ is the self-inductance of inductor $i$, and $K_{i,j}$, $i \neq j$, is the mutual inductance between inductor $i$ and inductor $j$. From [7], [15], we know that $K_{i,j} = K_{j,i}$, and that

$$K_{i,i} > \sum_{\substack{j\in\{1,\cdots,N\}\\ j\neq i}} |K_{i,j}|. \tag{62}$$

The contribution of inductor $i$ to matrix $F_2$ is shown below, with the row and column indices marked outside [14].

$$\begin{array}{c} \\ e_{i,1} \\ e_{i,2} \end{array} \begin{array}{c} \begin{array}{cc} e_{i,1} & e_{i,2} \end{array} \\ \begin{bmatrix} hK_{i,i} & -hK_{i,i} \\ -hK_{i,i} & hK_{i,i} \end{bmatrix} \end{array}.$$

Hence, the contribution of inductor $i$ to $\mathbf{y^T}F_2\mathbf{y}$ is

$$hK_{i,i}y_{e_{i,1}}^2 - hK_{i,i}y_{e_{i,1}}y_{e_{i,2}} - hK_{i,i}y_{e_{i,1}}y_{e_{i,2}} + hK_{i,i}y_{e_{i,2}}^2 = hK_{i,i}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2.$$

Therefore

$$\mathbf{y^T}F_2\mathbf{y} = \sum_{i=1}^{N} hK_{i,i}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2. \tag{63}$$

The contribution of the mutual inductances between inductor $i$ and inductor $j$ to matrix $H$ is shown below, with the row and column indices marked outside the matrix, and these entries correspond to the voltage-controlled current sources in Fig. 17 and Fig. 18(b) [14].

$$\begin{array}{c} \\ e_{i,1} \\ e_{i,2} \\ e_{j,1} \\ e_{j,2} \end{array} \begin{array}{c} \begin{array}{cccc} e_{i,1} & e_{i,2} & e_{j,1} & e_{j,2} \end{array} \\ \begin{bmatrix} & & hK_{i,j} & -hK_{i,j} \\ & & -hK_{i,j} & hK_{i,j} \\ hK_{j,i} & -hK_{j,i} & & \\ -hK_{j,i} & hK_{j,i} & & \end{bmatrix} \end{array}.$$

Hence, the contribution of $K_{i,j}$ and $K_{j,i}$ to $\mathbf{y^T}H\mathbf{y}$ is

$$\begin{aligned} &hK_{i,j}\left(y_{e_{i,1}}y_{e_{j,1}} - y_{e_{i,1}}y_{e_{j,2}} - y_{e_{i,2}}y_{e_{j,1}} + y_{e_{i,2}}y_{e_{j,2}}\right) \\ &\quad + hK_{j,i}\left(y_{e_{i,1}}y_{e_{j,1}} - y_{e_{i,1}}y_{e_{j,2}} - y_{e_{i,2}}y_{e_{j,1}} + y_{e_{i,2}}y_{e_{j,2}}\right) \\ &\quad = hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)\left(y_{e_{j,1}} - y_{e_{j,2}}\right) \\ &\quad\quad + hK_{j,i}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)\left(y_{e_{j,1}} - y_{e_{j,2}}\right) \\ &\quad = 2hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)\left(y_{e_{j,1}} - y_{e_{j,2}}\right). \end{aligned}$$

Therefore

$$\mathbf{y^T}H\mathbf{y} = \sum_{\substack{i,j\in\{1,\cdots,N\}\\ i\neq j}} 2hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)\left(y_{e_{j,1}} - y_{e_{j,2}}\right). \tag{64}$$

This leads to

$$\begin{aligned} |\mathbf{y^T}H\mathbf{y}| &= \left|\sum_{\substack{i,j\in\{1,\cdots,N\}\\ i\neq j}} 2hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)\left(y_{e_{j,1}} - y_{e_{j,2}}\right)\right| \\ &\leq \left|\sum_{\substack{i,j\in\{1,\cdots,N\}\\ i\neq j}} hK_{i,j}\left(\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2 + \left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2\right)\right| \\ &\leq \left|\sum_{\substack{i,j\in\{1,\cdots,N\}\\ i\neq j}} hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2\right| \\ &\quad + \left|\sum_{\substack{i,j\in\{1,\cdots,N\}\\ i\neq j}} hK_{i,j}\left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2\right| \\ &= \left|\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j\in\{1,\cdots,N\}\\ j\neq i}} hK_{i,j}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2\right| \\ &\quad + \left|\frac{1}{2}\sum_{j=1}^{N}\sum_{\substack{i\in\{1,\cdots,N\}\\ i\neq j}} hK_{i,j}\left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2\right| \\ &= \frac{1}{2}\sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2\left|\sum_{\substack{j\in\{1,\cdots,N\}\\ j\neq i}} hK_{i,j}\right| \\ &\quad + \frac{1}{2}\sum_{j=1}^{N}\left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2\left|\sum_{\substack{i\in\{1,\cdots,N\}\\ i\neq j}} hK_{i,j}\right| \\ &\leq \frac{1}{2}\sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2\sum_{\substack{j\in\{1,\cdots,N\}\\ j\neq i}} |hK_{i,j}| \\ &\quad + \frac{1}{2}\sum_{j=1}^{N}\left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2\sum_{\substack{i\in\{1,\cdots,N\}\\ i\neq j}} |hK_{i,j}| \\ &\leq \frac{1}{2}\sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2 hK_{i,i} \\ &\quad + \frac{1}{2}\sum_{j=1}^{N}\left(y_{e_{j,1}} - y_{e_{j,2}}\right)^2 hK_{j,j} \end{aligned}$$

$$
\begin{aligned}
&\text{(applying equation (62))}\\
&= \frac{1}{2}\sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2 hK_{i,i}\\
&\quad + \frac{1}{2}\sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2 hK_{i,i}\\
&= \sum_{i=1}^{N}\left(y_{e_{i,1}} - y_{e_{i,2}}\right)^2 hK_{i,i}\\
&= \mathbf{y^T} F_2 \mathbf{y} \quad \text{(applying equation (63))} \qquad (65)
\end{aligned}
$$

which in turn implies

$$\mathbf{y^T} F_2 \mathbf{y} \pm \mathbf{y^T} H \mathbf{y} \geq 0. \qquad (66)$$

Substituting (61) and (66) into (60), we get

$$\mathbf{y^T}(F+H)\mathbf{y} \geq 0. \qquad (67)$$

Now we need to show that (61) and (66) cannot both be equalities. Note that, in order for (65) to be an equality after applying inequality (62), vector $\mathbf{y}$ must satisfy the condition

$$y_{e_{i,1}} = y_{e_{i,2}} \quad \text{for } i \in \{1, \cdots, N\}.$$

For such a vector $\mathbf{y}$, we can merge $e_{i,1}$ and $e_{i,2}$ into one node and obtain a shortened vector $\mathbf{y'}$. In other words, nodes $B$ and $C$ in Fig. 18 are merged into one node. Correspondingly, the rows for $e_{i,1}$ and $e_{i,2}$ in matrix $F_1$ are merged into one row by adding entries, and columns for $e_{i,1}$ and $e_{i,2}$ in matrix $F_1$ are merged into one column by adding entries. Thus, we obtain a new matrix $F_1'$, which is the same as the modified nodal left-hand-side matrix if all inductors are ignored. Because $F_1'$ is an irreducibly diagonally dominant matrix with positive diagonal entries, for any connected power grid, we have

$$\mathbf{y^T} F_1 \mathbf{y} = \mathbf{y'^T} F_1' \mathbf{y'} > 0.$$

It follows that (61) and (66) cannot both be equalities.

Thus, (67) can never be equality and can be replaced by

$$\mathbf{y^T}(F+H)\mathbf{y} > 0. \qquad (68)$$

This is true for any real-valued nonzero vector $\mathbf{y}$. Therefore, matrix $(F+H)$ is positive definite.

Similarly, by (61) and (66), and the fact that they cannot both be equalities,

$$\mathbf{y^T}(F-H)\mathbf{y} = \mathbf{y^T} F_1 \mathbf{y} + \mathbf{y^T} F_2 \mathbf{y} - \mathbf{y^T} H \mathbf{y} > 0. \qquad (69)$$

This is true for any real-valued nonzero vector $\mathbf{y}$. Therefore, matrix $(F-H)$ is positive definite. Lemma 4 is proven.

Now we move on to use Lemma 4 to prove condition (59), which is replicated as follows.

*Lemma 5:* $\max_r |\lambda_r(F^{-1}H)| < 1$

Let $\lambda$ be any eigenvalue of matrix $F^{-1}H$, and let $\mathbf{y}$ be the corresponding eigenvector. By Lemma 4, $(F+H)$ is positive definite, and we have

$$
\begin{aligned}
\mathbf{y^T}(F+H)\mathbf{y} &> 0\\
\mathbf{y^T} F (I + F^{-1}H)\mathbf{y} &> 0\\
\mathbf{y^T} F (1+\lambda)\mathbf{y} &> 0\\
(1+\lambda)(\mathbf{y^T} F \mathbf{y}) &> 0. \qquad (70)
\end{aligned}
$$

By Lemma 4, $\mathbf{y^T} F \mathbf{y}$ must be a positive scalar, and therefore

$$
\begin{aligned}
1+\lambda &> 0\\
\lambda &> -1. \qquad (71)
\end{aligned}
$$

Similarly, from $(F-H)$ being positive definite, we get

$$\lambda < 1. \qquad (72)$$

Therefore

$$|\lambda| < 1. \qquad (73)$$

This is true for any eigenvalue of matrix $F^{-1}H$. Therefore Lemma 5 is true, and our iterative algorithm in Section IV-B is guaranteed to converge.

## ACKNOWLEDGMENT

The authors would like to thank H. Su for help with the benchmark circuits, N. Shenoy for introducing the last author to the Doyle and Snell monograph many years ago, and the six reviewers for their contributions in revising this manuscript.

## REFERENCES

[1] R. M. Bevensee, "Probabilistic potential theory applied to electrical engineering problems," *Proc. IEEE*, vol. 61, no. 4, pp. 423–437, Apr. 1999.

[2] S. Bodapati and F. N. Najm, "High-level current macro-model for power grid analysis," in *Proc. Design Automation Conf. (DAC)*, 2002, pp. 385–390.

[3] A. Brambilla and P. Maffezzoni, "Statistical method for the analysis of interconnects delay in submicron layouts," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 20, no. 8, pp. 957–966, Aug. 2001.

[4] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *Proc. Design Automation Conf. (DAC)*, 1997, pp. 638–643.

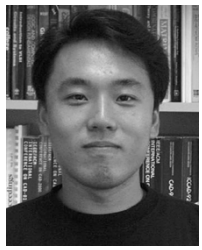
[5] T. Chen and C. C. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. Design Automation Conf. (DAC)*, 2001, pp. 559–562.

[6] J. H. Curtiss, "Sampling methods applied to differential and difference equations," in *Proc. IBM Seminar on Scientific Computation*, 1949, pp. 87–109.

[7] A. Devgan, H. Ji, and W. Dai, "How to efficiently capture on-chip inductance effects: Introducing a new circuit element K," in *Proc. Int. Conf. Computer Aided Design (ICCAD)*, 2000, pp. 150–155.

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

[9] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*. Washington, DC: Mathematical Assn. of America, 1984.
[10] G. E. Forsythe and R. A. Leibler, "Matrix inversion by a Monte Carlo method," *Math. Tables Other Aids Computation*, vol. 4, no. 31, pp. 127–129, Jul. 1950.
[11] F. W. Grover, *Inductance Calculations*. New York: Dover, 1954.
[12] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*. London, U.K.: Methuen & Co., Ltd., 1964.
[13] R. Hersh and R. J. Griego, "Brownian motion and potential theory," *Sci. Amer.*, pp. 67–74, Mar. 1969.
[14] C. Ho, A. E. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-22, no. 6, pp. 504–509, Jun. 1975.
[15] H. Ji, A. Devgan, and W. Dai, "KSim: A stable and efficient RKC simulator for capturing on-chip inductance effect," in *Proc. Asia South Pacific Design Automation Conf.*, 2001, pp. 379–384.
[16] R. Jiang, T. Chen, and C. C. Chen, "PODEA: Power delivery efficient analysis with realizable model reduction," in *Proc. Int. Symp. Circuits Systems*, vol. 4, 2003, pp. 608–611.
[17] C. N. Klahr, "A Monte Carlo method for the solution of elliptic partial differential equations," in *Mathematical Methods for Digital Computers*. New York: Wiley, 1962, ch. 14.
[18] A. W. Knapp, "Connection between Brownian motion and potential theory," *J. Math. Anal. Appl.*, vol. 12, pp. 328–349, 1965.
[19] J. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 21, no. 10, pp. 1148–1160, Oct. 2002.
[20] Y. L. Le Coz and R. B. Iverson, "A stochastic algorithm for high speed capacitance extraction in integrated circuits," *Solid-State Electron.*, vol. 35, no. 7, pp. 1005–1012, Jul. 1992.
[21] Y. L. Le Coz, D. Krishna, D. M. Petranovic, W. M. Loh, and P. Bendix, "A sum-over-paths impulse-response moment-extraction algorithm for IC-interconnect networks: Cerification, coupled RC Lines," in *Proc. Int. Conf. Computer Aided Design (ICCAD)*, 2003, pp. 665–670.
[22] M. E. Muller, "Some continuous Monte Carlo methods for the Dirichlet problem," *Annals Math. Statistics*, vol. 27, pp. 569–589, 1956.
[23] R. Panda, D. Blaauw, R. Chaudhury, V. Zolotov, B. Young, and R. Ramaraju, "Model and analysis for combined package and on-chip power grid simulation," in *Proc. Int. Symp. Low Power Electronics Design*, 2000, pp. 179–184.
[24] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. New York: Cambridge Univ., 1994.
[25] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Random walks in a supply network," in *Proc. Design Automation Conf. (DAC)*, 2003, pp. 93–98.
[26] H. Qian and S. S. Sapatnekar, "Hierarchical random-walk algorithms for power grid analysis," in *Proc. Asia South Pacific Design Automation Conf.*, 2004, pp. 499–504.
[27] Z. Qin and C. Cheng, "Realizable parasitic reduction using generalized Y-$\Delta$ transformation," in *Proc. Design Automation Conf. (DAC)*, 2003, pp. 220–225.
[28] G. M. Royer, "A Monte Carlo procedure for potential theory problems," *IEEE Trans. Microw. Theory Tech.*, vol. 19, no. 10, pp. 813–818, Oct. 1971.
[29] S. S. Sapatnekar and H. Su, "Analysis and optimization of power grids," *IEEE Design Test Computers*, vol. 20, no. 3, pp. 7–15, May/Jun. 2003.
[30] J. C. Shah, A. A. Younis, S. S. Sapatnekar, and M. M. Hassoun, "An algorithm for simulating power/ground networks using Padé approximations and its symbolic implementation," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Processing*, vol. 45, no. 10, pp. 1372–1382, Oct. 1998.
[31] SPEC CPU2000 Results [Online]. Available: http://www.specbench.org/cpu2000/results/cpu2000.html
[32] A. Srinivasan and V. Aggarwal, "Stochastic linear solvers," presented at the SIAM Conf. Applied Linear Algebra, Williamsburg, VA, Jul. 15–19, 2003.
[33] G. Steele, D. Overhauser, S. Rochel, and S. Z. Hussain, "Full-chip verification methods for DSM power distribution systems," in *Proc. Design Automation Conf. (DAC)*, 1998, pp. 744–749.
[34] H. Su, K. H. Gala, and S. S. Sapatnekar, "Fast analysis and optimization of power/ground networks," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 2000, pp. 477–480.
[35] C. J. K. Tan and M. F. Dixon, "Antithetic Monte Carlo linear solver," in *Proc. Int. Conf. Computational Science*, 2002, pp. 383–392.
[36] R. S. Varga, *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1962.
[37] W. Wasow, "A note on the inversion of matrices by random walks," *Math. Tables Other Aids Computation*, vol. 6, no. 38, pp. 78–81, Apr. 1952.
[38] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*. New York: Wiley, 1999.
[39] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 21, no. 2, pp. 159–168, Feb. 2002.



**Haifeng Qian** received the B.E. degree in microelectronics from Tsinghua University, Beijing, China, in 2000, and the M.S. degree in electrical engineering from the University of Texas at Dallas in 2002. Since August 2002, he has been working toward the Ph.D. degree at the University of Minnesota, Minneapolis.

In summer 2004, he worked at Synopsys, Sunnyvale, CA.

Mr. Qian received a Best Paper Award at the Design Automation Conference (DAC) 2003.

**Sani R. Nassif** (S'81–M'83–SM'03) received the Ph.D. from Carnegie Mellon University, Pittsburgh, PA, during the 1980s.

He worked for ten years at Bell Laboratories on various aspects of design and technology coupling, including device modeling, parameter extraction, worst-case analysis, design optimization, and circuit simulation. In 1996, he joined the IBM Austin Research Laboratory, Austin, TX, where he is presently managing the Tools and Technology Department, which is focused on design/technology coupling, timing simulation and analysis, testing, low-power design, and thermoelectric cooling.

**Sachin S. Sapatnekar** (S'86–M'93–SM'99–F'03) received the B.Tech. degree from the Indian Institute of Technology, Bombay, in 1987, the M.S. degree from Syracuse University, Syracuse, NY, in 1989, and the Ph.D. degree from the University of Illinois at Urbana–Champaign in 1992.

From 1992 to 1997, he was an Assistant Professor in the Department of Electrical and Computer Engineering at Iowa State University, Ames. He is currently the Robert and Marjorie Henle Professor in the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis. He is an author of four books and a coeditor of one volume, and has published mostly in the areas of timing and layout.

Dr. Sapatnekar has held positions on the editorial board of the IEEE TRANSACTIONS ON VLSI SYSTEMS, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, the IEEE TRANSACTIONS ON CAD, and has been a Guest Editor for the latter. He has served on the Technical Program Committee for various conferences, and as Technical Program and General Chair for the Tau workshop and for the International Symposium on Physical Design. He has been a Distinguished Visitor for the IEEE Computer Society and a Distinguished Lecturer for the IEEE Circuits and Systems Society. He is a recipient of the NSF Career Award, three best paper awards at the Design Automation Conference (DAC) and one at the International Conference on Computer Design (ICCD), and the Semiconductor Research Corporation (SRC) Technical Excellence award.