Hierarchical Random-walk Algorithms for Power Grid Analysis [*]

Haifeng Qian
Department of ECE
University of Minnesota
Minneapolis, MN 55414
qianhf@ece.umn.edu

Sachin S. Sapatnekar
Department of ECE
University of Minnesota
Minneapolis, MN 55414
sachin@ece.umn.edu

Abstract— This paper presents a power grid analyzer that combines a divide-and-conquer strategy with a random-walk engine. A single-level hierarchical method is first described and then extended to multi-level and "virtual-layer" hierarchy. Experimental results show that these algorithms not only achieve speedups over generic random-walk method, but also are more robust in solving various types of industrial circuits. For example, a 71K-node circuit is solved in 4.16 seconds, showing a more than 4 times speedup over the generic method; a 348K-node wire-bond power grid, for which the performance of the generic method degrades, is solved in 75.88 seconds.

I. Introduction

A reliable power grid is an important part of high-performance VLSI design, and accurate and efficient analysis to check for signal integrity is becoming a critical issue in nanometer designs.

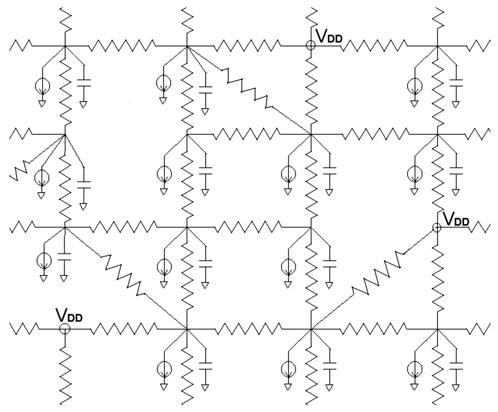

Fig. 1. Part of a typical power grid model.

Fig. 1 shows a power grid model consisting of wire resistances, wire capacitances, decoupling capacitors, VDD pads, and current sources that represent the currents drawn by logic gates or functional blocks. There are two sub-problems to power grid analysis: *DC analysis* and *transient analysis.* In this paper, the presentation will focus on DC analysis, but we will show how the method can be applied to transient analysis by pointing out the differences, where applicable. The DC analysis problem is formulated as:

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

where G is the conductance matrix for the interconnected resistors, $\mathbf{X}$ is the vector of node voltages, and $\mathbf{E}$ is a vector of independent sources. Solving this set of linear equations can become prohibitive for a modern-day power grid with tens of millions of nodes. Several methods have been proposed to overcome this and achieve an acceptable time and space computational complexity by sacrificing some accuracy [3][7].

A novel statistical algorithm based on the relationship between random walks and electrical networks is proposed in [4], and demonstrates an excellent accuracy-runtime tradeoff. Its complexity is linear in circuit size, and it has the feature of localizing computation, which makes it especially useful when only part of the grid is to be analyzed. The work in [4] addresses both the DC analysis and RC network transient analysis problems.

In this paper, we combine the divide-and-conquer idea of [7] with the generic random-walk method of [4], and present a *hierarchical random-walk method.* We use test results to show that it is faster and more robust than generic random-walk method. In particular, it solves certain circuit types for which the performance of the generic method is limited. This method also provides a framework that can perform transient analysis of RLC models.

II. The Generic Random Walk Method for Power Grid Analysis

For completeness, we now present a theoretical summary of [4], which is based on [1]. For a single node x in the DC analysis of a VDD net, its voltage can be expressed as follows:

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

where the nodes adjacent to x are labeled $1, 2, \cdots, degree(x)$, 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

[*]This research was supported in part by the SRC under contract 2003-TJ-1092, by the NSF under award CCR-0205227, and by an IBM Faculty Award.

node x, and I_x is the current load connected to node x. For a power grid with N non-VDD nodes, there are N linear equations similar to the one above, one for each node. Solving this set of equations gives the exact voltages.

A random walk "game" can be formulated similarly. Given a finite undirected connected graph representing a street map. A walker starts from one of the nodes, and goes to an adjacent node k every day with probability $p_{x,k}$ for $k = 1, 2, \cdots, degree(x)$, where x is the current node, and $degree(x)$ is the number of edges connected to node x. The sum of these probabilities is one. 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, his/her journey is complete and he/she will be awarded a certain amount of money, m_0. The problem is to find the gain function:

$$f(x) = E[\text{total money earned } | \text{walk starts at node } x] \quad (3)$$

It is obvious that

$$f(\text{one of the homes}) = m_0 \quad (4)$$

For a non-home node x, assuming that the adjacent nodes of x are labeled $1, 2, \cdots, degree(x)$, the f variables satisfy

$$f(x) = \sum_{i=1}^{degree(x)} p_{x,i} f(i) - m_x \quad (5)$$

For a random-walk problem with N non-home nodes, there are N linear equations similar to the one above, and the solution to this set of equations will give the exact values of f at all nodes.

Equation (5) becomes identical to (2), and Equation (4) reduces to the condition of perfect VDD nodes if

$$p_{x,i} = \frac{g_i}{\sum_{j=1}^{degree(x)} g_j} \qquad i = 1, 2, \cdots, degree(x)$$

$$m_x = \frac{I_x}{\sum_{j=1}^{degree(x)} g_j} \qquad m_0 = V_{DD} \qquad f(x) = V_x \quad (6)$$

In other words, for any DC power-grid analysis, a random walk problem can be constructed that is mathematically equivalent. The work in [4] proposes to perform a certain number of walks and uses the average money left in those experiments as the approximated voltage, using a convergence criterion base on the Central Limit Theorem [6]. The calculations for ground net analysis are analogous. When solving for multiple node voltages, each calculated node becomes a new home in the game with an award amount equal to its calculated voltage. This operation improves both speed and accuracy [4].

For transient analysis of RC supply grids, the backward Euler approximation with a timestep of h can be viewed as replacing each capacitor by a resistor and a voltage source. Thus the rules of the random walk game are changed to accommodate the changes. 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

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

where g_j is as defined in Equation (2), and C_x is the capacitance between node x and ground. By a bookkeeping technique proposed in [4], in follow-up timesteps after the first transient step, random walks are no longer needed, and computation can be done efficiently.

The above generic random-walk algorithm proposed in [4] shows excellent performance for many circuits, but it has been found that it could require large runtimes for certain circuit types:

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 center node takes a huge number of steps to terminate. In general, for a large graph with very few homes, runtime is high.

2. In certain power grids, wire resistances in a metal layer are significantly larger than the vias connecting this layer to the next metal layer beneath it. Because the random walker 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. 2 illustrates this effect.

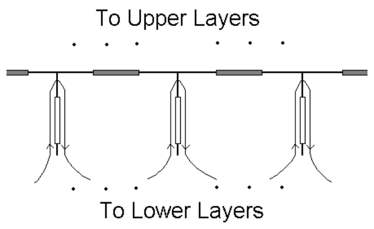

Fig. 2. The barrier effect. Grey rectangles represent high wire resistances, white rectangles represent low via resistances, and curves show the route of the random walker when he/she attempts to approach the top metal layer.

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

III. Hierarchical Random Walks

A. Principles

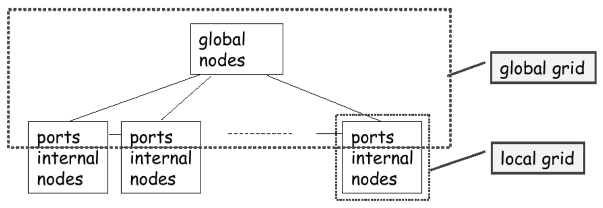

Fig. 3. Hierarchical strategy in [7].

The hierarchical strategy in [7] is illustrated in Fig. 3. The whole power grid is divided into a global net and multiple local nets, and interfacing nodes are defined as ports. From the global perspective, the behavior of a local grid is completely described by the following equation.

$$\mathbf{I_{ports}} = A\mathbf{V_{ports}} + \mathbf{S} \tag{7}$$

where $\mathbf{I_{ports}}$ is the vector of currents flowing from the global net into this local net, $\mathbf{V_{ports}}$ is the vector of port voltages, A is a square matrix, $\mathbf{S}$ is a constant vector. The algorithmic flow of [7] is: first, macro-models, i.e., the A matrices and $\mathbf{S}$ vectors, are extracted from local nets; next, the linear equation set of the global net is solved and port voltages obtained; finally, local nets are solved individually.

A.1. Constructing A and $\mathbf{S}$

An exact method for calculating A and $\mathbf{S}$ is provided in [7], and 0-1 integer linear programming 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.

If we look at one port x, the current flowing from the global net into the local net is

$$I_x = I_1 + I_2 + \cdots + I_N \tag{8}$$

where the neighbors of x inside the local net 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 node,

$$I_i = g_i V_x - g_i V_i \tag{9}$$

where V_x and V_i are voltages at node x and node i respectively, and g_i is the conductance between x and i.

Lemma 1 *If node i is an internal node, and we run M random walks from node i, with the ports being the terminals at which the walk ends, then the estimated V_i is in the following form:*

$$V_i = c_1 V_{port\,1} + c_2 V_{port\,2} + \cdots + c_k V_{port\,k} - \kappa \tag{10}$$

where κ is a constant, $V_{port\,j}$ is the voltage at a port node, and coefficients have the following format

$$c_j = \frac{\textit{number of walks ending at port j}}{M} \tag{11}$$

satisfying the relationship

$$c_1 + c_2 + \cdots + c_k \leq 1 \tag{12}$$

which is an exact equality for DC analysis, but is a strict inequality for transient analysis.

The proof of Lemma 1 comes from the procedure of random-walk estimation. For each individual walk, the money left in the end is composed of an award minus a sequence of motel expenses. This award could be a port voltage, if this walk ends at a port node; although the value of this voltage is unknown at this time, we keep a count for it. Alternatively, if this walk ends at a $V(t-h)$ connection of certain node during transient analysis, the award is the corresponding $V(t-h)$. After M walks, we sum up all results and divide the sum by M. The contribution of port voltage j is therefore given by equation (11), the percentage of walks that end at port j. Of course, one of the k ports is x itself. The κ term in (10) is the accumulated motel expenses divided by M, minus the contribution of $V(t-h)$'s. In DC analysis, there are no connections to $V(t-h)$ sources, and each walk must reach a port, therefore (12) is an exact equality. In transient analysis, due to the possibility of ending random walks at the $V(t-h)$ nodes, (12) is an inequality.

The current from port x to this internal node i is

$$\begin{aligned} I_i &= g_i V_{port\,x} - g_i V_i \\ &= g_i(1 - c_{port\,x}) V_{port\,x} - g_i \sum_{port\,j \neq x} c_j V_{port\,j} \\ &\quad + g_i \kappa \end{aligned} \tag{13}$$

Equation (9) can be viewed as a special case of (13). Substituting equations (9) and (13) into (8), we obtain

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

$$\begin{aligned} \text{where }\ \alpha_j &= -\sum_{i=1}^{N} g_i c_j(node\,i) \ \text{ for } j \neq x \\ \alpha_x &= \sum_{i=1}^{N} g_i (1 - c_{port\,x}(node\,i)) \\ \gamma &= \sum_{i=1}^{N} g_i \kappa(node\,i) \end{aligned} \tag{14}$$

This is what we need: $\alpha_1, \alpha_2, \cdots, \alpha_k$ is a row in matrix A, and the γ term is an element in vector $\mathbf{S}$. By doing this for every port node, we can construct A and $\mathbf{S}$.

The sparsity of the A matrix is controlled by the number of random walk experiments, M. When equation (10) is constructed, because of relationship shown in equation (11), insignificant items are dropped, and this leads to the sparsity of A. The larger M is, the more entries of equation (10) are kept and the denser A is.

Lemma 2 *The estimated A matrix is diagonally dominant.*

From equation (14), all weights of the port voltages are negative except for α_x. From equation (12), we get

$$1 - c_{port\,x} \geq \sum_{port\,j \neq x} c_j \tag{15}$$

Applying this inequality and equation (14), we get

$$\begin{aligned} \alpha_x &\geq \sum_{i=1}^{N} g_i \sum_{port\,j \neq x} c_j(node\,i) \\ &= \sum_{port\,j \neq x} \sum_{i=1}^{N} g_i c_j(node\,i) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{port\, j\neq x} (-\alpha_j) \\
&= \sum_{port\, j\neq x} |\alpha_j| \qquad (16)
\end{aligned}
$$

Thus we have proven Lemma 2. Equation (16) is an equality for DC analysis, and a strict inequality for transient analysis. 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 Section 3.A.2, this is not a barrier to solving the global net.

A.2. Solving the global net

Now we move on to step 2, solving the global net based on the extracted macromodels. In order to do so using random walks, we interpret each macromodel as an imaginary circuit, by converting equation (14) as follows

$$
\begin{aligned}
I_x &= \sum_{port\, j\neq x} (-\alpha_j)(V_x - V_{port\, j}) \\
&\quad +(\alpha_x + \sum_{port\, j\neq x} \alpha_j)V_x + \gamma \\
&= \sum_{port\, j\neq x} (-\alpha_j)(V_x - V_{port\, j}) \\
&\quad +(\alpha_x + \sum_{port\, j\neq x} \alpha_j)(V_x - \gamma_1) + \gamma_2 \qquad (17)
\end{aligned}
$$

There are many ways to split γ into γ_1 and γ_2. One of them is very meaningful: γ_1 is a weighted average of some $V(t-h)$'s inside this local net, γ_2 is a weighted sum of some current-loads inside this local net, as shown below.

$$
\gamma_1 = \frac{\frac{C_x}{h}V_x(t-h) + \sum_{i=1}^{N}\frac{g_i}{M}\sum_{m_c(i)\, ends} V_{end}(t-h)}{\frac{C_x}{h} + \sum_{i=1}^{N}\frac{g_i m_c(i)}{M}} \qquad (18)
$$

$$
\gamma_2 = \sum_{i=1}^{N}\frac{g_i}{M}\sum motel\, price \qquad (19)
$$

where the neighbors of x inside the local net are labeled $1, 2, \cdots, N$, C_x is capacitance between port x and ground, g_i is conductance between x and i, and $m_c(i)$ is the number of walks from node i and ending at some $V(t-h)$. (18) and (19) can be proven by investigating the constant items in (10) and (14). It is straightforward and is omitted due to space limitations.

Equation (17) can be viewed as a circuit, in which $(-\alpha_j)$ conductance connects node x and node j, $(\alpha_x + \sum_{port\, j\neq x}\alpha_j)$ conductance connects node x to a voltage source with voltage equal to γ_1, and an independent current source γ_2 flows out of node x. The connection $(\alpha_x+\sum_{port\, j\neq x}\alpha_j)$ only exists for transient analysis. This is an imaginary circuit, because each resistor only exists for one direction (corresponding to the asymmetry of the computed A matrix), i.e., conductance from node x to node j could be different from conductance from node j to node x. Fig. 4 illustrates this imaginary circuit composed of directed resistors.

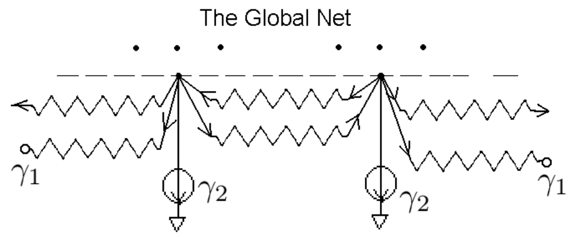

Fig. 4. The imaginary circuit interpretation of a macromodel.

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

A.3. Solving the local nets

Next, we move on to step 3, solving the bottom layer nodes in each local net, based on the port voltages computed in step 2. The ports correspond to "homes" in this random walk game, and each walk from bottom layer typically ends within a reasonably small number of steps.

B. Benefits of Hierarchy

A partitioner is required by the approach in [7], but is not necessary in our method. Given a power grid, we choose a layer of vias as the border between the global net and the local net, define the upper ends of these vias as ports, and the lower ends as the internal neighbors of the ports. In practice, we choose this layer of vias such that the global net is roughly 10% of the entire circuit size.

The hierarchical method is faster than generic random-walk algorithm. When solving the global net, each random walk starts from a port and ends at a perfect voltage source; when solving the local net, 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. Although we pay the overhead of building macromodels, the overall savings dominate this cost.

The hierarchical method is more robust than the generic random-walk algorithm. As illustrated in Fig. 2, 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 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 discussed in the previous paragraph.

A more general problem of generic random walk is that, in a large graph with very few homes, the runtime is high.

One example is wire-bond power grids. 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, although mutual inductances are not directly handled by the random-walk engine, if their effects are limited to the top few metal layers, the hierarchical method can abstract the lower metal layers into a macromodel, and pass the global net, with 10% the size of the original circuit, to a traditional direct solver that can handle mutual inductances.

C. Variations of Hierarchy

A natural extension of the proposed algorithm is to use multi-level 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 net, then port voltages are solved in a top-down order, and the bottom layer voltages are obtained in the end. Since single-level hierarchical method is better than generic method, we expect the multi-level method to be even faster and more robust. Test results in Section 4 show that the multi-level method has a similar accuracy-runtime tradeoff as the single-level method.

Another extension of the proposed algorithm leads to the concept of a "virtual-layer," when we choose ports such that the global net 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 [3]. When we abstract all connections of these ports into a macromodel, this macromodel provides imaginary connections between ports, and the global net is totally composed of such virtual connections, as illustrated in Fig. 5.

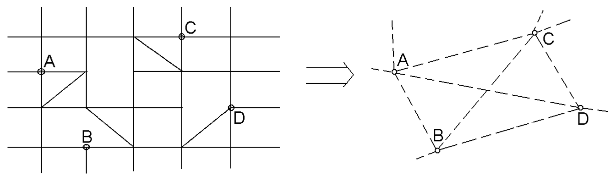

Fig. 5. The original graph with ports marked, and the extracted virtual layer.

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 huge number of steps. We traverse the graph and mark one port in every 10 nodes. Here 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 need to 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 net, i.e., the original graph, and because there are solved ports all over the graph, it can be solved efficiently.

IV. Experimental Results

In this section, we use three industrial benchmarks to evaluate the proposed hierarchical algorithms for DC analysis. Then, randomly generated circuits (whose structure is similar to real-life circuits) are used to test the performance of transient analysis.

Our first benchmark is a 70729-node circuit, and the voltage range of VDD bottom layer is 1.1324–1.1917V. The second benchmark has 218947 nodes. The voltage range of VDD bottom layer is 1.61248–1.79822V, that of GND bottom layer being 0.000334–0.066505V. The third benchmark is a wire-bond power grid with 347566 nodes. It is GND net only, and the bottom layer has a voltage range of 0.024347–0.110860. Computations are carried out on a Linux workstation with 2.8GHz CPU frequency.

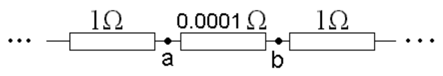

Fig. 6. An isolated low resistance forms a "trap."

One implementation 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 node a and b. Our algorithm employs a pre-processing step to detect such isolated low resistances and use the Y-Δ transformation to remove them without losing accuracy.

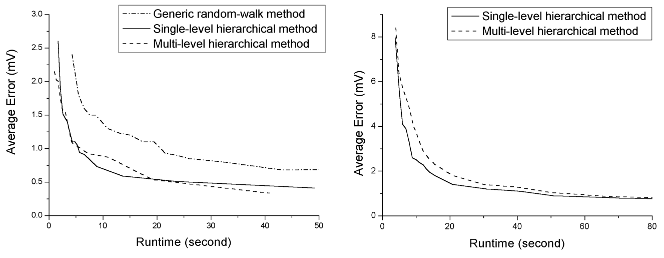

Fig. 7. Accuracy-runtime tradeoff curves for solving (a) the first benchmark, using the generic random-walk method, the single-level hierarchical method, and the multi-level hierarchical method; (b) the second benchmark, using the single-level hierarchical method and the multi-level hierarchical method.

Fig. 7(a) plots the tradeoff between average error and runtime solving the first benchmark, and both hierarchical methods achieve roughly 3–4 times speedup over the generic method, with the same average error.

Fig. 7(b) plots the tradeoff for the second benchmark. Data for the generic random-walk method are omitted because its runtime is unacceptably high for this circuit. The reason has been discussed in Section 3: 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 hierarchical methods.

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 curve, and list these points in Table I.

TABLE I
Runtime comparison. N is circuit size, E1 is average error, E2 is max error, T is runtime, and NT is normalized runtime, defined as runtime per thousand nodes. G denotes the generic random-walk method, S denotes the single-level hierarchical method, and M denotes the multi-level hierarchical method.

Benchmark		N	E1(mV)	E2(mV)	T	NT(sec)
Industry1	G	71K	1.1	9.8	17.40 sec	0.245
	S		1.1	6.6	4.34 sec	0.061
	M		1.1	9.4	4.16 sec	0.059
Industry2	S	219K	1.4	30.7	20.82 sec	0.095
	M		1.4	35.3	30.12 sec	0.138
Industry3		348K	3.1	24.7	75.88 sec	0.22
Chip2 by the method of [7]		*2.7M*	*N/A*	*N/A*	*25 min*	*0.56*

The runtime comparison is shown in Table I. The numbers for chip2 in [7] are listed as a baseline, noting that our computer is approximately 3 times faster than those used by [7], according to SPEC benchmarks [5]. Runtimes reported by [7] show superlinear time complexity. Since the time complexity of random-walk algorithms is linear in circuit size (for circuits with similar structure) [4], as power grid size increases, they will outperform [7] more.

The multi-level hierarchical method does not show a runtime advantage over the single-level method for the second benchmark. The reason is the overhead of building multiple macromodels for relatively small circuits. However, we believe that the multi-level method is potentially a more robust algorithm, and the third circuit is a case where it provides visible benefits.

The third benchmark 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 80K nodes, yet there are only 20 perfect voltage sources distributed on four sides of the top metal layer. Thus it requires high runtime if using the single-level method. We employ a two-level hierarchical method, the top level being a virtual layer, as discussed in Section 3.C. This scheme solves this benchmark in a reasonable amount of time, with acceptable error. Results are listed in Table I, and the normalized runtime is 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 randomly generated two circuits with realistic parameters. The results of our approach and the generic method are shown in Table II. CPU times are measured for the timesteps that follow the initial DC analysis and the first transient step. The solution for circuit 1 is compared with HSPICE, while circuit 2 is too large to be simulated in HSPICE. The runtimes are several times faster than traditional direct solver runtimes reported in [7], 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-layer nodes, but also for building and solving the global net. However, the peak memory of the hierarchical method is still lower than that of traditional methods reported in [7], in terms of memory consumption per million nodes.

TABLE II
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)
#1	G	3700	500	2.6m	1.6	11.9	N/A
	S			1.4m	2.0	13.7	N/A
#2	G	2.3M	1000	0.65	N/A	N/A	680
	S			0.64	N/A	N/A	854

V. Conclusion

A random-walk based hierarchical power grid analysis algorithm is proposed in this paper. It is shown to be faster and more robust than generic random-walk algorithm, and shows good accuracy-runtime tradeoffs for different types of power grids.

Acknowledgments

Our thanks go to Sani R. Nassif and Haihua Su for help with the benchmark circuits.

References

[1] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*, Mathematical Association of America, Washington DC, 1984.

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

[3] J. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Transactions on Computer-Aided Design*, vol. 21, no. 10, pp. 1148-1160, 2002.

[4] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Random Walks in a Supply Network," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 93-98, 2003.

[5] SPEC CPU2000 Results, available at http://www.specbench.org/cpu2000/results/cpu2000.html

[6] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers*, John Wiley and Sons, New York, 1999.

[7] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 2, pp. 159 -168, Feb. 2002.