More Realistic Power Grid Verification Based on Hierarchical Current and Power Constraints

[2]Chung-Kuan Cheng, [2]Peng Du, [2]Andrew B. Kahng, [1]Grantham K. H. Pang,
§[1]Yuanzhe Wang, [1]Ngai Wong
[1] Dept of Electrical & Electronic Engineering, The University of Hong Kong, Hong Kong
{gpang,yzwang,nwong}@eee.hku.hk
[2] Dept of Computer Science & Engineering, University of California, San Diego, La Jolla, CA
{ckcheng,pedu,abk}@ucsd.edu *

ABSTRACT

Vectorless power grid verification algorithms, by solving linear programming (LP) problems under current constraints, enable worst-case voltage drop predictions at an early design stage. However, worst-case current patterns obtained by many existing vectorless algorithms are time-invariant (i.e., are constant throughout the simulation time), which may result in an overly pessimistic voltage drop prediction. In this paper, a more realistic power grid verification algorithm based on hierarchical current and power constraints is proposed. The proposed algorithm naturally handles general RCL power grid models. Currents at different time steps are treated as independent variables and additional power constraints are introduced; this results in more realistic time-varying worst-case current patterns and less pessimistic worst-case voltage drop predictions. Moreover, a sorting-deletion algorithm is proposed to speed up solving LP problems by utilizing the hierarchical constraint structure. Experimental results confirm that worst-case current patterns and voltage drops obtained by the proposed algorithm are more realistic, and that the sorting-deletion algorithm reduces runtime needed to solve LP problems by $> 85\%$.

Categories and Subject Descriptors: B.7.2 [Design Aids]: Simulation
General Terms: Algorithms.

Keywords:

Power grid, worst-case voltage drop, hierarchical current and power constraints, sorting-deletion algorithm

1. INTRODUCTION

With decreasing feature size and increasing complexity of integrated circuits, IR and LdI/dt voltage drops on power grids are becoming increasingly significant, which may result in longer gate delays and logic errors. Thus, power grid verification is becoming an indispensable procedure to guarantee a functional and robust chip design. However, the extremely large size of power grid models (from tens of thousands to millions of nodes or circuit elements) renders traditional simulation tools such as SPICE inefficient. Much work has been done to find efficient methods for power grid simulation and optimization [1, 4, 8, 9, 12–14, 17].

Most existing power grid verification algorithms fall into the category of time-domain simulation. These algorithms model power grids as RC(L) circuits and model currents drawn by transistors and logic gates as ideal time-varying current sources. Nodal voltages can be solved given the waveforms of current sources [1,8,9,12,17]. Yet, such methods are not always feasible for two reasons. First, there may exist too many current sources, with each current source having various patterns. Hence it is expensive to determine which patterns result in the worst-case voltage drop. Second, one may wish to perform an early-stage power grid verification before the design of specific functional blocks, in which case current waveform information is unknown.

To facilitate early-stage power grid verification, a class of *vectorless* algorithms has been proposed [2, 3, 5–7, 11, 15, 16]. These algorithms determine the worst-case voltage drop by solving linear programming (LP) problems under a set of current constraints. The vectorless method is based on DC analysis in [7] and is extended to transient analysis in [3, 6]. An approximate matrix inversion method and a convex dual algorithm are proposed in [5] and [15], respectively, to speed up the LP solution. An impulse response-based approach considering the transition time of current sources is presented in [2]. Some vectorless algorithms rely on the assumption that the system matrix is an M-matrix [10] and hence cannot handle power grid models with inductors. Additionally, some methods assume constant current patterns; since such patterns may keep their peak values throughout the simulation period, voltage drop prediction may be overly pessimistic.

In this paper, we propose a novel algorithm, based on hierarchical current and power constraints, which generates more realistic time-varying current patterns and provides less pessimistic voltage drop predictions. Our main contributions are as follows.

1. The proposed algorithm is based on modified nodal analysis (MNA) and thereby naturally handles general RCL power grid models.

2. Currents at different time steps are treated as independent variables in LP problems, and additional power constraints are introduced which restrict the energy consumption of a current source. The current patterns solved are time-varying and do not stay at their peak values all the time. Consequently the proposed algorithm generates more realistic current patterns and provides less pessimistic voltage drop predictions.

*§ Author to whom correspondence should be addressed.

ISPD'11, March 27–30, 2011, Santa Barbara, California, USA.

3. A sorting-deletion algorithm is proposed which exploits hierarchical constraint structure for greater efficiency than standard LP methods. The time needed to solve the LP problems is reduced by $> 85\%$.

The paper is organized as follows. Background is introduced in Section 2. Hierarchical current and power constraints and LP optimization problem formulations are proposed in Section 3. Problem reductions and a sorting-deletion algorithm are proposed in Section 4. Experimental results are given in Section 5, and Section 6 draws conclusions.

2. BACKGROUND

2.1 RCL Power Grid Model

A typical 3D power grid consists of several metal layers, with each layer containing either horizontal or vertical conductors. Conductors in different layers are connected to each other by vias at their intersection points. External power supplies are connected to conductors of the top layer. Power drains, such as logic gates, transistors and memory units, are connected to conductors of the bottom layer. Such power grid structures are usually modeled as RCL circuits. Each conductor segment is modeled as a resistor in series with an inductor and each grid node is connected to the ground through a capacitor. External power supplies are modeled as ideal constant voltage sources and current drains are modeled as ideal time-varying current sources. The power grid model may or may not be regular.

Assume there exist a total of N grid nodes that are not terminals of ideal voltage sources. If only the voltage drops at these N nodes are of interest, a *revised* circuit model can be generated by short-circuiting all ideal voltage sources and reversing the directions of all ideal current sources [7]. Nodal voltages of the revised circuit are voltage drops of the original circuit. The MNA equation of the revised circuit can be written as

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

Here, $x(t) \in \mathbb{R}^n$ is the state vector of nodal voltages and inductor currents; n is the total number of nodes, voltage sources and inductors; $u(t) \in \mathbb{R}^m$ is the vector of current sources; $C \in M_{n,n}(\mathbb{R})$ is a diagonal matrix with its diagonal elements being capacitances and inductances; $G \in M_{n,n}(\mathbb{R})$ is a matrix of conductances and "± 1"; $H \in M_{n,m}(\mathbb{R})$ is the 0-1 current distribution matrix. Note that each row of H may contain more than one "1", which means that more than one current source can be attached to a single node. On the other hand, each column of H contains exactly one "1", which corresponds to the position at which a current source is attached.

2.2 Previous Vectorless Power Grid Verification Methods

Vectorless power grid verification is first proposed in [7], where only DC analysis is considered. The worst-case voltage drop is solved from LP problems under current constraints. In [3, 6], the algorithm is extended to transient analysis. [3] uses geometry-based methods to solve the LP problems. This algorithm achieves lower computational complexity with some sacrifice of accuracy. The current constraints in [3] are time-independent, and current patterns obtained there are constant throughout the simulation time span. [6] takes inductors into consideration and is applicable to general RCL power grid models. In [2] an impulse response-based algorithm is proposed which considers the transition time of current sources. With the transition time constraints, the current patterns generated are more realistic, the method is inefficient for large instances. In [5] an approximate matrix inversion method is proposed to more quickly formulate the LP problems faster. The small entries in the approximate matrix are set to zero, so this method introduces added inaccuracy in predicting the voltage drops. In [15], a dual formulation is proposed in which the dual problems are convex with fewer variables. Solving these "reduced" convex optimization problems is expected to be more efficient than solving the original LP problems. The problem formulations of [3, 5, 15] are based on M-matrix assumptions (i.e., the system matrix of the model is an M-matrix).

3. HIERARCHICAL CONSTRAINTS AND LINEAR PROGRAMS

Given a power grid model and current and power constraints, our objective is to predict the worst-case voltage drops on the power grid by solving LP problems. In this section, we focus on how to formulate the LP problems with hierarchical current and power constraints. The next section will focus on how to efficiently solve the LP problems.

3.1 Transient Analysis

In this subsection, backward Euler-based transient analysis is performed to derive the relationship between voltage drops and currents. Similar derivations appear in [5]. Using backward Euler method, (1) can be discretized as

$$\left(G + \frac{C}{\Delta t}\right) x(t + \Delta t) = \frac{C}{\Delta t} x(t) + Hu(t + \Delta t). \tag{2}$$

Under the stability assumption (all the poles of (1) are distributed on the left half of complex plane), the system matrix $G + \frac{C}{\Delta t}$ is invertible. Define

$$\mathcal{M} = \left(G + \frac{C}{\Delta t}\right)^{-1} \frac{C}{\Delta t}, \tag{3a}$$

$$\mathcal{N} = \left(G + \frac{C}{\Delta t}\right) H. \tag{3b}$$

We have

$$x(t + \Delta t) = \mathcal{M}x(t) + \mathcal{N}u(t + \Delta t). \tag{4}$$

By dividing the simulation time span into k_t time steps, and assuming a zero initial state (i.e., $x(0) = 0$), the voltage drops at the last time step can be represented as

$$x(k_t \Delta t) = \sum_{k=1}^{k_t} \mathcal{M}^{k_t - k} \mathcal{N} u(k\Delta t). \tag{5}$$

3.2 Hierarchical Constraints

In practice the peak value of a current source is usually bounded, i.e., $u_i(t) \le I_{L,i}$. Local current constraints can be formulated by combining all these inequalities as

$$0 \le u(t) \le I_L \quad \text{or} \quad 0 \le u(k\Delta t) \le I_L, \tag{6}$$

where $I_L \in \mathbb{R}^m$ is a vector with its i^{th} element being the upper bound of the i^{th} current source. Here, and in what follows, "$\le$" is taken to be element-wise. On the other hand, global current constraints are formulated as

$$Uu(t) \le I_G \quad \text{or} \quad Uu(k\Delta t) \le I_G, \tag{7}$$

where $U \in M_{p,m}(\mathbb{R})$ is a 0-1 matrix. Each inequality in (7) corresponds to a certain functional block, i.e., the total current of a

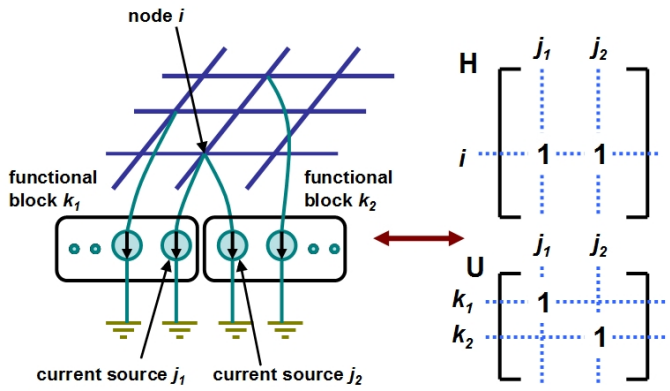


Figure 1: Block-level current constraints based on total currents of functional blocks.

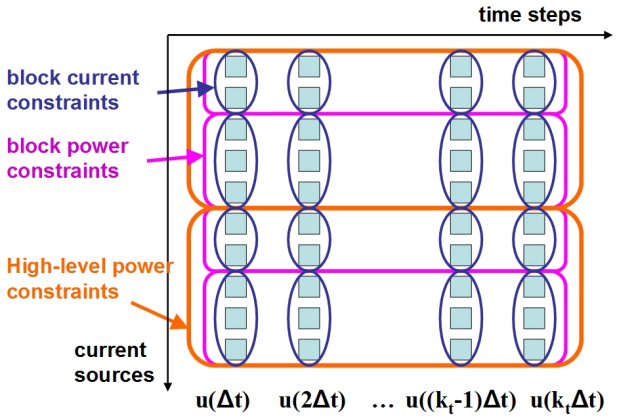


Figure 2: Illustration of hierarchical current and power constraints. Each column represents a current source vector at a single time step. Unlike current constraints, power constraints define an upper bound of the sum of currents at different time steps.

functional block is bounded. The global current constraints here are different from [5,7] in the sense that each column of U contains at most one nonzero entry. This implies that one current source belongs to only one specific functional block (i.e., appears in only one inequality). Hence if two different functional blocks draw currents from one grid node, the current drawn from this node should be modeled as two independent current sources. This agrees with the fact that more than one "1" may exist in one row of H, as shown in Fig. 1.

Besides current constraints, novel power constraints are introduced which restrict the average power consumption of a functional block:

$$U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_B. \tag{8}$$

Here V_{dd} is the voltage value of the external power supply. $P_B \in \mathbb{R}^p$ is a vector with its i^{th} element being the power limit of the i^{th} functional block. The power constraint is reasonable as power consumption of a functional block is usually bounded in the design requirements.

If by design some functional blocks have interactions with each other, high-level power constraints can be introduced which restrict the total power consumption of certain groups of functional blocks:

$$\begin{aligned} &[1^{st} \text{ level}]: U_1 U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_{T1}; \\ &\cdots \\ &[r^{th} \text{ level}]: U_r U_{r-1} \cdots U_1 U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_{Tr}. \end{aligned} \tag{9}$$

Here $U_1 \in M_{p,p}(\mathbb{R})$, $U_2 \in M_{p_2,p_1}(\mathbb{R})$, ...,$U_r \in M_{p_r,p_{r-1}}(\mathbb{R})$ are 0-1 matrices with each column containing at most one "1".

With the property that U, U_1, ..., U_r containing at most one nonzero entry in every column, (6)-(9) constitute a group of *hierarchical* constraints, as depicted in Fig. 2. Based on this special hierarchical structure, solving LP problems can be significantly simplified by applying a sorting-deletion algorithm, which will be detailed in Section 4.3.

3.3 LP Formulation

Linear programs to solve the worst-case voltage drop can be formulated as

$$\begin{aligned} \max_{\substack{i=1,...,N \\ k'=1,...,k_t}} \quad & x_i(k'\Delta t) = \sum_{k=1}^{k'} c_{i,k',k} u(k\Delta t) \\ \text{s.t.} \quad & \begin{cases} 0 \le u(k\Delta t) \le I_L, \quad Uu(k\Delta t) \le I_G, \\ U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_B, \\ U_{r'}U_{r'-1}\cdots U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_{Tr'} \ (r'=1...r). \end{cases} \end{aligned} \tag{10}$$

Here $c_{i,k',k}$ is the i^{th} row of $\mathcal{M}^{k'-k}\mathcal{N}$.

Equation (10) in fact contains $N \times k_t$ LP problems. We denote the solution of one single LP problem as $\Phi(i,k')$. The worst-case voltage drop is recognized as $\max\{\Phi(i,k')\}$ ($i = 1,\ldots,N$, $k' = 1,\ldots,k_t$). However, solving all these LP problems is prohibitively expensive. Therefore, the problem size is first reduced in both time and space domains. Then, an efficient parallel algorithm is proposed to calculate $c_{i,k',k}$'s. Moreover, a sorting-deletion algorithm is proposed which is more efficient than standard LP algorithms. The details will be elaborated in the next section.

4. PROBLEM REDUCTION AND SORTING-DELETION ALGORITHM

4.1 Problem Reduction in Both Time and Space Domains

LEMMA 4.1. *For any integers i, k_1, k_2 with $1 \le i \le N$ and $1 \le k_1 < k_2 \le k_t$, we have $\Phi(i,k_1) \le \Phi(i,k_2)$.*

PROOF. Assume that the maximum value $\Phi(i,k_1)$ is reached when currents are $u^{(1)}(k\Delta t)$ for $k = 1,\ldots,k_t$, where $u^{(1)}(k\Delta t)$ satisfy the constraints in (10). Then, for the LP problem (i,k_2), let $u^{(2)}(k\Delta t) = 0$ for $k = 1,\ldots,k_2 - k_1$ and $u^{(2)}(k\Delta t) = u^{(1)}((k-k_2+k_1)\Delta t)$ for $k = k_2 - k_1 + 1,\ldots,k_t$. Notice that since $\max\{u_i^{(2)}(k\Delta t)|k = 1,\ldots,k_t\} \le \max\{u_i^{(1)}(k\Delta t)|k =$

$1,\ldots,k_t\} \le I_{L,i}$, we have $u^{(2)}(k\Delta t)$ satisfies the current constraints. Besides, since $\sum_{k=1}^{k_t} u^{(2)}(k\Delta t) = \sum_{k=1}^{k_t-k_2+k_1} u^{(1)}(k\Delta t) \le \sum_{k=1}^{k_t} u^{(1)}(k\Delta t)$, $u^{(2)}(k\Delta t)$ also satisfies the power constraints. With the observation that the value of $c_{i,k',k}$ is determined by $k'-k$ only (i.e., $c_{i,k'_1,k_1} = c_{i,k'_2,k_2}$ if $k'_1 - k_1 = k'_2 - k_2$), we have

$$\begin{aligned} x_i(k_2\Delta t) &= \sum_{k=1}^{k_2} c_{i,k_2,k} u^{(2)}(k\Delta t) \\ &= 0 + \sum_{k=k_2-k_1+1}^{k_2} c_{i,k_2,k} u^{(2)}(k\Delta t) \\ &\overset{(j\triangleq k-k_2+k_1)}{=} \sum_{j=1}^{k_1} c_{i,k_2,j+k_2-k_1} u^{(2)}((j+k_2-k_1)\Delta t) \\ &= \sum_{j=1}^{k_1} c_{i,k_1,j} u^{(1)}(j\Delta t) = \Phi(i,k_1) \end{aligned}$$

Therefore $\Phi(i,k_2) \ge \Phi(i,k_1)$. $\square$

The main idea of this proof is that the longer the time span is, the worse the voltage drop can be. If the worst-case voltage drop at t_0 is obtained under a specific current pattern, the same voltage drop can also be obtained after t_0 by translating the same current pattern. Lemma 4.1 indicates that to obtain the worst-case voltage drop, we only have to solve LP problems (10) when $k = k_t$, which significantly reduces the computation load in the time domain.

Furthermore, we need not solve (10) for all $i = 1,\ldots,N$. In practice the most significant voltage drops often appear at nodes having longest distances to voltage sources. For example, the largest voltage drops of a mesh-like power grid with voltage sources at corners are most likely to occur near the center of the power grid. Hence we can choose a set of nodes (the indices of which form a set Ω) farthest from voltage sources and solve LP problems only for nodes belonging to Ω. Or we can first perform a DC analysis-based vectorless verification as in [7] and choose Ω to be the set of nodes with the largest voltage drops. This works in practice as the solutions of the DC analysis-based algorithm, although potentially inaccurate, are able to provide a rough picture of voltage drops and identify nodes where the worst-case is most likely to occur. It is also possible to determine Ω based on previous experience, or choose Ω to be the set of nodes which are critical to circuit performance. In any event, we need only to solve (10) for $i \in \Omega$, and the cardinality of Ω can be made much smaller than N, i.e., $|\Omega| \ll N$.

As a result, the LP problems (10) can be reduced to

$$\max_{i\in\Omega} x_i(k_t\Delta t) = \sum_{k=1}^{k_t} c_{i,k} u(k\Delta t)$$
$$\text{s.t.} \begin{cases} 0 \le u(k\Delta t) \le I_L, \quad Uu(k\Delta t) \le I_G, \\ U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_B, \\ U_{r'}U_{r'-1}\cdots U\left(\sum_{k=1}^{k_t} u(k\Delta t)\right) \le \frac{k_t}{V_{dd}} P_T \ (r'=1...r), \end{cases} \tag{11}$$

where $c_{i,k}$ is the i^{th} row of $\mathcal{M}^{k_t-k}\mathcal{N}$. If all nodes must be considered (i.e. $\Omega = \{1,\ldots,N\}$), we can solve $c_{i,k}$ in parallel as proposed in the next subsection to reduce the runtime.

4.2 Efficient Calculation of Coefficients

By definition

$$c_{i,k} = e_i^T \left[\left(G + \frac{C}{\Delta t}\right)^{-1} \frac{C}{\Delta t}\right]^{k_t-k} \left(G + \frac{C}{\Delta t}\right)^{-1} H, \tag{12}$$

where $e_i \in \mathbb{R}^n$ is the i^{th} elementary unit vector. Directly solving $c_{i,k}$ by computing the matrix inverse is prohibitively expensive. Now we propose an efficient method which costs only one sparse-LU decomposition and k_t forward/backward substitutions.

Performing transposition on both sides of (12), we have

$$c_{i,k}^T = H^T \left(G^T + \frac{C^T}{\Delta t}\right)^{-1} \left[\frac{C^T}{\Delta t}\left(G^T + \frac{C^T}{\Delta t}\right)^{-1}\right]^{k_t-k} e_i. \tag{13}$$

Assuming that $G^T + \frac{C^T}{\Delta t} = L_d U_d$ (sparse-LU decomposition), (13) can be written as

$$c_{i,k}^T = H^T U_d^{-1} L_d^{-1} \underbrace{\left(\frac{C^T}{\Delta t}\right) U_d^{-1} L_d^{-1} \cdots \left(\frac{C^T}{\Delta t}\right) U_d^{-1} L_d^{-1}}_{k_t-k \text{ times}} e_i. \tag{14}$$

Note that L_d^{-1} (U_d^{-1}) multiplied to a vector is in fact a forward (backward) substitution. Hence computing (14) for all $k = 1,\ldots,k_t$ only involves one sparse-LU decomposition, k_t matrix-vector multiplications and k_t forward (backward) substitutions. The matrix-vector multiplication is extremely efficient as $\frac{C}{\Delta t}$ is a diagonal matrix. Solving all the $c_{i,k}$'s ($i \in \Omega$) requires one sparse-LU and $k_t|\Omega|$ forward (backward) substitutions and matrix-vector multiplications. Note that $c_{i,k}$ can be solved in parallel as calculations of $c_{i,k}$'s for different i are independent of each other. The calculation of (14) is equivalent to the numerical computation of transient analysis and thus is numerically robust.

4.3 Sorting-Deletion Algorithm

Consider one LP problem in (11) (a specific $i \in \Omega$), with all the $c_{i,k}$ ($k = 1,\ldots,k_t$) vectors computed following Section 4.2. It can be seen that the objective function is a linear combination of all the currents at all time steps. The coefficient of the j^{th} current source at time step k is the j^{th} entry of the vector $c_{i,k}$. To simplify notations, we reorder the subscripts of coefficients and variables as

$$\begin{array}{ccc} \tilde{c}_1 = e_1^T c_{i,1}; & \cdots & \tilde{c}_m = e_m^T c_{i,1}; \\ \vdots & \ddots & \vdots \\ \tilde{c}_{(k_t-1)m+1} = e_1^T c_{i,k_t}; & \cdots & \tilde{c}_{k_t m} = e_m^T c_{i,k_t}; \end{array} \tag{15a}$$

$$\begin{array}{ccc} \tilde{u}_1 = u_1(\Delta t); & \cdots & \tilde{u}_m = u_m(\Delta t); \\ \vdots & \ddots & \vdots \\ \tilde{u}_{(k_t-1)m+1} = u_1(k_t\Delta t); & \cdots & \tilde{u}_{k_t m} = u_m(k_t\Delta t). \end{array} \tag{15b}$$

Consequently, any constraint in (11) is in the form of $\sum_{i\in\mathcal{L}} \tilde{u}_i \le \ell$, where $\mathcal{L}$ is a subset of the indices $\{1,\ldots,k_t m\}$. Assume the total number of constraints is κ_t. In the case of (11) $\kappa_t = mk_t + pk_t + p + \sum_{r'=1}^{r} p_{r'}$. (11) can be rewritten as

$$\max\ x_{i_{node}} = \left(\sum_{i=1}^{mk_t} \tilde{c}_i \tilde{u}_i\right) \quad \text{s.t.} \quad \sum_{i\in\mathcal{L}_\kappa} \tilde{u}_i \le \ell_\kappa (\kappa = 1,\ldots,\kappa_t), \tag{16}$$

where $i_{node} \in \Omega$ is the index of a node.

LEMMA 4.2. *The maximum of (16) is hit when all the $\tilde{u}_i$'s associated with negative $\tilde{c}_i$'s are set to zero.*

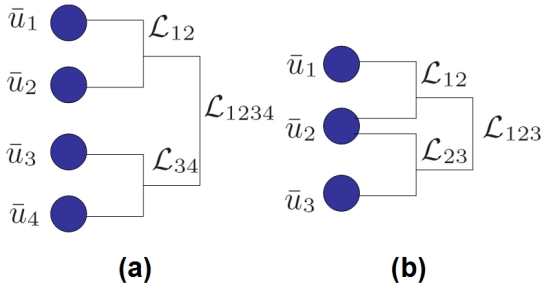


Figure 3: Different current and power constraints: (a) has a hierarchical structure as $\mathcal{L}_{12} \bigcap \mathcal{L}_{34} = \emptyset$, $\mathcal{L}_{12} \subset \mathcal{L}_{1234}$ and $\mathcal{L}_{34} \subset \mathcal{L}_{1234}$; (b) is not hierarchical as $\mathcal{L}_{12} \not\subset \mathcal{L}_{23}$ and $\mathcal{L}_{12} \not\supset \mathcal{L}_{23}$ and $\mathcal{L}_{12} \bigcap \mathcal{L}_{23} \neq \emptyset$.

PROOF. Assume the maximum is hit at a feasible point $\tilde{u}$ with at least one $\tilde{u}_j$ associated with a negative $\tilde{c}_j$ is set to a positive number. It is readily verified that the new point $\tilde{u}'$ obtained by setting $\tilde{u}_j$ to zero is still feasible (as the sum of any subset of $\tilde{u}_i'$'s is equal to or smaller than that of $\tilde{u}_i$'s). On the other hand, $x|_{\tilde{u}} < x|_{\tilde{u}'}$, which conflicts with the assumption that the maximum is hit at $\tilde{u}$. □

There exist negative $\tilde{c}_i$'s due to the existence of inductors. Sometimes the voltage drops on inductors caused by the "change" of currents are more significant than static voltage drops on resistors, so the worst-case voltage drop has a negative correlation with some current points. With Lemma 4.2, we can set all the $\tilde{u}_i$'s associated with negative $\tilde{c}_i$'s to zero and then delete these $\tilde{u}_i$ from the constraints of (16). Constraint κ is also deleted if $\mathcal{L}_\kappa$ becomes empty after deleting $\tilde{u}_i$'s. Suppose after the deletions there exist $\bar{k}$ variables and $\bar{\kappa}$ constraints. Reorder remaining $\tilde{c}_i$'s such that $\bar{c}_1 \geq \cdots \geq \bar{c}_{\bar{k}} > 0$ and reorder $\bar{u}_i$'s accordingly. (16) can be reformulated as

$$\max x_{i_{node}} = \sum_{i=1}^{\bar{k}} \bar{c}_i \bar{u}_i \quad \text{s.t.} \quad \sum_{i \in \mathcal{L}_\kappa} \bar{u}_i \leq \ell_\kappa \ \ (\kappa = 1, \ldots, \bar{\kappa}) \tag{17}$$

Now we digress to take a look at the constraints of (17), which have a hierarchical structure as depicted in Fig. 2. The hierarchical constraints structure implies that for any two sets $\mathcal{L}_{\kappa_1}$ and $\mathcal{L}_{\kappa_2}$, one of the following equations must hold: (i) $\mathcal{L}_{\kappa_1} \bigcap \mathcal{L}_{\kappa_2} = \emptyset$; (ii) $\mathcal{L}_{\kappa_1} \subset \mathcal{L}_{\kappa_2}$; (iii) $\mathcal{L}_{\kappa_1} \supset \mathcal{L}_{\kappa_2}$. See Fig. 3.

DEFINITION 4.3 (BOUNDARY POINT). *Suppose $\bar{u} = [\bar{u}_1, \ldots, \bar{u}_{\bar{k}}]^T$ is a feasible point under the constraints of (17). $\bar{u}$ is called a boundary point if for any $1 \leq i \leq \bar{k}$ and any $\epsilon > 0$, $[\bar{u}_1, \ldots, \bar{u}_i + \epsilon, \ldots, \bar{u}_{\bar{k}}]^T$ is not feasible.*

We look at an illustrative example as shown in Fig.3(a). The constraints in Fig.3(a) can be written explicitly as:

$$\begin{gathered} 0 < \bar{u}_1 < l_1, \ \ 0 < \bar{u}_2 < l_2, \ \ 0 < \bar{u}_3 < l_3, \ \ 0 < \bar{u}_4 < l_4; \\ \bar{u}_1 + \bar{u}_2 < l_{12}, \ \ \bar{u}_3 + \bar{u}_4 < l_{34}; \\ \bar{u}_1 + \bar{u}_2 + \bar{u}_3 + \bar{u}_4 < l_{1234} \end{gathered} \tag{18}$$

The upper bounds of $\{\bar{u}_1, \bar{u}_2, \bar{u}_3, \bar{u}_4\}$ and its subsets are:

$$\begin{gathered} \mathcal{B}(\bar{u}_1) = \min\{l_1, l_{12}, l_{1234}\}, \ \ \mathcal{B}(\bar{u}_2) = \min\{l_2, l_{12}, l_{1234}\}, \\ \mathcal{B}(\bar{u}_3) = \min\{l_3, l_{34}, l_{1234}\}, \ \ \mathcal{B}(\bar{u}_4) = \min\{l_4, l_{34}, l_{1234}\}; \\ \mathcal{B}(\bar{u}_1 + \bar{u}_2) = \min\{\mathcal{B}(\bar{u}_1) + \mathcal{B}(\bar{u}_2), l_{12}, l_{1234}\}; \\ \mathcal{B}(\bar{u}_3 + \bar{u}_4) = \min\{\mathcal{B}(\bar{u}_3) + \mathcal{B}(\bar{u}_4), l_{34}, l_{1234}\}; \\ \mathcal{B}(\bar{u}_1 + \bar{u}_2 + \bar{u}_3 + \bar{u}_4) = \min\{\mathcal{B}(\bar{u}_1 + \bar{u}_2) + \mathcal{B}(\bar{u}_3 + \bar{u}_4), l_{1234}\}; \end{gathered} \tag{19}$$

LEMMA 4.4. *The sum of elements (coordinates) of any boundary point is the same, i.e., $\sum_{i=1}^{\bar{k}} \bar{u}_i$ is a constant as long as $[\bar{u}_1, \ldots, \bar{u}_{\bar{k}}]^T$ is a boundary point.*

PROOF. Consider a general (arbitrary) hierarchical constraint structure,
which can be represented by a "tree". Let the depth of the "tree" be d and each node of the tree have at most w children. Denote the j^{th} node in the i level as $\mathcal{L}_{i,j}$. Assuming there exists a boundary point $(\bar{u}_1, \ldots, \bar{u}_{\bar{k}})$ with $\sum_{i=1}^{\bar{k}} \bar{u}_i < \mathcal{B}(\sum_{i=1}^{\bar{k}} \bar{u}_i)$, let $\epsilon = \mathcal{B}(\sum_{i=1}^{\bar{k}} \bar{u}_i) - \sum_{i=1}^{\bar{k}} \bar{u}_i > 0$. So there exists at least one child (w.l.o.g, we assume this is the first child) with $\sum_{i \in \mathcal{L}_{2,1}} \bar{u}_i \leq \mathcal{B}(\sum_{i \in \mathcal{L}_{2,1}} \bar{u}_i) - \epsilon/w$. Otherwise, $\sum_{i=1}^{\bar{k}} \bar{u}_i > \mathcal{B}(\sum_{i=1}^{\bar{k}} \bar{u}_i) - \epsilon$. Perform this deduction to the bottom of the tree, we conclude that there exists at least one child (w.l.o.g.. we assume it is the first child of its parent) with $\bar{u}_1 \leq \mathcal{B}(\bar{u}_1) - w^{-d}\epsilon$. Thus we conclude that $(\bar{u}_1 + w^{-d}\epsilon, \ldots, \bar{u}_{\bar{k}})$ is feasible. This contradicts the fact that $(\bar{u}_1, \ldots, \bar{u}_{\bar{k}})$ is a boundary point. □

The main idea behind this lemma is that all the boundary points belong to the plane $\bar{u}_1 + \cdots + \bar{u}_{\bar{k}} = \mathcal{B}(\bar{u}_1 + \cdots + \bar{u}_{\bar{k}})$. Now we return to the LP problem (17), and show that the solution computed by the sorting-deletion algorithm (Algorithm 1) is optimal. We begin with the following two lemmas.

Algorithm 1 : Sorting-deletion algorithm

for $i = 1,..,\bar{k}$ **do**
- (1) Select all the sets $\mathcal{L}_\kappa$ that satisfy $i \in \mathcal{L}_\kappa$. The subscripts of these $\mathcal{L}_\kappa$ form a set $\mathcal{K}_i$;
- (2) Set $\bar{u}_i$ to be $\min\{\ell_\kappa | \kappa \in \mathcal{K}_i\}$;
- (3) $\ell_\kappa = \ell_\kappa - \bar{u}_i$ for all $\kappa \in \mathcal{K}_i$;

LEMMA 4.5. *The solution computed by the sorting-deletion algorithm is a boundary point.*

PROOF. Assume that the solution $[\bar{u}_1, \ldots, \bar{u}_{\bar{k}}]^T$ computed by the sorting-deletion algorithm is not a boundary point. Then there exist an integer $i \in [1, \tilde{k}]$ and $\epsilon > 0$ such that $[\bar{u}_1, \ldots, \bar{u}_i + \epsilon, \ldots, \bar{u}_{\tilde{k}}]^T$ is feasible. However, from Algorithm 1 we know that $\bar{u}_i = \min\{\ell_\kappa | \kappa \in \mathcal{K}_i\}$. If the i^{th} variable is set to be $\bar{u}_i + \epsilon$, at least one constraint is violated. Therefore, $[\bar{u}_1, \ldots, \bar{u}_i + \epsilon, \ldots, \bar{u}_{\tilde{k}}]^T$ is not feasible which leads to a contradiction. □

LEMMA 4.6. *Any optimal solution of the LP problem (17) is a boundary point.*

PROOF. Let $\bar{u} = [\bar{u}_1, \ldots, \bar{u}_{\tilde{k}}]^T$ be any optimal solution to LP problem (17). Assume $\bar{u}$ is not a boundary point for the sake of

contradiction. Then there exists an integer $i \in [1, \tilde{k}]$ and $\epsilon > 0$ such that $[\bar{u}_1, \ldots, \bar{u}_i + \epsilon, \ldots, \bar{u}_{\tilde{k}}]^T$ is feasible. Since $\bar{c}_i > 0$ for all $1 \leq i \leq \tilde{k}$, we have $\bar{c}_1\bar{u}_1 + \ldots + \bar{c}_i(\bar{u}_i + \epsilon) + \ldots + \bar{c}_{\tilde{k}} > \bar{c}_1\bar{u}_1 + \ldots + \bar{c}_i\bar{u}_i + \ldots + \bar{c}_{\tilde{k}}$. Hence, $[\bar{u}_1, \ldots, \bar{u}_i + \epsilon, \ldots, \bar{u}_{\tilde{k}}]^T$ is a better solution for (17) than $\bar{u}$, a contradiction. □

THEOREM 4.7. *The solution computed by the sorting-deletion algorithm is an optimal solution of the LP problem (17).*

PROOF. Suppose the solution computed by the sorting-deletion algorithm is $\bar{u}^{(1)} = [\bar{u}_1^{(1)}, \ldots, \bar{u}_{\tilde{k}}^{(1)}]^T$ and an optimal solution of the LP problem is $\bar{u}^{(2)} = [\bar{u}_1^{(2)}, \ldots, \bar{u}_{\tilde{k}}^{(2)}]^T$. Assume the two points are different. From Lemma 4.4, Lemma 4.5 and Lemma 4.6 we know that $\sum_{i=1}^{\bar{k}} \bar{u}_i^{(1)} = \sum_{i=1}^{\bar{k}} \bar{u}_i^{(2)}$.

Suppose j is the lowest index at which $\bar{u}^{(1)}$ and $\bar{u}^{(2)}$ are different. We have $\bar{u}_j^{(2)} < \bar{u}_j^{(1)}$. Otherwise, $\bar{u}_j^{(2)} > \bar{u}_j^{(1)}$, and at least one constraint $\mathcal{L}_\kappa$ ($\kappa \in \mathcal{K}_j$) is not satisfied. As $\sum_{i=1}^{\bar{k}} \bar{u}_i^{(1)} = \sum_{i=1}^{\bar{k}} \bar{u}_i^{(2)}$, there exists at least one $j' > j$ such that $\bar{u}_{j'}^{(2)} > \bar{u}_{j'}^{(1)} \geq 0$. As for $\forall \kappa_1, \kappa_2 \in \mathcal{K}_j$, $\mathcal{L}_{\kappa_1} \bigcap \mathcal{L}_{\kappa_2} \neq \emptyset$ (both contain j), we have $\mathcal{L}_{\kappa_1} \subset \mathcal{L}_{\kappa_2}$ or $\mathcal{L}_{\kappa_1} \supset \mathcal{L}_{\kappa_2}$. Assume w.l.o.g. that $\mathcal{L}_1 \subset \ldots \subset \mathcal{L}_\mu$ (μ is the cardinality of $\mathcal{K}_j$). Suppose $\mathcal{L}_{\kappa_1}$ is the first (i.e. smallest) set among $\mathcal{L}_1, \ldots, \mathcal{L}_\mu$ that contains $\bar{u}_{j'}^{(2)} > 0$ ($j' > j$), we adapt the optimal point $\bar{u}^{(2)}$ by setting $\bar{u}_j^{(2)}$ to $\bar{u}_j^{(2)} + \delta$ and $\bar{u}_{j'}^{(2)}$ to $\bar{u}_{j'}^{(2)} - \delta$ with $\delta \triangleq \min\{\bar{u}_j^{(1)} - \bar{u}_j^{(2)}, \bar{u}_{j'}^{(2)}\}$. If no set among $\mathcal{L}_1, \ldots, \mathcal{L}_\mu$ contains such $\bar{u}_{j'} > 0$ ($j' > j$), choose the first $\bar{u}_{j'}^{(2)} > 0$ ($j' > j$) and perform the similar adaptation. It is readily verified that the adapted point still satisfies all the constraints and thus is feasible. On the other hand, because $\bar{c}_j \geq \bar{c}_{j'}$, we have $\bar{c}_1\bar{u}_1^{(2)} + \cdots + \bar{c}_j(\bar{u}_j^{(2)} + \delta) + \cdots + \bar{c}_{j'}(\bar{u}_{j'} - \delta) + \cdots + \bar{c}_{\tilde{k}}\bar{u}_{\tilde{k}} \geq \bar{c}_1\bar{u}_1^{(2)} + \cdots + \bar{c}_j\bar{u}_j^{(2)} + \cdots + \bar{c}_{j'}\bar{u}_{j'}^{(2)} + \cdots + \bar{c}_{\tilde{k}}\bar{u}_{\tilde{k}}$. Therefore the adapted point is also an optimal point. After repeatedly performing such adaptation, $\bar{u}_j^{(1)} = \bar{u}_j^{(2)}$. Then the first difference appears at a position after j. Perform all the steps so on and so forth we have $\bar{u}^{(1)} = \bar{u}^{(2)}$. As $\bar{u}^{(2)}$ is an optimal point, $\bar{u}^{(1)}$ is also an optimal solution. □

The intuition behind this theorem is that the optimal solution is obtained by giving the variable associated with the largest coefficient the largest possible value.

4.4 Algorithm Flow and Complexity

The algorithm flow for the worst-case voltage drop prediction is summarized as Algorithm 2. Its computational complexity is analyzed as follows.

1. Computing $c_{i_{node},k}$'s. Computing $c_{i_{node},k}$'s for each $i_{node} \in \Omega$ requires one sparse-LU decomposition and k_t forward or backward substitutions. As the system matrix is an n by n sparse matrix with $O(n)$ nonzero entries mainly distributing near the diagonal line, this procedure has a complexity of $O(n^\alpha k_t)$, with $1 < \alpha < 2$.
2. Sorting $\tilde{c}_i$'s. For each LP problem there exist mk_t variables and coefficients. Employing the most efficient sorting algorithm, this procedure has a complexity of $O(mk_t \log(mk_t))$.
3. Sorting-deletion algorithm. The sorting-deletion algorithm determines $\bar{u}_i$ one at a time and then subtract the value from the constraints. In practice each $\bar{u}_i$ involves only several constraints, i.e., $|\mathcal{K}_i| < 10$. This procedure has a complexity of $O(\bar{k})$ with $\bar{k} \leq mk_t$.

In summary, the complexity of Algorithm 2 is dominated by the computation of $c_{i_{node},k}$'s. If executed in sequence, the overall complexity is $O(n^\alpha k_t|\Omega|)$. If executed in parallel, the overall complexity is $O(n^\alpha k_t)$.

Algorithm 2 : Worst-case voltage drop prediction

```
Set up hierarchical current and power constraints based on previous experience and/or design requirements;
for ∀ i_node ∈ Ω, execute 3 : 12 (in parallel)
    Calculate c_{i_node,k}'s following Section 4.2;
    Set up the (reduced) LP problem as (11);
    Set all ũ_i's associated with negative c̃_i's to zero and delete them from constraint sets L_i's;
    Sort the coefficients c̃_i's in the descending order and reformulate the LP problem as (17);
    for i = 1 : k̄ do
        Select all the sets L_κ that satisfy i ∈ L_κ. The subscripts of these L_κ form a set K_i;
        Set ū_i to be min{ℓ_κ | κ ∈ K_i};
        ℓ_κ = ℓ_κ − ū_i for all κ ∈ K_i;
    end
    Compute x_{i_node} = Σ_{i=1}^{k̄} c̄_i ū_i;
end
Worst-case voltage drop is max{x_{i_node} | i_node ∈ Ω}.
```

5. EXPERIMENTAL RESULTS

We generate two 3D power grids as benchmarks. Each of the power grids has four metal layers and is modeled as an equivalent RCL circuit. Basic parameters of the power grids and corresponding RCL circuits are recorded in Table 1. The simulation time is $0 - 1ns$ and is divided into 100 intervals ($k_t = 100$) with each interval being $10ps$. LP problems are set up based on local current constraints and global (including block-level current, block-level power and high-level power) constraints. Sizes of the resulting LP problems are also recorded in Table 1.

The LP problems are solved for every node $i_{node} \in \Omega$ ($|\Omega| = 100$), first by standard LP methods and then by the proposed sorting-deletion algorithm. The program is executed on a Linux workstation with 3.0GHz 8-core Intel Xeon CPU and 16G memory. CPU times are reported in Table 2. The voltage drops are omitted in this table as they are the same for both methods. From the table we conclude that solving the LP problems is speeded up by approximately $> 7\times$ using the proposed algorithm when there exist no power constraints. If there exist power constraints, the standard LP method does not work because the iteration number exceeds an acceptable number. An intuitive explanation for this phenomenon is that in the LP problem without power constraints, the coefficient matrix of the inequality contains one entry in each column (like a "diagonal" matrix). Thus using standard methods to solve LP problems without power constraints is much faster than to solve LP problems with power constraints. The speedups for both single node case and multiple node case are roughly the same because each node is solved independently.

To show that omitting power constraints may result in an overly pessimistic voltage drop prediction, we solve the LP problems (using sorting-deletion algorithm) for nodes belonging to Ω both with and without the power constraints (pc's). The results are shown in Table 3. The worst-case current patterns of some specific current sources are plotted in Fig. 4(a) & 4(b). It can be seen from Table 3 that omitting power constraints may result in an 30% overestimation of the worst-case voltage drop. Fig. 4(a) & 4(b) show that current sources keep their peak current values in a much longer time period if power constraints are omitted, which are not realistic.

Table 1: Parameters of the power grids used in the experiment

	Power grid models						LP problems	
	Nodes (N)	Sources (m)	Matrix size (n)	No. of R's	No. of C's	No. of L's	Variables	$\|\Omega\|$
Power grid 1	75,762	37,881	113,499	54,350	37,684	37,684	3.7M	100
Power grid 2	980,313	490,157	146,9755	608,792	394,444	394,444	690M	100

The worst-case current patterns with power constraints are more realistic. To provide intuitive pictures of how voltage drop at the node where worst-case voltage drop occurs changes, we perform transient simulation using the worst-case current waveforms. It can be seen from Fig. 5(a) & 5(b) that omitting power constraints does result in overly pessimistic voltage drop predictions.

To show the impact of the constraint structure on voltage drops and CPU times, we do experiments on power grid 1 (one single node) using different constraint structures. The results are shown in Table 4. Both standard method and sorting-deletion algorithm are used. Sorting-deletion algorithm does not apply for non-hierarchical constraint structure ("×" in Table 4). Standard methods do not work for non-hierarchical constraints and constraints with pc's because the iteration number exceeds an acceptable value (CPU time too large). Table 4 indicates that more power constraint levels result in lower voltage drop prediction. In practice the number of constraint levels should be determined based on design requirements or experience. Table 4 also indicates that CPU time of sorting-deletion algorithm does not increase significantly with hierarchical levels.

Table 3: Voltage drop predictions with and without power constraints

		Without pc's	With pc's	Over estimation	Percentage
Power grid 1	Average for Ω	62.3 mV	46.9 mV	15.4 mV	33%
	Worst-case	63.4 mV	48.1 mV	15.3 mV	32%
Power grid 2	Average for Ω	80.2 mV	61.1 mV	19.1 mV	31%
	Worst-case	81.3 mV	63.2 mV	18.1 mV	29%

Table 4: Voltage drop and CPU times for different constraint structures

		non-hier	w/o pc's	L-1 pc's	L-2 pc's	L-3 pc's
Standard method	Voltage drop (mV)	–	61.5	–	–	–
	CPU time (s)	–	7.14	–	–	–
Sorting deletion	Voltage drop (mV)	×	61.5	45.7	37.4	33.2
	CPU time (s)	×	0.74	0.83	0.90	0.96

[2]Here "non-hier" represents non-hierarchical constraints. L-1 (L-2, L-3) pc's represent hierarchical constraints with $r = 1$ ($r = 2$, $r = 3$) level(s) of power constraints.

6. CONCLUSIONS

A more realistic early-stage power grid verification algorithm based on hierarchical current and power constraints has been proposed in this paper. The proposed algorithm does not rely on the M-matrix assumption and thus naturally handles general RCL power grid models. Besides, currents at different time steps are treated as independent variables in LP problems and additional power constraints are introduced which restrict the energy consumed by certain current sources. As a result, worst-case current patterns solved by the proposed algorithm are more realistic in the sense that they are time-varying and cannot keep peak values all the time. Consequently, the worst-case voltage drop prediction is less pessimistic. Moreover, a sorting-deletion algorithm is proposed which significantly speeds up the solutions of LP problems. Experimental results have verified that the proposed algorithm generates more realistic worst-case current patterns and voltage drops. Utilizing the proposed sorting-deletion algorithm, the CPU time needed to solve LP problems is reduced by >85%.

7. REFERENCES

[1] T. Chen and C. Chen. Efficient large-scale power grid analysis based on preconditioned krylov-subspace iterative methods. In *DAC*, pages 559–562, 2001.

[2] P. Du, X. Hu, S. Weng, A. Shayan, X. Chen, E. Engin, and C. Cheng. Worst-case noise prediction with non-zero current transition times for early power distribution system verification. In *ISQED*, pages 624–631. IEEE, 2010.

[3] I. Ferzli, F. Najm, and L. Kruse. A geometric approach for early power grid verification using current constraints. In *ICCAD*, pages 40–47, 2007.

[4] J. Fu, Z. Luo, X. Hong, Y. Cai, S. Tan, and Z. Pan. A fast decoupling capacitor budgeting algorithm for robust on-chip power delivery. In *ASPDAC*, pages 505–510, 2004.

[5] A. Ghani and F. Najm. Fast vectorless power grid verification using an approximate inverse technique. In *DAC*, pages 184–189, 2009.

[6] N. Ghani and F. Najm. Handling inductance in early power grid verification. In *ICCAD*, page 134, 2006.

[7] D. Kouroussis and F. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *DAC*, pages 99–104, 2003.

[8] S. Nassif and J. Kozhaya. Fast power grid simulation. In *DAC*, pages 156–161, 2000.

[9] S. Pant, D. Blaauw, V. Zolotov, S. Sundareswaran, and R. Panda. A stochastic approach to power grid analysis. In *DAC*, pages 171–176, 2004.

[10] R. Plemmons. M-matrix characterizations. I–nonsingular M-matrices. *Linear Algebra and its Applications*, 18(2):175–188, 1977.

[11] H. Qian, S. Nassif, and S. Sapatnekar. Early-stage power grid analysis for uncertain working modes. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 24(5):676–682, 2005.

[12] H. Qian, S. Nassif, and S. Sapatnekar. Power grid analysis using random walks. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 24(8):1204–1224, 2005.

[13] Y. Wang, C. U. Lei, G. K. H. Pang, and N. Wong. MFTI: Matrix-Format Tangential Interpolation for Modeling Multi-Port Systems. In *DAC*, pages 683–686, 2010.

[14] Y. Wang, Z. Zhang, C. K. Koh, G. K. H. Pang, and N. Wong. PEDS: Passivity Enforcement for Descriptor Systems via Hamiltonian-Symplectic Matrix Pencil Perturbation. In *ICCAD*, pages 800–807, 2010.

[15] X. Xiong and J. Wang. An efficient dual algorithm for vectorless power grid verification under linear current constraints. In *DAC*, pages 837–842, 2010.

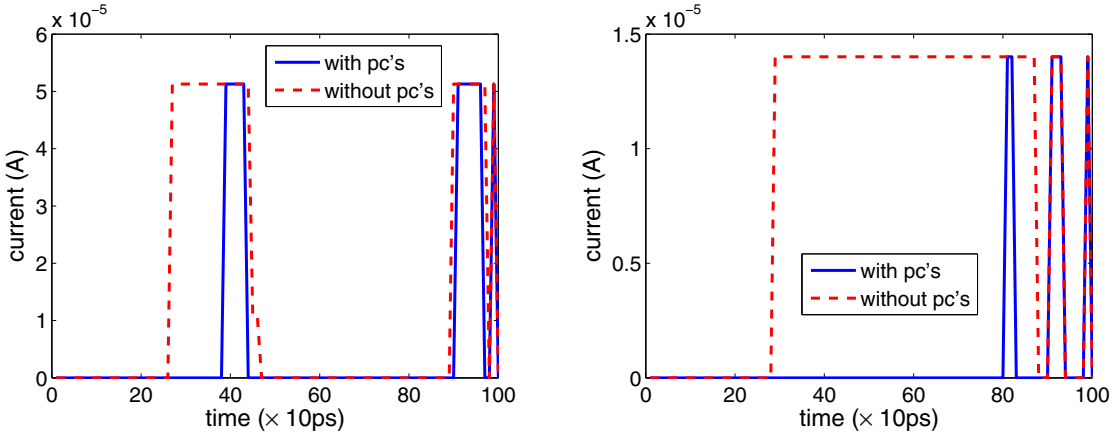
[16] W. Zhang, W. Yu, X. Hu, L. Zhang, R. Shi, H. Peng, Z. Zhu, L. Chua-Eoan, R. Murgai, T. Shibuya, et al. Efficient power network analysis considering multidomain clock gating. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 28(9):1348–1358, 2009.

[17] M. Zhao, R. Panda, S. Sapatnekar, and D. Blaauw. Hierarchical analysis of power distribution networks. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 21(2):159–168, 2002.

Table 2: Runtime comparison of standard LP algorithms and the proposed sorting-deletion algorithm

			Without pc's			With pc's		
			Standard method	Proposed algorithm	Speed-up	Standard method	Proposed algorithm	Speed-up
Power grid 1	Single node	Setup	9.86 sec	9.86 sec	–	9.86 sec	9.86 sec	–
		Solving	6.08 sec	0.71 sec	8.56×	–[1]	0.77 sec	–
	$\|\Omega\|$ nodes	Setup	901 sec	901 sec	–	901 sec	901 sec	–
		Solving	577 sec	70.2 sec	8.22×	–[1]	76.5 sec	–
Power grid 2	Single node	Setup	278 sec	278 sec	–	278 sec	278 sec	–
		Solving	74.4 sec	9.91 sec	7.51×	–[1]	10.87 sec	–
	$\|\Omega\|$ nodes	Setup	417 min	417 min	–	417 min	417 min	–
		Solving	120 min	15.4 min	7.83×	–[1]	17.1 min	–

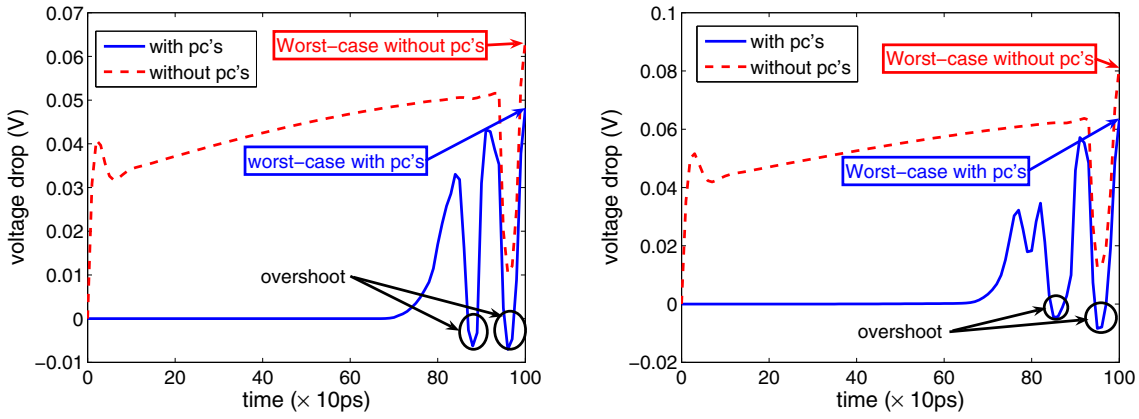
[1]Here the standard LP solver does not work because the iteration number is too large and exceeds "MaxIter".



(a) Current source 18,941 of power grid 1

(b) Current source 113,990 of power power grid 2

Figure 4: Worst-case current patterns with and without power constraints at some nodes of power grid 1 and power grid 2.



(a) Power grid 1

(b) Power grid 2

Figure 5: Voltage drop pattern at the node where worst-case voltage drop occurs for both power grid 1 and power grid 2. The values of the red and blue curves at $t = 1ns$ are the worst-case voltage drops with and without power constraints.