Power Grid Correction Using Sensitivity Analysis Under an RC Model *

Pamela Al Haddad
Department of ECE
University of Toronto
Toronto, Ontario, Canada
pamelah@eecg.utoronto.ca

Farid N. Najm
Department of ECE
University of Toronto
Toronto, Ontario, Canada
f.najm@utoronto.ca

ABSTRACT

Verifying an *RC* model of the power grid requires one to check if the steady state voltage drops on all the nodes of the grid do not exceed a certain threshold. We propose an approach to correct the grid, in case some voltage drops violate the threshold condition, by making minor changes to the original design. Previous work has been done in [1] on the DC model of the grid and this paper deals with the transient model. Rather than directly reducing the steady state voltage drops below the threshold we work on reducing the first time step voltage drops. The method uses *current constraints* proposed in [2] to find the first time step voltage drop whose distance to the corresponding threshold is the largest. It then tries to estimate it as a function of the metal widths on the grid. A non-linear optimization problem is then formulated and the required metal line width changes that reduce the first time step voltage drops by a sufficient amount are then determined. The reduction of the first time step voltage drop by that amount will make the steady state voltage drops of all the nodes less than the threshold.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*

General Terms

Algorithms, Verification

Keywords

Power Grid, voltage drop, sensitivity

1. INTRODUCTION

With technology scaling, the supply and threshold voltages are decreasing. Hence, voltage drops on the power grid become more significant and can result in longer circuit delays, leading to soft errors. Checking the integrity of the voltage on the power grid has become crucial in reliable chip design.

Power grid verification via traditional circuit simulation, requires full knowledge of the current waveforms drawn by every circuit block attached to the grid. Once these waveforms are known, the grid is simulated and the voltage drop at every node is determined. This voltage drop is then compared to a threshold to check if any node on the grid is *unsafe*. Verifying the grid using this approach requires the simulation of a comprehensive set of currents and full knowledge of the circuit. The latter is problematic in case one would like to do the verification early in the design flow, before all the circuit details are available, which is typically the case.

A new approach was proposed in [2] to deal with these problems. It is based on the *current constraints* concept to capture the uncertainty about the circuit details and circuit behavior. These current constraints can be obtained from simulations of the circuit or from the knowledge of the overall power dissipation of the circuit blocks. They are a set of upper bounds on the currents that would be drawn by the underlying circuit. A linear program (LP) is then formulated using these constraints, to check if the voltage drop at any of the nodes exceeds a certain threshold under all possible current waveforms that satisfy the constraints. In case none of the nodes violates its voltage drop requirement, the grid is said to be *robust*. Unlike the simulation based approach, the *current constraints* approach allows the verification to be done early in the design process when grid modifications can be most easily incorporated.

Once the grid verification is completed, we may have some nodes violating the voltage threshold. It then becomes useful to be able to make some minor changes on the grid so that it becomes safe without having to redo all the design from scratch. A method has been presented in [1] to correct an R model of the grid. This work presents an efficient method to correct an *RC* model of the grid. Rather than working on reducing the steady state voltage drops below the threshold in a direct way, we work with the first time step voltage drops. This is possible and in fact much easier because of the relation between the upper bound of the steady state voltage drop and the first time step voltage drop. Reducing the first time step voltage drop also reduces the steady state voltage drop, as we will see in this paper. The correction of the grid will be achieved by doing minor changes to the widths of some metal branches.

The method presented here uses linear programming concepts to express the first time step voltage drops as a function of the width parameters. It then uses a nonlinear optimization method to find the width changes that cause a sufficient reduction to the first time step voltage drop. The steady state voltage drops are consequently reduced.

This paper is organized as follows. In sections 2 and 3, the grid model and the constraint-based approach are explained. The concept of transient robustness is then defined in section 4. In section 5, the basic linear programming terminology that will be used to formulate the proposed method is introduced. The formulation of our problem is then presented in section 6, and the correction approach is given in section 7. Finally, in sections 9 and 10 the experimental results and some concluding remarks are given.

2. POWER GRID MODEL

We give a brief review of the RC model of the power grid, where each branch is represented by a resistor and where there exists a capacitor from every node to ground. In addition, some nodes have current sources (to ground) that represent the current drawn by the underlying circuit, and some grid nodes have voltage sources (to ground) that represent the external voltage supply.

*This work was supported by Advanced Micro Devices (AMD)

DAC 2011, June 5-10, 2011, San Diego, California, USA.

Let the power grid consist of $n+p$ nodes, where nodes $1,\ldots,n$ have no voltage sources attached, and nodes $(n+1),\ldots,(n+p)$ are the nodes where the p voltage sources are connected. Let c_k be the capacitance from every node k to ground. Let $i_k(t)$ be the current source connected to node k, where the direction of current is from the node to ground. We assume that $i_k(t) \geq 0$ and that $i_k(t)$ is defined for every node so that the nodes which have no current source attached have $i_k(t) = 0,\ \forall t$. Let $i(t)$ be the vector of all current sources $i_k(t)$, and $u(t)$ be the vector of all node voltages. If we apply Modified Nodal Analysis (MNA) to the grid, we have [3]:

$$Gu(t) + C\dot{u}(t) = -i(t) + GV_{dd} \tag{1}$$

where G is the $n \times n$ conductance matrix of the grid, C is the $n \times n$ diagonal capacitance matrix, and V_{dd} is a constant vector each entry of which is equal to the supply voltage value. Let $v(t) = V_{dd} - u(t)$ be the vector of voltage drops. Then, (1) can be written as:

$$Gv(t) + C\dot{v}(t) = i(t) \tag{2}$$

3. CURRENT CONSTRAINTS

As already stated, we deal with circuit current uncertainties by using the current constraints approach proposed in [2], which we now review briefly. We distinguish two types of current constraints, *local constraints* and *global constraints*.

Local constraints are upper bounds on individual current sources. For example, one may specify that the peak value of the current i_k at node k is less than a certain bound, $i_{L,k}$. We obtain this value from prior simulations of the block, from power budgets, or from engineering judgment, based on the area of the cell or block. We assume that every current source tied to the grid has an upper bound associated with it, so that if a node does not have a current source attached, the upper bound for that current is 0. We can express these constraints as:

$$0 \leq i(t) \leq i_L, \quad \forall t \geq 0 \tag{3}$$

Global constraints are upper bounds on the sums of certain subsets of current sources. For example, if the total power consumption of a certain functional block is known, then an upper bound can be specified on the sum of currents drawn by all its internal sub-blocks or cells.

Assuming we have a total number of m global constraints, then we can express them in matrix form as:

$$0 \leq Si(t) \leq i_G, \quad \forall t \geq 0 \tag{4}$$

where S is an $m \times n$ matrix that contains only 0s and 1s, which indicate if that node is included in the constraint or not, and i_G is the vector of the upper bound values. The local and global constraints can be combined into a single inequality as follows:

$$0 \leq Ui(t) \leq i_m, \quad \forall t \geq 0 \tag{5}$$

where U is an $(n+m) \times n$ matrix whose first n rows form an identity matrix corresponding to the local constraints, and whose remaining m rows form the S matrix, and where i_m is a $(n+m)\times 1$ vector which is the combination of the vectors i_L and i_G.

4. TRANSIENT ROBUSTNESS

Applying a time-discretization to (2) leads to:

$$Gv(t) + C\frac{v(t) - v(t-\Delta t)}{\Delta t} = i(t) \tag{6}$$

As in previous work on power grid verification [4], the problem is formulated by assuming the grid had zero current stimulus for all time $t \leq 0$ and then monitoring the solution of the grid under all possible feasible currents as $t \to \infty$. With zero initial currents, $v(0) = 0$, and at the first time step, $t = \Delta t$, (6) leads to:

$$Av(\Delta t) = i(\Delta t) \tag{7}$$

$$v(\Delta t) = A^{-1} i(\Delta t) \tag{8}$$

where $A = G + \frac{C}{\Delta t}$ can be shown to be a symmetric positive definite M-matrix [5], for which $A^{-1} \geq 0$ so that $v(\Delta t) \geq 0$. In general, at $t = p\Delta t$ we have [4]:

$$v(p\Delta t) = \sum_{j=0}^{p-1} \left(A^{-1}\frac{C}{\Delta t}\right)^k A^{-1} i((p-j)\Delta t) \tag{9}$$

In the rest of the paper, we use the term *steady state voltage drop* to refer to the value of the worst-case voltage drop when p tends to infinity. We are now ready to define transient robustness. A grid is said to be robust if, for every node k, the steady state voltage drop at k over all possible currents satisfying (5) is less than a given threshold v_{th}. Let $v_{ub}(p\Delta t)$ be the vector of upper bounds on the peak voltage drop at time point $p\Delta t$, defined in [6]. In particular, we will denote $v_{ub}(\Delta t)$ by V_a. From [6], we have that, as p tends to infinity, $v_{ub}(p\Delta t)$ converges to:

$$V_\infty = \left(I + G^{-1}\frac{C}{\Delta t}\right) V_a \tag{10}$$

for all $\Delta t \geq 0$. Hence, in this paper we consider that a grid is robust if the following inequality is satisfied:

$$V_\infty = \left(I + G^{-1}\frac{C}{\Delta t}\right) V_a \leq V_{th} \tag{11}$$

where V_{th} is a vector whose every entry is equal to v_{th}. Using [2], the k^{th} entry V_{a_k} of V_a can be expressed as the linear program (LP):

$$\text{maximize} \quad e_k^T v(\Delta t) \tag{12}$$

$$\text{subject to} \quad 0 \leq UAv(\Delta t) \leq i_m \tag{13}$$

where e_k is an $n \times 1$ vector of all 0's except that its k^{th} entry is 1, and the constraints are obtained using (5) and (7).

CLAIM 1. *Let $V_{a_{th}} = (I + G^{-1}\frac{C}{\Delta t})^{-1} V_{th}$. If $V_a \leq V_{a_{th}}$, then $V_\infty \leq V_{th}$ and the grid is robust. The converse is however not true.*

PROOF. From (3), we know that the source current vector is element-wise positive. Because A is an M matrix, then $A^{-1} \geq 0$, and hence from (8), $v(\Delta t) \geq 0$ for all $\Delta t \geq 0$. As we mentioned earlier, V_a=$v_{ub}(\Delta t)$ is the vector of upper bounds on the peak voltage drop at time point Δt. Hence, $V_a \geq v(\Delta t) \geq 0$ and both sides of the inequality $V_a \leq V_{a_{th}}$ are therefore positive, and multiplying them by the positive matrix $(I + G^{-1}\frac{C}{\Delta t})$ results in $V_\infty \leq V_{th}$. To prove that the converse of the statement is not true, we will consider the following counter example where: $\frac{C}{\Delta t} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $G = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}$, and $V_{th} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$. Then $A^{-1}G = \frac{1}{11}\begin{bmatrix} 7 & -1 \\ -1 & 8 \end{bmatrix}$ and $V_{a_{th}} = A^{-1}GV_{th} = \frac{1}{11}\begin{bmatrix} 0.6 \\ 0.7 \end{bmatrix}$. Now let $V_a = \begin{bmatrix} 0.001 \\ 0.07 \end{bmatrix}$, then $V_a \not\leq V_{a_{th}}$. However, $(I + G^{-1}\frac{C}{\Delta t})V_a = G^{-1}AV_a = \frac{1}{5}\begin{bmatrix} 0.078 \\ 0.491 \end{bmatrix} = \begin{bmatrix} 0.0156 \\ 0.0982 \end{bmatrix}$ and thus $(I+G^{-1}\frac{C}{\Delta t})V_a \leq V_{th}$.

□

5. LINEAR PROGRAMMING

We solve the LP problem presented in (12)-(13) using a similar approach to the one given in [1], which we now briefly review. The inequality constraints in (13) are converted to equality constraints by introducing slacks:

$$x = \begin{bmatrix} v(\Delta t) \\ s \end{bmatrix}, D = \begin{bmatrix} UA & I \\ -UA & \end{bmatrix}, b = \begin{bmatrix} i_m \\ 0 \end{bmatrix}, c = \begin{bmatrix} e_k \\ 0 \end{bmatrix} \tag{14}$$

where $s \geq 0$ is a $(2n+2m) \times 1$ vector of *slack variables*, I is the identity matrix of size $2n+2m$, b is a vector of size $2n+2m$,

and c is a vector of size $3n + 2m$. The problem can be written in standard LP form as:

$$\begin{aligned} \text{maximize} \quad v_k(\Delta t) &= c^T x & (15)\\ \text{such that} \quad Dx &= b & (16)\\ x &\geq 0 & (17)\end{aligned}$$

The set of all feasible solutions $\mathcal{X}$ [7] is defined by:

$$\mathcal{X} = \{x \mid Dx = b,\ x \geq 0\} \tag{18}$$

D has a rank of at least $2n + 2m$ because it includes an identity matrix of size $2n + 2m$. Hence, there are at least $2n + 2m$ linearly independent columns $\{d_{j_1}, d_{j_2}, \ldots, d_{j_{2n+2m}}\}$ of D, called the basis. The corresponding variables $\{x_{j_1}, x_{j_2}, \ldots, x_{j_{2n+2m}}\}$ are called the *basic variables* of the LP. Given a basis, we denote the index set of these variables by $\mathcal{B} = \{j_1, j_2, \ldots, j_{2n+2m}\}$. We also denote the index set of the remaining variables by $\mathcal{R}$. Assuming that the columns forming the basis are moved to the first $2n + 2m$ columns of D by permutation, we get:

$$D = \begin{bmatrix} B & R \end{bmatrix}, \quad x = \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{R}} \end{bmatrix}, \quad c = \begin{bmatrix} c_{\mathcal{B}} \\ c_{\mathcal{R}} \end{bmatrix} \tag{19}$$

Using (19), rewrite (15) and (16) as:

$$\begin{aligned} v_k(\Delta t) &= c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{R}}^T x_{\mathcal{R}} & (20)\\ Bx_{\mathcal{B}} + Rx_{\mathcal{R}} &= b & (21)\end{aligned}$$

Because B has full rank, then B^{-1} exists, and we have:

$$x_{\mathcal{B}} = B^{-1}(b - Rx_{\mathcal{R}}) \tag{22}$$

A basic feasible solution is a feasible solution for which

$$x_{\mathcal{B}} = B^{-1}b, \quad x_{\mathcal{R}} = 0 \tag{23}$$

If the LP has a feasible solution, then it also has a basic feasible solution that gives the same objective function value $v_k(\Delta t)$. The reader is referred to [7] for a detailed proof. So, if a problem has an optimal solution then it also has an optimal basic feasible solution and it is enough to deal with the basic feasible solutions only.
A feasible basis $\mathcal{B}$ is optimal if:

$$g = c_{\mathcal{R}}^T - c_{\mathcal{B}}^T B^{-1} R \leq 0 \tag{24}$$

The Simplex Method [8] uses this criterion to find the optimal solution of the LP. When the Simplex Method terminates, we must have, not only (24), but also:

$$y^T = c_{\mathcal{B}}^T B^{-1} R \geq 0 \tag{25}$$

because, for our problem, $c_R = 0$ [1].

6. GRID CORRECTION

If the grid is found to be *unsafe*, our optimal goal would be to do minor changes to the design so that it becomes *safe*. Let r be the vector of parameters that can be modified, typically the widths of metal branches in the grid. In our case, we consider that the grid may span several metal layers, each of which may be composed of several regions, and each region has metal lines of uniform width value r_i. Given this, then $G(r)$ is linear in r because a change in widths on any branch would have a direct proportional effect on the conductances of that branch. The diagonal capacitance matrix C also depends on r. Each entry $C_{ii}(r_i)$ is the sum of capacitance due to the grid metal branches and other fixed capacitance due to the circuit MOSFETS, and other sources, which we denote by $C_{ii}(0)$. Using the simple parallel plate capacitance model, $C_{ii}(r_i)$ is linear in r_i and we can write:

$$C_{ii}(r_i) = \frac{\epsilon_{ox} l r_i}{d} + C_{ii}(0) \tag{26}$$

where ϵ_{ox} is the permittivity of the metal-oxide, l is the length of the branch and d is the thickness of the dielectric. As well, D, x and v_k are functions of r. Our LP problem formulated in (15)-(17), can hence be rewritten as:

$$\begin{aligned} \text{maximize} \quad v_k(\Delta t, r) &= c^T x(r) & (27)\\ \text{subject to} \quad D(r)x(r) &= b & (28)\\ x(r) &\geq 0 & (29)\end{aligned}$$

Solving this LP corresponds to finding the k^{th} entry $V_{a_k}(r)$ of the vector $V_a(r)$ defined earlier. Recall from section 4 that $V_{a_{th}}$ is a function of G and C, hence $V_{a_{th}}$ is also a function of r and it can be expressed as:

$$V_{a_{th}}(r) = \left(I + G(r)^{-1}\frac{C(r)}{\Delta t}\right)^{-1} V_{th} \tag{30}$$

To determine the effect of changing r on the worst-case steady state voltage drop, the brute-force approach would be to solve the LP (27)-(29) at every value of r, for all values of k, which gives $V_a(r)$, multiply $V_a(r)$ by $\left(I + G^{-1}(r)\frac{C(r)}{\Delta t}\right)$ to get $V_\infty(r)$, and then check if $V_\infty(r) \leq V_{th}$. However this is too expensive. Instead, we propose a much more efficient approach based on the the claim presented earlier, as follows.

7. PROPOSED SOLUTION

Due to the claim of section 4, if we are able to reduce $V_a(r)$ below $V_{a_{th}(r)}$, then we are guaranteed that $V_\infty(r)$ is less than V_{th} and hence that the grid is robust. However this might lead to an overestimation of the width changes needed to fix the grid, because the condition is only sufficient but not necessary, and thus even if V_a is not less than $V_{a_{th}}$, the grid may already be safe. What we propose, therefore, is to use nonlinear optimization to reduce $V_{a_k}(r)$ as an objective function, but also to occasionally compute $V_\infty(r)$ and check if $V_\infty(r) \leq V_{th}$. The next two subsections will describe, respectively, the computation of $V_{a_k}(r)$ and the nonlinear optimization loop around it.

7.1 First Time Step

An expensive brute-force approach to evaluate the objective function $V_{a_k}(r)$, would be to solve the LP at every given value of r. Instead, we borrow an approach for doing this from [1], which turns out to require much fewer solutions of the LP. To briefly summarize this approach, suppose we have solved the LP at some nominal point r_0. It can be shown that, in a neighborhood around this value of r_0, defined by:

$$y(r) \geq 0 \quad \text{and} \quad x_{\mathcal{B}}(r) \geq 0 \tag{31}$$

the solution of the LP (27), ie. $V_{a_k}(r)$, can be obtained by solving the linear system:

$$\begin{aligned} v_k(\Delta t, r) &= c_{\mathcal{B}}^T x_{\mathcal{B}}(r) & (32)\\ B(r)x_{\mathcal{B}}(r) &= b & (33)\end{aligned}$$

This neighborhood is referred to as the *safety region* and the points along its boundary as the *breakpoints*. This approach is efficient because solving the linear system is much faster than solving the LP. In fact, we will see below that it's possible to estimate $V_{a_k}(r)$ inside the safety region using an even faster approach based on Taylor series expansion, and this works very well in practice as we will observe later on. If, in modifying r, we reach the boundary of the safety region, then a new LP is formulated and solved, a new safety region is discovered, and the algorithm continues modifying r and reducing $V_{a_k}(r)$.

In order to identify the safety region and the breakpoints, we start by finding the multi-variable Taylor series expansions of $x_{\mathcal{B}}$ and y around the initial operating point r_0. Because the conductance and capacitance matrices are linear in r, then A is linear in r and hence B and R are also linear in r and the second derivatives of A, B, and R with respect to r are zero. Furthermore, to simplify the notation, we will drop the arguments r or r_0 in connection with matrices like A, B,and R.

Table 1: Required widths changes for the case of 10 parameters

Power Grid	Percentage increase in each of the 10 parameters									
Nodes	$r1$	$r2$	$r3$	$r4$	$r5$	$r6$	$r7$	$r8$	$r9$	$r10$
1594	11.48	0.068	11.48	0.068	11.48	0.068	11.48	0.068	1.11	0.48
3427	11.12	0.060	11.12	0.060	11.12	0.060	11.12	0.060	1.13	0.59
13241	11.5	0.12	11.5	0.12	11.5	0.12	11.5	0.12	0.29	0.65
23378	11.5	0.12	11.5	0.12	11.5	0.12	11.5	0.12	0.29	0.65
70421	15.5	0.11	15.5	0.11	15.5	0.11	15.5	0.11	0.43	0.44
103927	14.99	0.10	14.99	0.10	14.99	0.10	14.99	0.10	0.42	0.41

Table 2: Maximum steady state voltage drops on the grids before and after correction and the run-times of Algorithms 1 and 2

Number of nodes	maximum voltage drop before correction(mV)	maximum voltage drop after correction(mV)	Number of Simplex solutions	Individual node correction time	Total runtime
1594	110.75	98.20	35	2.21min	3.17min
3427	105.78	94.29	28	4.9min	7.34min
13241	108.90	97.71	24	21.66min	34.2min
23378	108.89	97.70	23	41.71min	1.11hrs
70421	104.15	91.45	31	4.9hrs	6.8hrs
103927	102.19	90.32	31	9.06hrs	12.55hrs

7.1.1 First Time Step Voltage Drop Estimation

As in [1], the multi-variable Taylor series expansion for $x_{\mathcal{B}}(r)$ around r_0 can be written using multi-index notation [9] as:

$$x_{\mathcal{B}}(r) = x_{\mathcal{B}}(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^\alpha x_{\mathcal{B}}(r_0)}{\alpha!}(r-r_0)^\alpha \tag{34}$$

where,

$$\partial^\alpha x_{\mathcal{B}}(r) = \sum_{|\beta|=1} -i_\beta B^{-1}\partial^\beta B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) \tag{35}$$

and the scalar constant i_β is the value of the nonzero index β in α. For example, let us take $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}$. For $\beta = \{1, 0, \ldots, 0\}$, i_β is α_1, for $\beta = \{0, 1, \ldots, 0\}$, i_β is α_2, and so on.

Using (32), we can express the worst-case first time step voltage drop in the safety region as:

$$v_k(\Delta t, r) = v_k(\Delta t, r_0) + c_{\mathcal{B}} \sum_{|\alpha|=1}^{N} \frac{\partial^\alpha x_{\mathcal{B}}(r_0)}{\alpha!}(r-r_0)^\alpha \tag{36}$$

7.1.2 Safety Region Estimation

As in [1], the multi-variable Taylor series expansion for $y(r) = R^T\pi(r)$ (we define $\pi(r) = B^{-T}(r)c_B$ in (25)) around the initial point r_0 can be written as:

$$y(r) = y(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^\alpha y(r_0)}{\alpha!}(r-r_0)^\alpha \tag{37}$$

where $\partial^\alpha y(r_0)$ is given by:

$$\partial^\alpha y(r) = \left(\sum_{|\beta|=1} i_\beta \partial^\beta R^T \partial^{\alpha-\beta}\pi(r) \right) + R^T\partial^\alpha\pi(r) \tag{38}$$

and where the previous definition of i_β is valid, and $\partial^\alpha\pi(r)$ is given by:

$$\partial^\alpha \pi(r_0) = \sum_{|\beta|=1} -i_\beta B^{-T}\partial^\beta B^T \partial^{\alpha-\beta}\pi(r_0) \tag{39}$$

which has the same form as (35). As a result, the safety region is defined by the values of r for which all the elements of the vectors defined in (34) and (37) remain nonnegative. In practice, these expressions can be truncated up to an order N. At the breakpoints, at least one entry, j, of one of these vectors must satisfy:

$$x_{\mathcal{B}j}(r_0) + \sum_{|\alpha|=1}^{N} \frac{\partial^\alpha x_{\mathcal{B}j}(r_0)}{\alpha!}(r-r_0)^\alpha = 0 \tag{40}$$

$$y_j(r_0) + \sum_{|\alpha|=1}^{N} \frac{\partial^\alpha y_j(r_0)}{\alpha!}(r-r_0)^\alpha = 0 \tag{41}$$

These are N^{th} order polynomials in r. For example, if we take $N = 3$ and, say, there are two parameters under consideration, i.e. $r = (r_1, r_2)$ and $r_0 = (r_{10}, r_{20})$, then (40) and (41) would have the form:

$$\begin{aligned} &m_{111j}(r_1-r_{10})^3 + m_{112j}(r_1-r_{10})^2(r_2-r_{20}) \\ &+m_{122j}(r_1-r_{10})(r_2-r_{20})^2 + m_{222j}(r_2-r_{20})^3 \\ &+m_{11j}(r_1-r_{10})^2 + m_{12j}(r_1-r_{10})(r_2-r_{20}) \\ &+m_{22j}(r_2-r_{20})^2 \\ &+m_{1j}(r_1-r_{10}) + m_{2j}(r_2-r_{20}) \\ &+m_{0j} \\ &= 0 \end{aligned} \tag{42}$$

where the m's are parameters that depend on the partial derivatives of $x_{\mathcal{B}j}(r)$ and $y_j(r)$.

The breakpoint in a given direction is given by the intersection of these equations and the direction vector (which will be explained below). For example, if the direction vector is $r = (u_1 t, u_2 t)$ where $u_1^2 + u_2^2 = 1$ and t is a scalar variable, then (42) would be a third order polynomial in t:

$$n_{3j}t^3 + n_{2j}t^2 + n_{1j}t + n_{0j} = 0 \tag{43}$$

where the n's are parameters that depend on the m's in (42). We can easily solve these equations for all entries in $x_{\mathcal{B}}$ and y to find the smallest t corresponding to the breakpoint.

7.2 Nonlinear Optimization

Inside a safety region, and in order to reduce $V_{a_k}(r)$, we make use of the nonlinear optimization problem:

$$\begin{aligned} \text{minimize} \quad & v_k(\Delta t, r) &=& \ c_{\mathcal{B}}^T x_{\mathcal{B}}(r) && (44) \\ \text{such that} \quad & x_{\mathcal{B}}(r) &\geq& \ 0 && (45) \\ & y(r) &\geq& \ 0 && (46) \end{aligned}$$

where $x_{\mathcal{B}}(r)$ and $y(r)$ are given by (34) and (37). A number of nonlinear optimization algorithms are available to solve such problems [10]. We use the steepest descent line search method, with cubic interpolation, which leads to our node correction algorithm summarized in Algorithm 1. In the remainder of this section, we will describe how this algorithm works.

Algorithm 1 node_correction

1: $r_{req} = r_0$
2: **while** ($V_{a_k} > V_{a_{th_k}}$ and not_infty_check) **do**
3: Solve the LP given in (27)-(29) using the Simplex method at $r = r_{req}$ and get the optimal basis.
4: Find the Taylor expansion for $v_k(\Delta t, r)$using (36).
5: Find the safety region expressions using (34) and (37)
6: $max\_step\_taken = FALSE$.
7: **while** (max_step_taken is $FALSE$) **do**
8: $p = FIND\_SEARCH\_DIRECTION(r_{req})$,
9: $\lambda_{max} = FIND\_MAX\_STEP\_LENGTH(p, r_{req})$,
10: $\lambda = FIND\_STEP\_LENGTH(p, r_{req}, \lambda_{max})$,
11: $r_{req} = r_{req} + \lambda p$,
12: recompute $V_{a_{th}}(r_{req})$,
13: Using Taylor expansion with $r = r_{req}$, find $V_{a_k}(r_{req}) = v_k(\Delta t, r_{req})$.
14: if($\lambda = \lambda_{max}$) then $max\_step\_taken = TRUE$.
15: **end while**
16: $If(0 \leq (V_{a_k} - V_{a_{th_k}}) \leq 0.9 * (old\_V_{a_k} - old\_V_{a_{th_k}}))$
17: do check the voltage drops at infinity.
18: if ($V_\infty < V_{th}$) then $not\_infty\_check = FALSE$
19: **end while**
20: return r_{req}, not_infty_check

As a first step, at line 3, we find the optimal basis using the solution of the nominal linear program at a given r. We then compute the Taylor expressions for the first time step voltage drop and the safety region, using this basis. The search direction is then found using the steepest descent line search method. The maximum step length that can be taken is calculated by finding the breakpoint in that direction. Having the maximum step length, we then compute an appropriate step length that reduces the objective function. Step length computation is done using the cubic interpolation method [10]. Because the first time step threshold vector $V_{a_{th}}$ is not a constant and varies with r, then that threshold vector is evaluated after every step. The cost involved in the computation is due to the LU factorizations. However, in all test cases, we observed that no more than 35 LU factorizations are needed and the cost was only moderate. Using the step length and the direction, the parameter values are updated and the new $V_{a_k}(r)$ is computed. If the maximum step is taken (a breakpoint is reached), the LP is re-solved to get the optimal basis in the new region.

While doing all the above steps, we track the difference between $V_{a_k}(r)$ and $V_{a_{th_k}}(r)$. When this difference decreases by 10%, we check if $V_\infty(r) \leq V_{th}$ based on (10). Finding $V_\infty(r)$ effectively checks the voltage drop at all the grid nodes, not just at node k. If this check succeeds, then the grid is robust and the algorithm terminates, returning with the appropriate flag setting to the Top Level Algorithm (Algorithm 2) which causes that algorithm to perform no additional work and exit. Note that it is important to keep the number of V_∞ checks somewhat small, because each check is as expensive as a full grid verification which can be costly.

However, it is possible for the sufficient condition in the claim of section 4 to be met before the V_∞ check is observed. In this case, we know that node k is safe, but we don't know about the safety of other nodes. So, the algorithm exits and returns to the Top Level Algorithm with a different setting of the flag, which then checks if all nodes have indeed been corrected.

8. TOP LEVEL ALGORITHM

Our overall approach, which uses the node correction as a subroutine, is given in Algorithm 2 and works as follows. It starts with a grid verification for the given r, and finds $V_\infty(r)$. If $V_\infty(r) \leq V_{th}$, then the grid is safe. Otherwise, it finds $V_{a_{th}}(r)$ and identifies the node k such that $V_{a_k} - V_{a_{th_k}}$ is the largest. The required parameter values that result in V_{a_k} less than $V_{a_{th_k}}$ or V_∞ less than V_{th} are found using Algorithm 1. In the first case, the grid is re-verified using the new parameter values to check if any other nodes exceed the threshold and the above procedure is repeated until $V_\infty \leq V_{th}$. In the second case, the verification was already done in Algorithm 1 and we therefore exit the Top Level Algorithm. In all our test cases, Algorithm 1 was called only once and we did not need to re-verify all the nodes in the Top Level Algorithm which was hence executed only once.

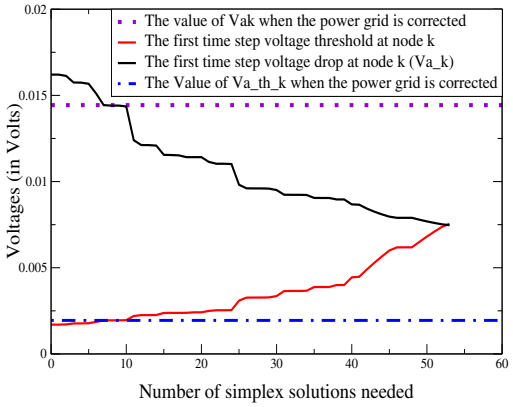

Figure 1: The variation of the worst case first time step voltage drop at node k and its threshold with respect to the number of Simplex solutions in a 598 node *RC* grid

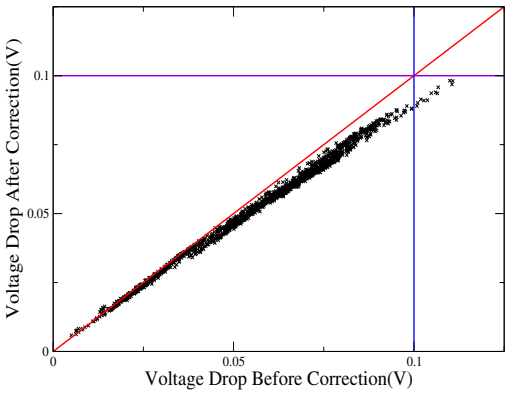

Figure 2: Correlation plot of the steady state voltage drops before and after correction for a 1594 node grid.

9. EXPERIMENTAL RESULTS

We implemented the grid correction algorithm in C++. Our algorithm was tested on a set of grids generated according to user specifications of metal layers(M1-M9), pitch and width per layer, current source distribution and grid dimensions. The computations were performed on a 64-bit Linux machine with 24 GB memory. Some of the width changes returned by the algorithm are very small, such as 1% increase or less on many parameters, and the largest widths changes are less than 16%.

In our tests, we varied the number of parameters between 10 and 20. Table 1 shows the percentage width increase for each of the 10 parameters used so that the voltage drops on all the nodes at steady state are below the threshold. Table 2 gives the time needed to correct a single node and the total CPU time of the algorithm. The latter includes full grid verification done before the correction algorithm to pick the node whose first time step voltage drop peak is farthest away from its corresponding threshold,

Algorithm 2 Top Level Algorithm

1: $grid\_unsafe = TRUE$, $not\_infty\_check = TRUE$.
2: $r_f = r_0$.
3: **while** ($grid\_unsafe = TRUE$ and $not\_infty\_check = TRUE$) **do**
4: Verify all the nodes at $r = r_f$. If ($V_\infty \leq V_{th}$) **then**, $grid\_unsafe = FALSE$.
5: **else** find $V_{a_{th}}(r_f) = (I + G^{-1}(r_f)\frac{C(r_f)}{\Delta t})^{-1}V_{th}$.
6: Pick the node k such that the difference $V_{a_k} - V_{a_{th_k}}$ is the largest.
7: Call Algorithm 1 to find the required change in parameter, r_{req} to either reduce V_{a_k} below $V_{a_{th_k}}$ or reduce V_∞ below V_{th}.
8: $r_f = r_{req}$
9: **end while**

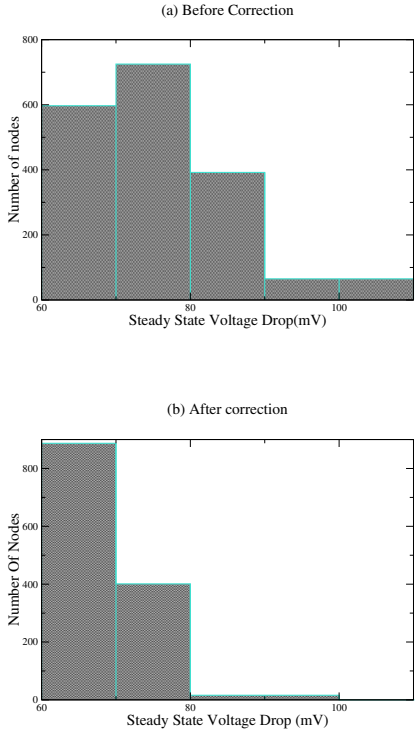

Figure 3: Steady state voltage drop histograms before and after correction for a 70421 node grid. All drops have been reduced below the threshold of 100mV.

and the time needed to perform the steady state check. Therefore, at least 2 full grid verifications are required. In all the tests that we did, we observed that no more than 2 full grid verifications were needed. The reason why the run time of the algorithm is larger than that of the correction of a single node is because of the full grid verifications, which are highly parallelizable operations. Hence, the total run time can be reduced with more efficient techniques of full grid verification. While testing our algorithm on different grids, we observed many interesting results. As the algorithm progresses, $V_{a_{th_k}}$ is increasing and V_{a_k} is decreasing. Hence there will be an eventual intersection between the two curves. However, this intersection usually happened after a large number of Simplex solutions because 1) the increase and the decrease in $V_{a_{th_k}}$ and V_{a_k} , respectively, slow down as the algorithm progresses, 2) for most of the grids we tested, the gap between $V_{a_{th_k}}$ and V_{a_k} was very large (in fact in most cases $\frac{V_{a_k}}{V_{a_{th_k}}} \leq 2.6\%$). The need to check the voltage drops at steady state after a certain number of Simplex solutions is therefore justified. This number is determined by tracking the difference between $V_{a_{th_k}}$ and V_{a_k}. The first time we do the infinity check is when the value of that difference has dropped by 10% from its initial value. Subsequent steady state checks are done when the difference drops by 10% from the last steady state check. In all our tests, the 10% decrease was sufficient and the grid was fixed the first time we did an infinity check. Hence, only two full grid verifications were needed before the grid was deemed to be *safe*. The first was needed before the correction algorithm to select the node to be corrected, and the second was needed when we did an infinity check while in the correction algorithm. The intersection point (corresponding to $V_{a_k} = V_{a_{th_k}}$) was never reached and the correction happened much earlier than that. Fig.1 illustrates how our algorithm avoids correction overestimation on a 598 node RC grid. If no infinity checks were done and we waited for the intersection between V_{a_k} and $V_{a_{th_k}}$ to happen, then we would end up with 55 Simplex solutions and therefore an overestimated correction. In fact, the grid was *safe* after 8 Simplex solutions only and therefore for smaller changes in r. Our algorithm successfully captured that by performing an infinity check.

In Fig. 3, the histograms of the voltage drops before and after correction for a 70421 node grid, are given. We only show the voltage drops larger than 60 mV for better visibility. We see from the plots that the algorithm successfully reduced all the voltage drops exceeding the threshold of 100mV. The correlation plot in Fig. 2, also shows that all the voltage drops on a 1594 grid were reduced under the threshold of 100mV.

10. CONCLUSION

With the rising demand of low voltage designs, supply voltage integrity verification has become a crucial step in reliable high-speed chip design. In this paper, we propose a fast and easy approach to correct the grid with minimal changes, when some nodes do not satisfy the threshold. We extend the work done in [1] to the case of an RC model of the grid. Instead of reducing the steady state voltage drops directly, we work on reducing the first time step voltage drops, which will have an impact at infinity. We use linear programming concepts and the current constraints-based verification approach and we formulate a non-linear optimization problem to find the required width changes needed. Our method showed good results: A grid of 70421 nodes was corrected in a total time of 6.8 hours and the changes needed in the widths of metal branches on some levels of the grid were less than 16%.

11. REFERENCES

[1] M. Aydonat and F.N. Najm. Power grid correction using sensitivity analysis. In *IEEE/ACM ICCAD*, pages 11–14, San Jose, CA, November 5-9 2010.
[2] D. Kouroussis and F. N. Najm. Static pattern-independent technique for power grid voltage integrity verification. In *IEEE/ACM Design Automation Conference*, pages 99–104, Anaheim, CA, June 2-6 2003.
[3] F. N. Najm. *Circuit Simulation*. John Wiley and Sons, Hoboken, NJ, USA, 2010.
[4] N. H. Abdul Ghani and F. N. Najm. Fast vectorless power grid verification using an approximate inverse technique. In *IEEE/ACM Design Automation Conference (DAC 2009)*, San Fransisco, CA, July 26-31 2009.
[5] M. Nizam, F. N. Najm, and A. Devgan. Power grid voltage integrity verification. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 239–244, San Diego, CA, August 8-10 2005.
[6] F. N. Najm I. A. Ferzli and L. Kruse. A geometric approach for early power grid verification using current constraints. In *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 40–47, San Jose, CA, November 5-8 2007.
[7] I. Maros. *Computational Methods of the Simplex Method*. Springer, 2003.
[8] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.
[9] X.S. Raymond. *Introduction to the Theory of Pseudodifferential operators*. CRC Press, 1991.
[10] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.