

Power Grid Correction Using Sensitivity Analysis

Meriç Aydonat
Department of ECE
University of Toronto
Toronto, Ontario, Canada
E-mail: maydonat@eecg.utoronto.ca

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

Abstract—Power grid voltage integrity verification requires one to check if all the voltage drops on the grid are less than a certain threshold. This paper addresses the problem of correcting the grid when some voltage drops exceed this threshold, by making minor modifications to the existing design. The method uses *current constraints* that capture the uncertainty about the underlying circuit behavior to find the maximum voltage drop on the grid, and then to estimate the voltage drop as a function of the metal widths on the grid. It formulates a non-linear optimization problem and finds the required change in widths that reduces the maximum voltage drop below the threshold while keeping the total area cost at a minimum.

I. INTRODUCTION

As the supply voltages have been reduced in nanometer chip technologies, modern integrated circuit (IC) designs have become more susceptible to supply voltage fluctuations. With lower supply voltages, smaller voltage drops become more significant and can reduce the timing performance of the chip, leading to soft errors. Thus, voltage integrity verification has become a crucial step in reliable high-speed chip design.

Power grid verification is traditionally done by simulation, which requires full knowledge of the current waveforms drawn by every circuit block attached to the grid. These waveforms would be used to simulate the grid and determine the voltage drop at every node. However, this approach requires 1) a comprehensive set of currents to be simulated, and 2) full knowledge of current waveforms which is a problem if one would like to verify the grid early in the design flow, before all the circuit details are available.

To overcome these problems, *current constraints* concept was proposed in [1]. These current constraints are a set of upper bounds on the currents that would be drawn by the underlying circuit. They can be obtained by simulations or from the knowledge of overall power dissipation of the circuit blocks. Using these constraints, a linear program (LP) is formulated to check if the voltage drop at any node exceeds a certain threshold under all possible current waveforms that satisfy the constraints. If all the nodes meet their voltage drop requirements, we call this grid a *robust* grid. An important advantage of the constraint-based approach over the simulation based approach is that it can be applied early in the design process when grid modifications can be most easily incorporated.

Once the power grid verification has been done, some nodes may be found to exceed the threshold. In this case, it is critical to find some means to fix this problem without the need to re-design the whole grid from scratch. This paper proposes a novel approach to correct a given non-robust grid by making minor changes, namely by changing the widths of metal branches on some level or levels of the grid. Previous work [2], [3] tackled the problem of determining

This work was supported in-part by the Semiconductor Research Corporation (SRC) and by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

the widths of metal branches to achieve a robust grid for a given set of currents drawn by the underlying circuit. This work, on the other hand, determines the required widths with incomplete information on the circuit currents.

The method presented here builds on linear programming theory to find the maximum voltage drop on the grid as a function of the metal widths. Using non-linear optimization, it then finds the required change in parameters that reduces the maximum voltage drop below the threshold. In this paper, the method is restricted to “DC grids”, i.e., for the case where all the currents are DC. We are working to extend this to the general case of time-varying currents.

The paper is organized as follows. In the next section, the grid model and the constraint-based approach are explained. In section III, the basic linear programming terminology that will be used to formulate the proposed method is introduced. The problem is defined in section IV, and the correction approach is formulated in section V. Finally, in sections VI and VII the experimental results and some concluding remarks are given.

II. POWER GRID VERIFICATION

The power grid model and the constraint-based voltage integrity verification method first introduced in [1] will lay the groundwork of our proposed grid correction approach. This is a vectorless method in the sense that it does not require complete information on the circuit currents. It can be performed early in the design process, when the circuit currents are not yet known.

A. The Power Grid Model

Consider an 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 ideal current sources (to ground) to represent the current drawn by the underlying circuit, and some grid nodes have ideal voltage sources (to ground) to represent the connections to the external voltage supply. Let the power grid consist of $n + p$ nodes, where nodes $1, \dots, n$ have no voltage sources attached, and nodes $(n + 1), \dots, (n + p)$ are the nodes where p voltage sources are attached. Let c_k be the capacitance from 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 such 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:

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

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 voltage source value. Let $v(t) = V_{dd} - u(t)$ be the vector of voltage drops. Then, (1) can be written as:

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

This is a revised system equation which represents the same circuit, but with all the current sources reversed and the voltage sources set to zero. In the rest of the paper, we will consider the DC version of this model which can be easily seen as:

$$Gv = i \quad (3)$$

B. Current Constraints

Local constraints are upper bounds on individual current sources. One may specify that the current i_k at node k does not exceed a certain bound, $i_{L,k}$. This value may be known from prior simulation or it might be the result of 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:

$$0 \leq i(t) \leq i_L, \quad \forall t \geq 0 \quad (4)$$

Global constraints are upper bounds on the sums of currents for groups 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:

$$0 \leq Si(t) \leq i_G, \quad \forall t \geq 0 \quad (5)$$

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:

$$0 \leq Ui(t) \leq i_m, \quad \forall t \geq 0 \quad (6)$$

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 i_L and i_G vectors.

C. DC Robustness

A grid is called *robust* if the maximum worst-case voltage drop of all nodes is less than a given threshold. Therefore, checking the robustness of a grid entails checking if the voltage drop at a node is lower than a threshold over all the possible currents that satisfy (6). A solution for the DC problem is presented in [1] which formulates the problem so that it can be solved as an LP.

Making use of (3), we can express the DC constraints in terms of DC voltages and DC currents in the voltage domain as:

$$0 \leq UGv \leq i_m \quad (7)$$

Let e_k be an $n \times 1$ vector consisting of all 0s, except that its k^{th} entry is 1. We can, therefore, express the DC power grid verification problem for the k^{th} node as:

$$\begin{aligned} &\text{maximize} && v_k = e_k^T v \\ &\text{such that} && 0 \leq UGv \leq i_m \end{aligned} \quad (8)$$

III. LINEAR PROGRAMMING BASICS

The inequality constraints in (8) can be converted into equality constraints by introducing *slacks*, and redefining the variables as follows:

$$x = \begin{bmatrix} v \\ s \end{bmatrix}, A = \begin{bmatrix} UG & I \\ -UG & \end{bmatrix}, b = \begin{bmatrix} i_m \\ 0 \end{bmatrix}, c = \begin{bmatrix} e_k \\ 0 \end{bmatrix} \quad (9)$$

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 $3n+2m$. Because the source current vector is element-wise positive and the conductance matrix G is an M -matrix [4], then $v = G^{-1}i \geq 0$. Using this new notation, we can write the problem in the standard LP form, as:

$$\text{maximize} \quad v_k = c^T x \quad (10)$$

$$\text{such that} \quad Ax = b \quad (11)$$

$$x \geq 0 \quad (12)$$

Using standard linear programming terminology [5], any vector x that satisfies (11) is called a *solution* of the LP. If it also satisfies (12), x is called a *feasible solution*. The set of all feasible solutions \mathcal{X} can be expressed as:

$$\mathcal{X} = \{x \mid Ax = b, x \geq 0\} \quad (13)$$

There is a column a_j in A corresponding to every variable x_j . Because A contains an identity matrix of size $2n+2m$, it has rank $2n+2m$, and we can always find $2n+2m$ linearly independent columns $\{a_{j_1}, a_{j_2}, \dots, a_{j_{2n+2m}}\}$ of A . These columns form a *basis*, and the corresponding variables $\{x_{j_1}, x_{j_2}, \dots, x_{j_{2n+2m}}\}$ are called *basic variables* of the LP. Given a basis, we will denote the index set of these variables by $\mathcal{B} = \{j_1, j_2, \dots, j_{2n+2m}\}$, and the index set of the remaining variables by \mathcal{R} . To simplify the notation, we can assume that the columns forming the basis are moved to the first $2n+2m$ columns of A , by permutation. Therefore, we can write:

$$A = [B \quad R], \quad x = \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{R}} \end{bmatrix}, \quad c = \begin{bmatrix} c_{\mathcal{B}} \\ c_{\mathcal{R}} \end{bmatrix} \quad (14)$$

where $B = A_{\mathcal{B}}$, and $R = A_{\mathcal{R}}$ are submatrices of A corresponding to \mathcal{B} and \mathcal{R} . Using (14), we can rewrite (10) and (11) as follows:

$$v_k = c_{\mathcal{B}}^T x_{\mathcal{B}} + c_{\mathcal{R}}^T x_{\mathcal{R}} \quad (15)$$

$$Bx_{\mathcal{B}} + Rx_{\mathcal{R}} = b \quad (16)$$

Because the columns of B are linearly independent, then B^{-1} exists, and from (16), we have:

$$x_{\mathcal{B}} = B^{-1}(b - Rx_{\mathcal{R}}) \quad (17)$$

From this, we can see that the values of the basic variables are uniquely determined by the values of the non-basic variables. A feasible solution for which $x_{\mathcal{R}} = 0$ is said to be a *basic feasible solution*, and it has:

$$x_{\mathcal{B}} = B^{-1}b, \quad x_{\mathcal{R}} = 0 \quad (18)$$

Theorem 1. *If the LP has a feasible solution, then it also has a basic feasible solution that gives the same objective function value v_k .*

The proof of this theorem can be found in [5]. A feasible solution is called *optimal* if it solves the LP. As a corollary, if the problem (10-12) has an optimal solution, then it has an *optimal basic solution*, and it is enough therefore to deal with basic feasible solutions only.

Given a basis \mathcal{B} , if, upon setting $x_{\mathcal{R}} = 0$, we get an $x_{\mathcal{B}} \geq 0$, so that the resulting solution x is feasible, then \mathcal{B} is said to be a *feasible*

basis. If the resulting x is in fact an optimal solution of the LP, then \mathcal{B} is said to be an *optimal basis*, and (18) gives the optimal solution of the LP.

Theorem 2. *A feasible basis \mathcal{B} is optimal if*

$$d = c_{\mathcal{R}}^T - c_{\mathcal{B}}^T B^{-1} R \leq 0 \quad (19)$$

The proof of this theorem is given in [5]. The Simplex Method [6] uses the above result to find the optimal solution of the LP. In this method, a starting basis is selected, and the columns of A are swapped in and out of the basis until $d \leq 0$. Thus, the final basis obtained by solving the LP using Simplex Method is an optimal basis, and it satisfies the condition (19). Most solvers that implement this method return the optimal basis. Therefore, once we have solved the LP using Simplex, we can make use of the available optimal basis.

Assuming that the power grid is connected, and assuming that the local and global constraints are not (trivially) all zero, then the worst-case voltage drop for any node k cannot be zero, and must be *strictly* positive, $v_k > 0$. Recall, from (9) and the definition of e_k , that c has at most one non-zero entry. Therefore, from (14), it must be the case that either $c_{\mathcal{R}} = 0$ or $c_{\mathcal{B}} = 0$. Clearly, if, for an optimal basis \mathcal{B} , we have $c_{\mathcal{B}} = 0$, then the optimal basic solution of the LP is $v_k = 0$, due to (15) and (18). This contradicts our assertion that the worst-case voltage drop on any node must be *strictly* positive. Therefore, $c_{\mathcal{B}} \neq 0$, and the optimal basis must be such that $c_{\mathcal{R}} = 0$.

As a result, when the Simplex method has “terminated”, we must have, not only (19), but also in fact:

$$y^T = c_{\mathcal{B}}^T B^{-1} R \geq 0 \quad (20)$$

IV. PROBLEM DEFINITION

If a power grid is found to be non-robust, one would like to know how to modify it, so that it becomes robust. Typically, this would involve increasing the width of metal branches on some level of the grid. Let r be the vector of parameters that can be changed. In this paper, we assume that the elements of r correspond to metal widths, so that the grid conductance matrix $G = G(r)$ is *linear* in r . All other matrices and vectors, such as A , x , v_k are also functions of r , so that the verification LP can be restated as:

$$\text{maximize } v_k(r) = c^T x(r) \quad (21)$$

$$\text{such that } A(r)x(r) = b \quad (22)$$

$$x(r) \geq 0 \quad (23)$$

where an initial value for r is available, which we denote as r_0 . We refer to the verification problem at r_0 as the *nominal problem*, and its solution the *nominal solution*.

In order to determine the impact of a change in r on the worst-case voltage drop, the brute-force approach would be to re-solve the verification problem at every value of r , but this is too expensive. Instead, we propose an approach where the nominal solution at r_0 is used to directly find the worst case solutions in a neighborhood around it. This approach will rely on the validity of the optimal basis at r_0 in a neighborhood around r_0 . In fact, Theorem 2, along with (23), can be used to determine the boundary of the neighborhood around r_0 in which the same optimal basis remains valid, as we will see below.

V. PROPOSED SOLUTION

Suppose we identify a neighborhood around r_0 in which $y \geq 0$, throughout. For any r in this neighborhood, let us maintain the *same*

basis \mathcal{B} that was found as optimal at r_0 , so that $c_{\mathcal{R}} = 0$ is also maintained. If we can find a *basic feasible solution* at r , then, with $c_{\mathcal{R}} = 0$ and $y \geq 0$, this $x(r)$ must be *optimal*, by Theorem 2. This can be achieved by, requiring (22) to ensure that $x(r)$ is a *solution*, requiring (23) to ensure it is *feasible*, and setting $x_{\mathcal{R}}(r) = 0$ to ensure that $x(r)$ is *basic*. Using (15) and (16), this leads to:

$$v_k(r) = c_{\mathcal{B}}^T x_{\mathcal{B}}(r) \quad (24)$$

$$B(r)x_{\mathcal{B}}(r) = b \quad (25)$$

which, as long as $x_{\mathcal{B}}(r) \geq 0$, gives us directly the optimal solution of the LP at r .

This leads us to a revised definition of a (possibly smaller) neighborhood around r_0 , determined by:

$$y(r) \geq 0 \quad \text{and} \quad x_{\mathcal{B}}(r) \geq 0 \quad (26)$$

Throughout this neighborhood, the solution of (24) and (25) is the optimal solution of the LP. We will call this neighborhood the *safety region* and the points along its boundary the *breakpoints*. Thus, we see that the notion of the safety region is crucial to an efficient approach, because it allows us to discover the solution of the LP efficiently, by simply solving the linear system in (25).

Our approach follows from the single and multi-variable Taylor series expansions of $x_{\mathcal{B}}$ and y around the initial operating point r_0 , which we use to determine the breakpoints. Because the conductance matrix is linear in r , then B and R are also linear in r and the second derivatives of G , B , and R with respect to r are always zero. This fact will be useful in the following sections. Furthermore, to simplify the notation, we will drop the arguments r or r_0 in connection with the matrices like G , B , and R .

A. Single Parameter Variations

In this section, we assume that there is only one parameter that can be changed, i.e., r consists of a single element; it is a scalar. This makes the safety region simply an interval around r_0 in one dimension.

1) *Voltage Drop Estimation:* The Taylor series expansion for $x_{\mathcal{B}}(r)$ around the nominal point r_0 gives:

$$x_{\mathcal{B}}(r) = x_{\mathcal{B}}(r_0) + \sum_{i=1}^{\infty} \frac{x_{\mathcal{B}}^{(i)}(r_0)}{i!} (r - r_0)^i \quad (27)$$

where $x_{\mathcal{B}}^{(i)}(r_0)$ is the i^{th} derivative of $x_{\mathcal{B}}(r)$, evaluated at r_0 . It is shown in [7] that this derivative is given by:

$$x_{\mathcal{B}}^{(i)}(r_0) = (-1)^i i! \left(B^{-1} \frac{dB}{dr} \right)^i x_{\mathcal{B}}(r_0) \quad (28)$$

for $i \geq 1$. We can also write (28) as:

$$x_{\mathcal{B}}^{(i)}(r_0) = -i \left(B^{-1} \frac{dB}{dr} \right) x_{\mathcal{B}}^{(i-1)}(r_0) \quad (29)$$

Combining (27) with (24), leads to:

$$v_k(r) = v_k(r_0) + c_{\mathcal{B}}^T \sum_{i=1}^{\infty} \frac{x_{\mathcal{B}}^{(i)}(r_0)}{i!} (r - r_0)^i \quad (30)$$

In practice, the summation does not need to be carried out to infinity. In fact, in all test cases that we have seen, the nonlinearity is not very severe at all, so that truncation can be done quite accurately, and we write:

$$v_k(r) = v_k(r_0) + c_{\mathcal{B}}^T \sum_{i=1}^N \frac{x_{\mathcal{B}}^{(i)}(r_0)}{i!} (r - r_0)^i \quad (31)$$

where N is a small integer, in practice about 3. In order to make use of this result, we need to 1) have $x_{\mathcal{B}}(r_0)$, which we already know from the nominal solution, 2) evaluate dB/dr at r_0 , and this can be easily done because the dependence of G on r is well known based on the construction of G from element stamps during modified nodal analysis (MNA), and 3) to be able to compute the product of B^{-1} by a vector, which we can do using LU factorization. As a result, we have an easy-to-evaluate, polynomial expression for the solution $v_k(r)$ throughout the neighborhood. It remains to discover the safety region.

2) *Safety Region Estimation*: In order to discover the safety region, we will develop an expression for $y(r)$. We start by writing the Taylor series expansion for:

$$y(r) = R^T B^{-T} c_{\mathcal{B}} \quad (32)$$

as we did for $x_{\mathcal{B}}(r)$, which gives:

$$y(r) = y(r_0) + \sum_{i=1}^{\infty} \frac{y^{(i)}(r_0)}{i!} (r - r_0)^i \quad (33)$$

We define:

$$\pi(r) = B^{-T} c_{\mathcal{B}} \quad (34)$$

so that

$$y(r) = R^T \pi(r) \quad (35)$$

Claim. For $i \geq 1$, the i^{th} derivative of $y(r)$ with respect to r is given by:

$$y^{(i)}(r) = i \frac{dR^T}{dr} \pi^{(i-1)}(r) + R^T \pi^{(i)}(r) \quad (36)$$

Proof: We will prove this claim by induction. The basis case, for $i = 1$, is trivially true due to (35), which gives:

$$y^{(1)}(r) = \frac{dR^T}{dr} \pi(r) + R^T \pi^{(1)}(r) \quad (37)$$

Now, assuming the claim is true for $i - 1$, then:

$$y^{(i-1)}(r) = (i-1) \frac{dR^T}{dr} \pi^{(i-2)}(r) + R^T \pi^{(i-1)}(r) \quad (38)$$

Let us take the derivative of this equation:

$$\begin{aligned} y^{(i)}(r) &= (i-1) \frac{dR^T}{dr} \pi^{(i-1)}(r) + \frac{dR^T}{dr} \pi^{(i-1)}(r) \\ &\quad + R^T \pi^{(i)}(r) \\ &= i \frac{dR^T}{dr} \pi^{(i-1)}(r) + R^T \pi^{(i)}(r) \end{aligned} \quad (39)$$

which is the desired result. \blacksquare

As was done for $x_{\mathcal{B}}$, we can write the i^{th} derivative of $\pi(r)$ with respect to r at r_0 as:

$$\pi^{(i)}(r_0) = -i \left(B^{-T} \frac{dB^T}{dr} \right) \pi^{(i-1)}(r_0) \quad (40)$$

If we insert this in (36) at r_0 and reorder, we get:

$$y^{(i)}(r_0) = i \left(\frac{dR^T}{dr} - R^T B^{-T} \frac{dB^T}{dr} \right) \pi^{(i-1)}(r_0) \quad (41)$$

In (27) and (33) we model the behavior of $x_{\mathcal{B}}(r)$ and $y(r)$ based on their Taylor series expansions around r_0 . In practice these Taylor series can be safely truncated, up to an appropriate order, say N . Since we require all the entries of these two vectors to remain nonnegative,

we can equate the resulting N^{th} order polynomials to zero, and find their roots, i.e., $\forall j$:

$$x_{\mathcal{B}_j}(r_0) + \sum_{i=1}^N \frac{x_{\mathcal{B}_j}^{(i)}(r_0)}{i!} (r - r_0)^i = 0 \quad (42)$$

$$y_j(r_0) + \sum_{i=1}^N \frac{y_j^{(i)}(r_0)}{i!} (r - r_0)^i = 0 \quad (43)$$

From this, the smallest root to the right of r_0 and the largest root to the left of r_0 determine the breakpoints. Between these two breakpoints, we are guaranteed that $y(r) \geq 0$ and $x_{\mathcal{B}}(r) \geq 0$.

B. Multi-Parameter Variations

In this section we will generalize our findings to the case of a vector of parameters. This corresponds to changing the widths of different metal branches, possibly by different amounts.

1) *Voltage Drop Estimation*: We can write the multivariable Taylor series expansion for $x_{\mathcal{B}}(r)$ around r_0 using multi-index notation (see Appendix 1) as:

$$x_{\mathcal{B}}(r) = x_{\mathcal{B}}(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^{\alpha} x_{\mathcal{B}}(r_0)}{\alpha!} (r - r_0)^{\alpha} \quad (44)$$

Similar to (29) in the single parameter case, $\partial^{\alpha} x_{\mathcal{B}}(r)$ is given by (see Appendix 2 for proof):

$$\partial^{\alpha} x_{\mathcal{B}}(r) = \sum_{|\beta|=1} -i_{\beta} B^{-1} \partial^{\beta} B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) \quad (45)$$

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

Using (24), we can express the voltage drop in the safety region as:

$$v_k(r) = v_k(r_0) + c_{\mathcal{B}} \sum_{|\alpha|=1}^N \frac{\partial^{\alpha} x_{\mathcal{B}}(r_0)}{\alpha!} (r - r_0)^{\alpha} \quad (46)$$

2) *Safety Region Estimation*: Let us write the multivariable Taylor series expansion for $y(r) = R^T \pi(r)$ around the initial point r_0 as:

$$y(r) = y(r_0) + \sum_{|\alpha|=1}^{\infty} \frac{\partial^{\alpha} y(r_0)}{\alpha!} (r - r_0)^{\alpha} \quad (47)$$

Similar to (36), $\partial^{\alpha} y(r_0)$ is given by:

$$\partial^{\alpha} y(r) = \left(\sum_{|\beta|=1} i_{\beta} \partial^{\beta} R^T \partial^{\alpha-\beta} \pi(r) \right) + R^T \partial^{\alpha} \pi(r) \quad (48)$$

where the previous definition of i_{β} is valid (see Appendix 2 for proof). And, $\partial^{\alpha} \pi(r)$ is given by:

$$\partial^{\alpha} \pi(r_0) = \sum_{|\beta|=1} -i_{\beta} B^{-T} \partial^{\beta} B^T \partial^{\alpha-\beta} \pi(r_0) \quad (49)$$

which has the same form as (45). As a result, the safety region is defined by the values which make all the elements of the vectors defined in (44) and (47) stay nonnegative. In practice, these expressions can be truncated up to an order N . Therefore, at the breakpoints, at least one entry, j , of one of these vectors satisfy:

$$x_{\mathcal{B}_j}(r_0) + \sum_{|\alpha|=1}^N \frac{\partial^{\alpha} x_{\mathcal{B}_j}(r_0)}{\alpha!} (r - r_0)^{\alpha} = 0 \quad (50)$$

$$y_j(r_0) + \sum_{|\alpha|=1}^N \frac{\partial^{\alpha} y_j(r_0)}{\alpha!} (r - r_0)^{\alpha} = 0 \quad (51)$$

These are N^{th} order polynomials in r . For example, in the case when $N = 3$, and say, there are two parameters under consideration, i.e. $r = (r_1, r_2)$ and $r_0 = (r_{10}, r_{20})$, (50) and (51) will have the form:

$$\begin{aligned}
& m_{111j}(r_1 - r_{10})^3 + m_{112j}(r_1 - r_{10})^2(r_2 - r_{20}) \\
& + m_{122j}(r_1 - r_{10})(r_2 - r_{20})^2 + m_{222j}(r_2 - r_{20})^3 \\
& + m_{11j}(r_1 - r_{10})^2 + m_{12j}(r_1 - r_{10})(r_2 - r_{20}) \\
& + m_{22j}(r_2 - r_{20})^2 \\
& + m_{1j}(r_1 - r_{10}) + m_{2j}(r_2 - r_{20}) \\
& + m_{0j} \\
& = 0
\end{aligned} \tag{52}$$

where the m_s are constants that depend on the partial derivatives of $x_{Bj}(r)$ and $y_j(r)$.

The breakpoint in a given direction is given by the intersection of these equations and the direction vector. For example, if the direction vector is $r = (u_1t, u_2t)$ where $u_1^2 + u_2^2 = 1$, (52) will be a third order polynomial in t :

$$n_{3j}t^3 + n_{2j}t^2 + n_{1j}t + n_{0j} = 0 \tag{53}$$

where the n_s are constants that depend on the m_s in (52). We can easily solve these equations for all entries in x_B and y to find the smallest t corresponding to the breakpoint.

C. Nonlinear Optimization

Once we have parametric equations for the voltage drop and the breakpoints, we can construct a nonlinear optimization problem to minimize the voltage drop in the safety region:

$$\text{minimize } v_k(r) = c_B^T x_B(r) \tag{54}$$

$$\text{such that } x_B(r) \geq 0 \tag{55}$$

$$y(r) \geq 0 \tag{56}$$

where $x_B(r)$ and $y(r)$ are given by (27) and (33) in the single parameter case, and (44) and (47) in the multiparameter case by truncating them up to an order N .

A number of nonlinear optimization algorithms are given in [8]. We used the steepest descent line search method with cubic interpolation to determine the step length in our implementations.

Note that the safety region represents the region where the optimal basis (found at the nominal solution) remains optimal. If the algorithm reaches a breakpoint as the minimizer, we can re-solve the LP given by (21-23) to determine the new optimal basis and restate the functions to reduce the voltage drop in the new safety region.

Algorithm 1 describes the reduction of the worst-case voltage drop at a given node. It starts with the solution of the nominal linear program to determine the optimal basis. Using this basis, the expressions for the voltage drop estimation and the safety region are computed. Then, the nonlinear optimization algorithm is employed. First, the search direction is found. Then, the maximum step length that can be taken is calculated by finding the breakpoint in that direction. Using the maximum step length, an appropriate step length that reduces voltage drop is computed. If a step length that reduces the voltage drop cannot be found, the algorithm exits. Using the step length and the direction, the parameter values are updated and the new voltage drop is calculated. If the maximum step is taken, the LP is re-solved to get the optimal basis in the new region. This procedure is repeated until the voltage drop on the node in consideration is less than the threshold. It remains to check that *all* node voltages are also now below the threshold.

Algorithm 1 Nonlinear Optimization

```

1:  $r_{req} = r_0$ 
2: while ( $v_{max} > v_{th}$ ) do
3:   Solve the LP given in (21-23) using Simplex for at  $r = r_{req}$  and
   get the optimal basis
4:   Find the expression for  $v_k(r)$  using (31) or (46)
5:   Find the safety region expressions using (27) and (33), or (44) and
   (47)
6:    $max\_step\_taken = FALSE$ 
7:   while ( $max\_step\_taken$  is  $FALSE$ ) do
8:      $p = FIND\_SEARCH\_DIRECTION(r_{req})$ 
9:      $\lambda_{max} = FIND\_MAX\_STEPLENGTH(p, r_{req})$ 
10:     $\lambda = FIND\_STEP\_LENGTH(p, r_{req}, \lambda_{max})$ 
11:     $r_{req} = r_{req} + \lambda p$ 
12:     $v_{max} = v_k(r_{req})$ 
13:    if ( $\lambda = \lambda_{max}$ ) then
14:       $max\_step\_taken = TRUE$ 
15: return  $r_{req}$ 

```

D. Top Level Algorithm

Recall that a robust grid refers to a grid in which the highest voltage drop among *all* nodes is less than a given threshold value. However, Algorithm 1 was focused on a single node. Algorithm 2 describes the overall procedure. It starts with identification of the maximum worst-case voltage drop on the grid and the offending node. If it is less than the threshold, the grid is deemed safe. If not, the required parameter values that result in the maximum voltage drop on that node to be less than the threshold are found using Algorithm 1. Then, the grid is re-verified using the new parameter values to check if any other nodes exceed the threshold, and the procedure is repeated until the maximum voltage drop on the grid is less than the threshold. In all cases that we tested, we found that no other nodes ever exceeded the threshold, once the initial maximum node voltage was reduced below the threshold. Thus, a single run of Algorithm 1 was enough in all cases and, therefore, there was no practical benefit in re-verifying the whole grid after Algorithm 1 had terminated.

Algorithm 2 Full grid correction

```

1:  $grid\_unsafe = TRUE$ 
2:  $r_f = r_0$ 
3: while ( $grid\_unsafe$ ) do
4:   Verify all nodes at  $r = r_f$  to identify the node,  $k$ , with the
   maximum worst-case voltage drop,  $v_{max}$ .
5:   if ( $v_{max} \leq v_{th}$ ) then
6:      $grid\_unsafe = FALSE$ 
7:   else
8:     Find the required change in parameter,  $r_{req}$ , to reduce the
     maximum voltage drop at  $k$ , below  $v_{th}$  using Algorithm 1
9:      $r_f = r_{req}$ 

```

VI. EXPERIMENTAL RESULTS

The grid correction algorithm was implemented in C++. The parameters were taken to be the widths of metal lines on different layers. A number of test grids were generated based on user specifications, including grid dimensions, metal layers (M1-M9), pitch and width per layer, and current source distribution. For the full grid verification, the DC equivalent of the algorithm in [9] was used. The computations were carried out on a 64-bit Linux machine with 8GB memory. As it stands, this algorithm can suggest changes of width that may be extremely small, such as 1% increase or less on many parameters. For practical reasons, it makes more sense to change the width by

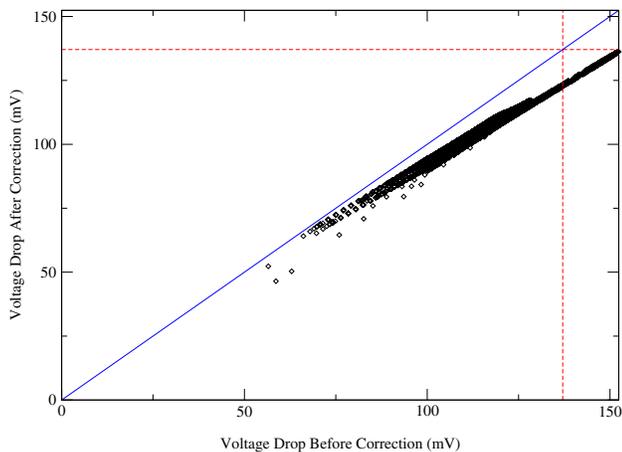
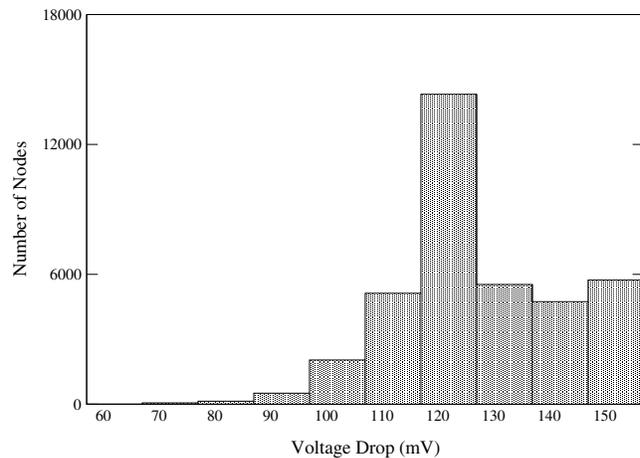


Fig. 1. Correlation plot of voltage drops before and after correction for 38,201 node grid.

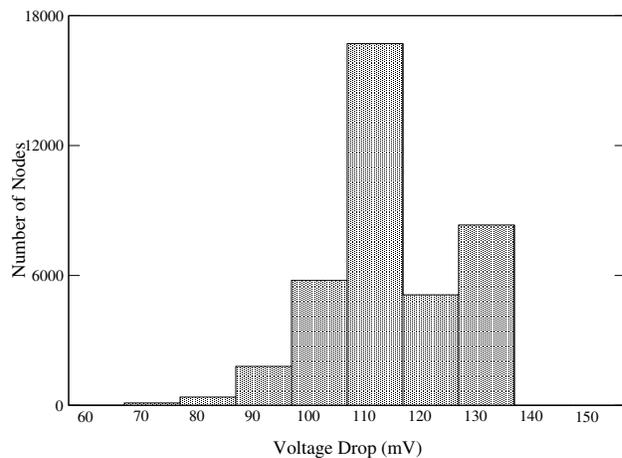
only some significant increment, say 5% or 10% increase, and not to bother with very small increases. We can accommodate this easily by rounding the values returned by the algorithm to, say, the nearest 5% or 10% setting. Our implementation includes this optional rounding step, and, as a sanity check, we apply a final check that this node voltage drop is still below threshold, before leaving Algorithm 1. In all test cases that we have run, we have not found that any further work is required, and the node voltage has always remained below the threshold.

We tested the multi-parameter variations on a number of grids, varying the number of parameters between 6-18. The maximum voltage drops on the grids before and after correction are given in Table I. We also show the number of Simplex solutions required for Algorithm 1, the runtime of individual node correction, and the total runtime of Algorithm 2. The individual node correction time is the time it takes for Algorithm 1 to reduce the voltage drop on the node with the initial maximum voltage drop. We see that the grids were corrected in a reasonable amount of time. The total runtime takes into account the full grid verifications done before the correction to determine the node with the maximum voltage drop and after the correction to see if any other voltage drop requires reduction. It can be improved with faster grid verification techniques.

Table II gives the required changes in grid parameters to reduce the maximum voltage drops on the grid below the threshold. We can see that, in some cases, only some parameters may need to be changed, which means the voltage drop is more sensitive to those parameters. It can be seen that the required area increase is modest, generally below 10% for the test cases that we tested. In the absence of an approach like ours, the only option available today for correcting the grid is, perhaps, to simply increase the metal widths everywhere. It is clear that, generally, an increase in metal width everywhere will fix the grid. However, with no further guidance as to which metal widths should be increased, the required overall area increase can be substantial. We have tested this, for the grids under study, by increasing all metal line widths until the grid becomes robust, and the overall area increase is reported in the last column of Table II, as



(a) Before correction



(b) After correction

Fig. 2. Voltage drop histograms before and after correction for 38,201 node grid. All drops have been reduced below the threshold of 137 mV.

“Fixed area cost”. The overall metal area increase is large, ranging from 15% to 20%, and this area overhead can seriously complicate signal line routing. Thus, the value of our approach is that it allows one to selectively and intelligently increase metal width to achieve grid robustness.

In Figure 1, the correlation plot of the voltage drops before and after the correction for the 38,101 node grid is given. We see that the algorithm successfully reduced all the voltage drops exceeding the threshold. Finally, the histograms of the voltage drops before and after the correction is given in Figure 2.

TABLE I
MAXIMUM VOLTAGE DROPS ON THE GRIDS BEFORE AND AFTER CORRECTION AND THE RUNTIMES 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
1,407	113.02	99.90	2	3.3s	5.56m
6,757	123.56	111.2	5	37.2s	59.33m
13,685	91.55	80.43	2	3.55m	50.57m
26,228	59.70	53.27	14	23.83m	14.53h
38,201	152.30	136.37	4	21.3m	32.9h
67,712	129.48	115.4	21	5.41h	4.97d

TABLE II
AREA COST COMPARISON OF THE PROPOSED APPROACH WITH THE BRUTE-FORCE CORRECTION APPROACH

Number of nodes	Required Change in Parameter (%)																		Total area cost	Fixed area cost
	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	r16	r17	r18		
1,407	10	20	10	20	0	5													9.68 %	15 %
6,757	0	10	0	10	0	10	0	20											1.11 %	15 %
13,685	5	5	20	20	5	5	20	20	10	0									8.54 %	15 %
26,228	5	5	20	10	0	0	20	10	0	0	15	10	35	5					5.62 %	20 %
38,201	5	5	20	10	0	0	25	15	0	0	5	0	0	0	10	5	35	5	5.43 %	15 %
62,712	5	5	15	10	0	0	20	15	0	0	0	0	0	0	5	5	35	0	5.24 %	20 %

VII. CONCLUSION

Voltage drop on the power grid is a key concern for design of modern integrated circuits. It is traditionally done by comparing the voltage drops on the grid with a certain threshold that guarantees reliable circuit performance. In this work, we propose a novel method to correct the grid with minimal change when some nodes exceed the given threshold. It builds on linear programming and the current constraints-based verification approach, and formulates the problem as a non-linear optimization problem to find the required change in metal widths that reduces the maximum voltage drop on the grid below the threshold. We only considered the DC currents in this work; correction in the case of transient currents is a part of ongoing research.

APPENDIX 1

A multi-index [10] $\alpha = (\alpha_1, \dots, \alpha_n)$ is an n -tuple of nonnegative integers. Here are the definitions of some common mathematical concepts used in multi-index notation. The norm:

$$|\alpha| = \alpha_1 + \dots + \alpha_n$$

The factorial:

$$\alpha! = \prod_{i=1}^n \alpha_i!$$

The power of a vector $x = (x_1, \dots, x_n)$:

$$x^\alpha = \prod_{i=1}^n x_i^{\alpha_i}$$

Similarly, the partial differential operator:

$$\partial^\alpha = \left(\frac{\partial}{\partial x_1} \right)^{\alpha_1} \dots \left(\frac{\partial}{\partial x_n} \right)^{\alpha_n} = \prod_{i=1}^n \frac{\partial^{\alpha_i}}{\partial x_i^{\alpha_i}}$$

For example if $\alpha = (3, 1, 2)$, then $|\alpha| = 3 + 1 + 2 = 6$ and $\alpha! = 3!1!2! = 12$. If $x = (-4, 2, 1)$, then $x^\alpha = (-4)^3 2^1 1^2 = -128$.

Next, we give the multivariable Taylor series expansion using multi-index notation:

$$f(x) = f(x_0) + \sum_{|\alpha|=1}^{\infty} \partial^\alpha f(x_0) \frac{(x - x_0)^\alpha}{\alpha!}$$

Here, x and x_0 are vectors of length n and the sum is over all norms of multi-indices of length n . The Taylor series expansion of order N sums the terms up to the multi-indices of norm N .

For illustration we will “unpack” Taylor series expansion of order 3, in the case when $n = 2$. Let $x = (u, v)$, $x_0 = (u_0, v_0)$, and $(h, k) = (u - u_0, v - v_0)$. There are two indices $\alpha = (1, 0)$ and $\alpha = (0, 1)$ that have $|\alpha| = 1$. Their contribution to the expression will be:

$$\frac{\partial}{\partial u} f(x_0) \frac{h^1 k^0}{1!0!} + \frac{\partial}{\partial v} f(x_0) \frac{h^0 k^1}{1!0!} = f_u(x_0)h + f_v(x_0)k$$

There are three indices which have norm 2. They are $\alpha = (2, 0)$, $\alpha = (1, 1)$, and $\alpha = (0, 2)$. They contribute:

$$\begin{aligned} f_{uu}(x_0) \frac{h^2 k^0}{2!0!} + f_{uv}(x_0) \frac{h^1 k^1}{1!1!} + f_{vv}(x_0) \frac{h^0 k^2}{2!0!} \\ = f_{uu}(x_0) \frac{h^2}{2} + f_{uv}(x_0)hk + f_{vv}(x_0) \frac{k^2}{2} \end{aligned}$$

Finally, there are four multi-indexes of norm 3: $\alpha = (3, 0)$, $\alpha = (2, 1)$, $\alpha = (1, 2)$, and $\alpha = (0, 3)$. They contribute:

$$\begin{aligned} f_{uuu}(x_0) \frac{h^3 k^0}{3!0!} + f_{uuv}(x_0) \frac{h^2 k^1}{2!1!} \\ + f_{uvv}(x_0) \frac{h^1 k^2}{1!2!} + f_{vvv}(x_0) \frac{h^0 k^3}{0!3!} \\ = f_{uuu}(x_0) \frac{h^3}{6} + f_{uuv}(x_0) \frac{h^2 k}{2} + f_{uvv}(x_0) \frac{h k^2}{2} + f_{vvv}(x_0) \frac{k^3}{6} \end{aligned}$$

APPENDIX 2

Claim.

$$\partial^\alpha x_B(r) = \sum_{|\beta|=1} -i_\beta B^{-1} \partial^\beta B \partial^{\alpha-\beta} x_B(r) \quad (57)$$

where

$$Bx_{\mathcal{B}}(r) = b \quad (58)$$

Proof: Since B is independent of β , we can rearrange (57) as:

$$\sum_{|\beta|=1} i_{\beta} \partial^{\beta} B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) + B \partial^{\alpha} x_{\mathcal{B}}(r) = 0 \quad (59)$$

We will prove this claim by induction. Let us assume we have p parameters. The base case is for $\alpha = e_j = (0, \dots, 1, \dots, 0)$. Here, the only index β with non-zero i_{β} is $\beta = e_j$, thus (59) reads:

$$\frac{\partial B}{\partial r_j} x_{\mathcal{B}}(r) + B \frac{\partial x_{\mathcal{B}}(r)}{\partial r_j} = 0 \quad (60)$$

which gives the same result as taking the derivative of (58) with respect to r_j .

Next, we assume that (59) is true for $\alpha - e_j = (\alpha_1, \dots, \alpha_j - 1, \dots, \alpha_p)$:

$$\sum_{|\beta|=1} i'_{\beta} \partial^{\beta} B \partial^{\alpha-e_j-\beta} x_{\mathcal{B}}(r) + B \partial^{\alpha-e_j} x_{\mathcal{B}}(r) = 0 \quad (61)$$

Here,

$$i'_{\beta} = \begin{cases} \alpha_k - 1, & \beta = e_k = e_j \\ \alpha_k, & \text{otherwise} \end{cases} \quad (62)$$

Now, let us take the derivative of (61) with respect to r_j :

$$\sum_{|\beta|=1} i'_{\beta} \partial^{\beta} B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) + \frac{\partial B}{\partial r_j} \partial^{\alpha-e_j} x_{\mathcal{B}}(r) + B \partial^{\alpha} x_{\mathcal{B}}(r) = 0 \quad (63)$$

Here, the second term:

$$\frac{\partial B}{\partial r_j} \partial^{\alpha-e_j} x_{\mathcal{B}}(r) = \partial^{\beta} B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) \quad (64)$$

for $\beta = e_j$. Therefore, we can include it in the summation. This adds 1 to the previously defined i'_{β} when $\beta = e_j$, making $i_{\beta} = \alpha_k, \forall k$. As a result, we can rewrite (63) as:

$$\sum_{|\beta|=1} i_{\beta} \partial^{\beta} B \partial^{\alpha-\beta} x_{\mathcal{B}}(r) + B \partial^{\alpha} x_{\mathcal{B}}(r) = 0 \quad (65)$$

which is the same as (59), concluding the proof. ■

Claim.

$$\partial^{\alpha} y(r) = \left(\sum_{|\beta|=1} i_{\beta} \partial^{\beta} R^T \partial^{\alpha-\beta} \pi(r) \right) + R^T \partial^{\alpha} \pi(r) \quad (66)$$

where

$$y(r) = R^T \pi(r) \quad (67)$$

Proof: Once more, we will prove this claim by induction. The base case for $\alpha = e_j$ gives:

$$\frac{\partial y(r)}{\partial r_j} = \frac{\partial R^T}{\partial r_j} \pi(r) + R^T \frac{\partial \pi(r)}{\partial r_j} \quad (68)$$

which is trivially true from the derivative of (67) with respect to r_j .

Next, let us assume the claim is true for $\alpha - e_j = (\alpha_1, \dots, \alpha_j - 1, \dots, \alpha_p)$:

$$\partial^{\alpha-e_j} y(r) = \left(\sum_{|\beta|=1} i'_{\beta} \partial^{\beta} R^T \partial^{\alpha-e_j-\beta} \pi(r) \right) + R^T \partial^{\alpha-e_j} \pi(r) \quad (69)$$

Here, i'_{β} s are defined the same as in (62). If we take the derivative of with respect to r_j , we get:

$$\begin{aligned} \partial^{\alpha} y(r) &= \left(\sum_{|\beta|=1} i'_{\beta} \partial^{\beta} R^T \partial^{\alpha-\beta} \pi(r) \right) \\ &\quad + \frac{\partial R^T}{\partial r_j} \partial^{\alpha-e_j} \pi(r) + R^T \partial^{\alpha} \pi(r) \end{aligned} \quad (70)$$

Here, the second term:

$$\frac{\partial R^T}{\partial r_j} \partial^{\alpha-e_j} \pi(r) = \partial^{\beta} R^T \partial^{\alpha-\beta} \pi(r) \quad (71)$$

for $\beta = e_j$. Thus, it can be moved inside the summation sign, by adding 1 to the previously defined i'_{β} when $\beta = e_j$, making $i_{\beta} = \alpha_k, \forall k$. Consequently, we can rewrite (70) as:

$$\partial^{\alpha} y(r) = \left(\sum_{|\beta|=1} i_{\beta} \partial^{\beta} R^T \partial^{\alpha-\beta} \pi(r) \right) + R^T \partial^{\alpha} \pi(r) \quad (72)$$

which is the same as (66). ■

REFERENCES

- [1] D. Kouroussis and F. N. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *ACM/IEEE 40th Design Automation Conference (DAC-03)*, pages 99–104, Anaheim, CA, June 2-6 2003.
- [2] S. U. Chowdhury and M. A. Breuer. Minimal area design of power/ground nets having graph topologies. *IEEE Transactions on Circuits and Systems*, CAS-34(12):1441–1451, December 1987.
- [3] R. Dutta and M. Marek-Sadowska. Automatic sizing of power/ground (p/g) networks in VLSI. In *ACM/IEEE 26th Design Automation Conference (DAC-89)*, pages 783–789, June 25-29 1989.
- [4] J. N. Kozhaya, S. R. Nassif, and F. N. Najm. A multigrid-like technique for power grid analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(10):1148–1160, October 2002.
- [5] I. Maros. *Computational Methods of the Simplex Method*. Springer, 2003.
- [6] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1998.
- [7] R. M. Freund. Postoptimal analysis of a linear program under simultaneous changes in matrix coefficients. *Mathematical Programming Study*, 24:1–13, 1985.
- [8] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.
- [9] N. H. Abdul Ghani and F. N. Najm. Fast vectorless power grid verification using an approximate inverse technique. In *ACM/IEEE 46th Design Automation Conference (DAC-09)*, pages 184–189, July 26-31 2009.
- [10] X. S. Raymond. *Elementary Introduction to the Theory of Pseudodifferential Operators*. CRC Press, 1991.