Early P/G Grid Voltage Integrity Verification[*]

Mehmet Avci
Department of ECE
University of Toronto
Toronto, Ontario, Canada
mehmet.avci@utoronto.ca

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

ABSTRACT

As part of power delivery network verification, one should check if the voltage fluctuations exceed some critical threshold. In this work, we consider the power and ground grids together and describe an early verification approach under the framework of current constraints where tight lower and upper bounds on worst-case voltage fluctuations are computed via linear programs. Experimental results indicate that the proposed technique results in errors in the range of a few mV.

1. INTRODUCTION

The feature size of modern integrated circuits (ICs) has been dramatically reduced in order to improve speed, power and cost. The scaling of CMOS is expected to continue for at least another decade and future nanometer circuits will contain billions of transistors [1]. As CMOS technology is scaled, the power supply voltage will continue to decrease [1]. With reduced supply voltages and more functions integrated into ICs, the impact of voltage fluctuation is increasing and voltage integrity is becoming a big concern for chip designers.

There are many sources of on-chip voltage fluctuations, such as *IR*-drop, *Ldi/dt* drop, and the resonance between the on-chip grid and the package. Most available grid verification techniques use some form of circuit simulation to simulate the grid. Such an approach requires full knowledge of the current waveforms drawn by underlying transistor circuitry. These waveforms would then be used to simulate the grid and to find the voltage fluctuation at each node. However, since the number of possible circuit behaviours is very large, one needs to simulate the grid for a large number of vector sequences at each node, which is prohibitively expensive. Another disadvantage of the simulation based approach is that it does not allow the designer to perform early grid verification, when grid modifications can be most easily done. To overcome these problems, we will adopt the notion of *current constraints* [2] to capture the uncertainty about the circuit details and behaviours. Under these constraints, grid verification becomes a problem of computing the worst-case voltage fluctuations subject to current constraints.

In the literature, the ground grid has always been assumed to be symmetric to the power grid. In [3], the authors claim that the power and ground grids have the same electrical requirements and therefore the structures of these grids are often symmetric, particularly at the initial and intermediate phases of the design. They show that this symmetry can be exploited in a way to reduce the complexity of the power delivery network by an introduction of a virtual ground. The resulting circuit model contains two independent symmetric grids, and therefore the analysis of only one circuit is necessary. However, the assumption that the power and ground grids are symmetric is not reliable, since even in initial stages of the design, some regions of the power delivery network are removed to make way for signal routing. This introduces non-symmetry in the grid, which might lead to erroneous results if symmetry is assumed. We note in particular that the presence of non-symmetry can cause the voltage on a given node of the grid to fluctuate in both directions, i.e. voltage drop and overshoot, even for an RC grid (for an RC model of the power grid, voltage levels can normally only be below v_{dd}, under the assumption that the circuit does not inject current into the power grid). To see why, consider the simple unsymmetrical 5-node grid shown in Figure 1. Figure 2 shows the current waveform assigned to the current source in the circuit and the node voltage at node A as a result of an HSPICE simulation. The simulation shows an overshoot where the node voltage at node A goes above v_{dd}. Therefore, there is a necessity to verify the power and ground grids together (i.e. a P/G grid verification), which is the focus of this paper.

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

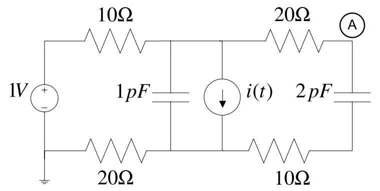

Figure 1: A 5-node grid

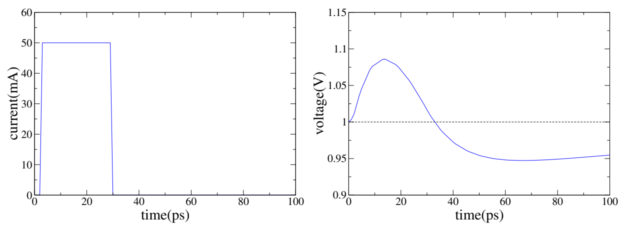

Figure 2: Current configuration resulting in voltage overshoot for node A

The remainder of the paper is organized as follows. In section 2, we present the P/G grid model and formulate the problem under the notion of current constraints. Section 3 proposes an efficient solution for the lower and upper bounds on the worst-case voltage fluctuations of the P/G grid. Implementation details are given in section 4 followed by experimental results in section 5. Finally, we conclude the paper in section 6.

2. PROBLEM FORMULATION

2.1 P/G Grid Model

We consider an RC model of the P/G grid where each branch is represented either by a resistor or capacitor. We define the nodes that are on the power grid as power grid nodes, and similarly, the nodes that are on the ground grid as ground grid nodes. Resistors are located between two power grid nodes or between two ground grid nodes, i.e. no resistor exists between a power and a ground grid node. In addition, the capacitors are located only between power and ground grid nodes and, unlike previous work, we assume that a node can have multiple capacitors.

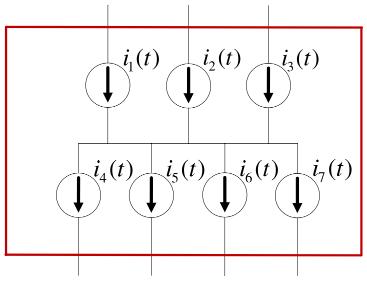

Figure 3: Macroblock model

One common simplification in the literature is to model the current drawn by the underlying transistor circuitry in a logic block as a single current source. We usually know the current drawn by a logic block, but that logic block is usually attached to multiple nodes in the grid. Therefore, modeling the current drawn by a logic block as a single current source is not valid, and yields pessimistic voltage fluctuations. In order to capture this notion, we introduce the model of a *macroblock*, which groups multiple current sources into a single block as shown in Figure 3.

The macroblock model in Figure 3 captures the true behaviour of a logic block, in the sense that it draws current from multiple nodes. Note that we do not require having the same number of current sources for power and ground grid nodes. However, for each macroblock, we have to ensure that the current leaving power grid nodes must equal the current entering ground grid nodes. This will be an important equality constraint that defines the feasibility space of currents.

A simple P/G grid is shown in Figure 4. Notice that the macroblock has multiple connections to the grid, and that some nodes have multiple capacitors attached.

2.2 The System Equations

Let the P/G grid consist of $n+\alpha$ nodes where nodes $1,2,\dots,n$ have no voltage sources attached, and nodes $(n+1),\dots,(n+\alpha)$ are connected to voltage sources. Following the approach in [4], the MNA equation that describes the network can be written as:

$$\tilde{G}\tilde{u}(t) + \tilde{C}\dot{\tilde{u}}(t) = \tilde{i}(t) + \tilde{G}_0 v_{dd(n)} \tag{1}$$

where $\tilde{G}$ and $\tilde{G}_0$ are $n \times n$ conductance matrices and $\tilde{C}$ is an $n \times n$ capacitance matrix. $\tilde{u}(t)$ is an $n \times 1$ vector of node voltages except the nodes that are connected to voltage sources. $\tilde{i}(t)$ is $n \times 1$ vector of current sources and $v_{dd(n)}$ is an $n \times 1$ vector whose each entry is equal to the value of the supply voltage. The node voltages $\tilde{u}(t)$ are with respect to some reference (datum) node that is part of the ground grid.

Let h be the number of nodes in the power grid, and l be the number of nodes in the ground grid, so that $h + l = n$. We can rewrite (1) by partitioning the vector of node voltages with respect to power and ground grid nodes, and reordering the rows and columns of $\tilde{G}$, $\tilde{C}$, $\tilde{G}_0$ and the entries of $\tilde{i}(t)$ accordingly as follows:

$$\begin{bmatrix} G_p & 0 \\ 0 & G_g \end{bmatrix} \begin{bmatrix} u_p(t) \\ u_g(t) \end{bmatrix} + \begin{bmatrix} C_p & N \\ N^T & C_g \end{bmatrix} \begin{bmatrix} \dot{u}_p(t) \\ \dot{u}_g(t) \end{bmatrix} = \begin{bmatrix} -i_p(t) \\ i_g(t) \end{bmatrix} + G_0 v_{dd(n)} \tag{2}$$

Since no resistor exists between power and ground grid nodes, $\tilde{G}$ can be partitioned into two submatrices G_p ($h \times h$ matrix) and G_g ($l \times l$ matrix). $i_p(t)$ ($h \times 1$ vector) and $i_g(t)$ ($l \times 1$ vector) are non-negative vectors defining the current sources attached to power and ground grid nodes, respectively. Since capacitors exist only between power and ground grid nodes, C_p ($h \times h$ matrix) and C_g ($l \times l$ matrix) are non-negative diagonal matrices, and N is an $h \times l$ non-positive matrix.

If we set all current sources to 0, $\forall t$, then $u_p(t) = v_{dd(h)}$ and $u_g(t) = 0$, $\forall t$, where $v_{dd(h)}$ is an $h \times 1$ vector whose each entry is equal to the value of the supply voltage. For this case, the system of equations becomes:

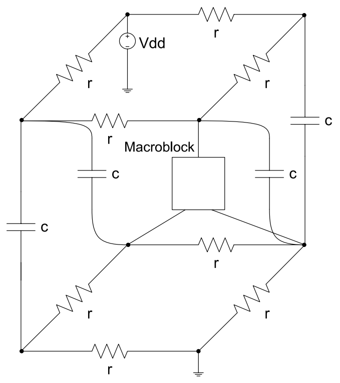

Figure 4: The P/G grid model

$$\begin{bmatrix} G_p & 0 \\ 0 & G_g \end{bmatrix} \begin{bmatrix} v_{dd(h)} \\ 0 \end{bmatrix} = G_0 v_{dd(n)} \tag{3}$$

Substituting $G_0 v_{dd(n)}$ from (3) into (2) and rearranging the terms, we obtain:

$$\begin{bmatrix} G_p & 0 \\ 0 & G_g \end{bmatrix} \begin{bmatrix} u_p(t) - v_{dd(h)} \\ u_g(t) \end{bmatrix} + \begin{bmatrix} C_p & N \\ N^T & C_g \end{bmatrix} \begin{bmatrix} \dot{u}_p(t) \\ \dot{u}_g(t) \end{bmatrix} = \begin{bmatrix} -i_p(t) \\ i_g(t) \end{bmatrix} \tag{4}$$

Defining $v_p(t) = v_{dd(h)} - u_p(t)$ to be the vector of voltage drops at power grid nodes, we can write (4) as two separate equations:

$$G_p v_p(t) + C_p \dot{v}_p(t) - N\dot{u}_g(t) = i_p(t) \tag{5}$$

$$G_g u_g(t) + C_g \dot{u}_g(t) - N^T \dot{v}_p(t) = i_g(t) \tag{6}$$

In matrix notation, (5) and (6) can be combined to yield:

$$\begin{bmatrix} G_p & 0 \\ 0 & G_g \end{bmatrix} \begin{bmatrix} v_p(t) \\ u_g(t) \end{bmatrix} + \begin{bmatrix} C_p & -N \\ -N^T & C_g \end{bmatrix} \begin{bmatrix} \dot{v}_p(t) \\ \dot{u}_g(t) \end{bmatrix} = \begin{bmatrix} i_p(t) \\ i_g(t) \end{bmatrix} \tag{7}$$

In this notation, $v_p(t)$ is positive when power grid nodes experience undershoots and $u_g(t)$ is positive when ground grid nodes experience overshoots as shown in Figure 5. Now define:

$$\hat{G} = \begin{bmatrix} G_p & 0 \\ 0 & G_g \end{bmatrix}, \quad \hat{v}(t) = \begin{bmatrix} v_p(t) \\ u_g(t) \end{bmatrix}$$
$$\hat{C} = \begin{bmatrix} C_p & -N \\ -N^T & C_g \end{bmatrix}, \quad \hat{i}(t) = \begin{bmatrix} i_p(t) \\ i_g(t) \end{bmatrix}$$

So that (7) becomes:

$$\hat{G}\hat{v} + \hat{C}\dot{\hat{v}}(t) = \hat{i}(t) \tag{8}$$

Notice that the circuit described by (8) is the original P/G grid, but with all the voltage sources set to zero and the directions of current sources attached to the power grid nodes reversed. Furthermore, this equation is useful, because now the current vector $\hat{i}(t)$ is a non-negative vector and the matrix $\hat{C}$ consists of only non-negative elements.

Now assume that only m of n nodes of the P/G grid have current sources attached. Then we can reorder the rows and columns of the matrices and the entries of the vectors in (8) to yield:

$$Gv(t) + C\dot{v}(t) = \begin{bmatrix} i(t) \\ 0_{(n-m)} \end{bmatrix} = \bar{i}(t) \tag{9}$$

where G and C are matrices of size $n \times n$, that are simply reordered replicas of $\hat{G}$ and $\hat{C}$. $i(t)$ is the vector of size m representing the current loads, and $0_{(n-m)}$ is the zero-vector of size $n-m$. Finally,

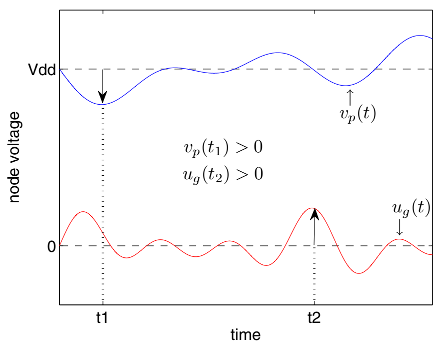

Figure 5: Voltages on the P/G grid

using the backward Euler formula, (9) can be discretized in time as:

$$Av(t) = Bv(t - \Delta t) + \bar{i}(t) \tag{10}$$

where $A = (G + \frac{C}{\Delta t})$ and $B = \frac{C}{\Delta t}$.

2.3 Current Constraints

We adopt the notion of current constraints in order to perform verification of the P/G grid. This approach [2] does not require complete information about the currents drawn by the underlying circuitry, and may be called a vectorless approach. The currents are typically hard to specify for at least two reasons. First, the number of combinations of possible current waveforms is very large, and simulation of a large set of waveforms is very time-consuming. Second, the simulation approach does not allow the designer to verify the grid early in the chip design. For the simulation, the details of the underlying circuitry must be already known, but it might not be available or complete early in the design, when most of the major changes in grid characteristics can be most easily incorporated.

We use three types of constraints: local constraints, global constraints and equality constraints. Local constraints define upper bounds on individual current sources. They can be expressed mathematically as:

$$0 \le i(t) \le i_L \tag{11}$$

where i_L is a vector of size m and stands for the peak value of currents that the current sources can draw. In this paper, we restrict our work to the case of DC constraints, i.e. the upper bound is fixed over time. However, note that it is only the constraints that are DC, the currents themselves are transient. An alternative is to use transient current constraints, which are more difficult to use in practice, both from the user's standpoint (supplying transient constraints) and the verification tools that would deal with them.

Local constraints do not completely capture the behaviour of the grid, because it is never the case that all chip components draw their currents simultaneously. Therefore, we need global constraints, which are upper bounds on the sums of groups of current sources. They might represent the maximum current that the group of current sources in each macroblock can draw or the peak total power dissipation of a group of macroblocks. If we assume that we have μ global constraints, then they can be expressed as:

$$0 \le Ui(t) \le i_G \tag{12}$$

where U is a $\mu \times n$ matrix that consists only of 0s and 1s. If a 1 is present in a row of U, it indicates that the corresponding current source is included in that global constraint. Similar to the case of local constraints, i_G is a constant time-independent DC constraint, but the currents themselves are transient waveforms.

As previously mentioned, we need to ensure that the currents leaving the power grid are equal to currents entering the ground grid, which we will call an equality constraint. If we assume that we have γ macroblocks, then the equality constraints can be expressed as:

$$Mi(t) = 0 \tag{13}$$

where M is a $\gamma \times n$ matrix that only consists of 1s, -1s and 0s. For each macroblock, 1s correspond to current sources that are attached to the power grid, and -1s correspond to current sources that are attached to the ground grid.

To simplify the notation, we will use $\mathcal{F}$ to denote the feasible space of currents, so that $i(t) \in \mathcal{F}$ if and only if it satisfies (11), (12) and (13) at all time.

2.4 Problem Definition

Our problem is to find, for every node, the worst-case node voltage fluctuation over all possible currents in $\mathcal{F}$. To simplify the notation, let $E = A^{-1}$ and $D = A^{-1}B$, so that we can write (10) as:

$$v(t) = Dv(t - \Delta t) + E\bar{i}(t) \tag{14}$$

We first write the matrix E as follows:

$$E = [e_1, e_2, \ldots, e_n] \tag{15}$$

where e_i is the ith column of E. Now define:

$$H = [e_1, e_2, \ldots, e_m] \tag{16}$$

where H is $n \times m$ matrix formed by the first m columns of E. Since we know that the last $n - m$ elements in $\bar{i}(t)$ are 0, we can write (14) as:

$$v(t) = Dv(t - \Delta t) + Hi(t) \tag{17}$$

Now consider the case in which the grid had no stimulus for $t \le 0$, which leads to $v_0 = v(0) = 0$. Then, writing at time Δt, $2\Delta t$ and $3\Delta t$, we obtain:

$$v(\Delta t) = Dv_0 + Hi(\Delta t) = Hi(\Delta t) \tag{18}$$

$$v(2\Delta t) = Dv(\Delta t) + Hi(2\Delta t) = DHi(\Delta t) + Hi(2\Delta t) \tag{19}$$

$$\begin{aligned} v(3\Delta t) &= Dv(2\Delta t) + Hi(3\Delta t) \\ &= D^2Hi(\Delta t) + DHi(2\Delta t) + Hi(3\Delta t) \end{aligned} \tag{20}$$

Repeating this procedure for any future time $p\Delta t$, we have:

$$v(p\Delta t) = \sum_{k=0}^{p-1} D^k Hi((p-k)\Delta t) \tag{21}$$

At every point in time $t \in [0, p\Delta t]$, the input vector $i(t)$ must be feasible, i.e. we must have $i(t) \in \mathcal{F}$. Under these conditions, we are interested in the worst-case voltage fluctuations attained (separately) by each component of $v(p\Delta t)$. In order to capture this notion, we use the following notation, introduced in [5].

Suppose $f(c) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a vector function whose components are denoted $f_1(c), \ldots, f_n(c)$, and let $\mathcal{A} \subset \mathbb{R}^n$. Now, define a vector $x \in \mathbb{R}^n$, such that, with $i \in \{1, 2, \ldots, n\}$, x_i is the maximum of $f_i(c)$ over all $c \in \mathcal{A}$. We will denote this by the following operator:

$$x = \underset{\forall c \in \mathcal{A}}{\text{emax}}(f(c)) \tag{22}$$

Notice that each component $x_i, \forall i = 1, \ldots, n$ may be found separately by solving the following maximization problem:

$$\begin{aligned} &\text{maximize: } f_i(c) \\ &\text{subject to: } c \in \mathcal{A} \end{aligned} \tag{23}$$

Similarly, define a vector $y \in \mathbb{R}^n$, such that y_i is the minimum of $f_i(c)$ over all $c \in \mathcal{A}$. We will denote this by the following operator:

$$y = \underset{\forall c \in \mathcal{A}}{\text{emin}}(f(c)) \tag{24}$$

and each component $y_i, \forall i = 1, \ldots, n$ may be found separately by solving the following minimization problem:

$$\begin{aligned} &\text{minimize: } f_i(c) \\ &\text{subject to: } c \in \mathcal{A} \end{aligned} \tag{25}$$

Using the emax$(\cdot)$ and emin$(\cdot)$ operators, we can express the worst-case voltage fluctuation at all nodes at time $p\Delta t$ by:

$$v^+(p\Delta t) = \underset{\forall i(t) \in \mathcal{F}}{\text{emax}} \left(\sum_{k=0}^{p-1} D^k H i((p-k)\Delta t) \right) \tag{26}$$

$$v^-(p\Delta t) = -\underset{\forall i(t) \in \mathcal{F}}{\text{emin}} \left(\sum_{k=0}^{p-1} D^k H i((p-k)\Delta t) \right) \tag{27}$$

where the notation $\forall i(t) \in \mathcal{F}$ means that, for every time point $t \in [0, p\Delta t]$, the current vector $i(t)$ satisfies all the (local, global and equality) constraints. $v^+(t)$ is a non-negative vector defining the worst-case voltage drops on power grid nodes and the worst-case voltage overshoots on ground grid nodes, and similarly, $v^-(t)$ is a non-negative vector defining the worst-case voltage overshoots on power grid nodes and the worst-case voltage drops on ground grid nodes. We used a minus sign in front of emin$(\cdot)$ operator in (27) to avoid confusion about the notion of the lower and upper bounds in the rest of the paper. Using $v^+(t)$ and $v^-(t)$, $v(t)$ can be bounded as:

$$-v^-(t) \le v(t) \le v^+(t) \tag{28}$$

Although the RC model is dynamic, i.e., its currents and voltages vary with time, the constraints are DC and do not depend on time. Hence, $\mathcal{F}$ is the same for each time step. With this, the components of (26) and (27) can be decoupled [5], leading to:

$$v^+(p\Delta t) = \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i \right] \tag{29}$$

$$v^-(p\Delta t) = -\sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emin}} \left[D^k H i \right] \tag{30}$$

where i is simply an $m \times 1$ vector of variables that satisfies the (local, global and equality) constraints, without reference to any particular point in time. This is an important simplification of the problem, as it has the advantage that the number of constraints for each optimization is fixed and does not span all previous time points. The advantage of using the matrix H instead of E is clear now, since at each time step one needs to compute multiplication of an $n \times n$ matrix with an $n \times m$ matrix instead of two $n \times n$ matrices. Furthermore, the optimization variables do not include the redundant variables.

CLAIM 1. $\underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i \right] \ge 0$ *and* $\underset{\forall i \in \mathcal{F}}{\text{emin}} \left[D^k H i \right] \le 0$, $\forall k$.

PROOF. Let $x^* = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i \right]$ and $y^* = \underset{\forall i \in \mathcal{F}}{\text{emin}} \left[D^k H i \right]$. Assume that the claim is not true, i.e. $x^*_j < 0$ and $y^*_j > 0$. Since $i^\dagger = 0$ is feasible, i.e. $i^\dagger \in \mathcal{F}$, we can choose $i^\dagger$, so that $\left(D^k H i^\dagger\right)_j = 0 > x^*_j$ and $\left(D^k H i^\dagger\right)_j = 0 < y^*_j$, which is a contradiction. This completes the proof. $\square$

Using (29), (30) and claim 1, and taking the difference in two consecutive time steps, we have:

$$v^+(p\Delta t) - v^+((p-1)\Delta t) = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^{p-1} H i \right] \ge 0 \tag{31}$$

$$v^-(p\Delta t) - v^-((p-1)\Delta t) = -\underset{\forall i \in \mathcal{F}}{\text{emin}} \left[D^{p-1} H i \right] \ge 0 \tag{32}$$

meaning that $v^+(p\Delta t)$ and $v^-(p\Delta t)$ are monotone non-decreasing functions of the time point p, for any integer $p \ge 1$.

In practice, we are interested in the steady state solution where the system becomes independent of the initial condition $v(t) = 0, \forall t \le 0$. Since the RC grid model is a dynamical system with a limited memory of its past, the steady state solution can be obtained by evaluating (29) and (30) at points far away from the initial condition, i.e. as $p \to \infty$. Thus, the general solution to the problem is:

$$v^+(\infty) = \lim_{p \to \infty} \sum_{k=0}^{p-1} \underset{\forall i(t) \in \mathcal{F}}{\text{emax}} \left[D^k H i \right] \tag{33}$$

$$v^-(\infty) = -\lim_{p \to \infty} \sum_{k=0}^{p-1} \underset{\forall i(t) \in \mathcal{F}}{\text{emin}} \left[D^k H i \right] \tag{34}$$

3. PROPOSED SOLUTION

Using (33) and (34) is intractable, because they have to be evaluated for a large number of time steps until convergence is achieved and the emax$(\cdot)$ and emin$(\cdot)$ operators require linear programs proportional to the number of nodes in the grid, which for modern designs is in the order of millions. As an alternative, we propose an efficient solution to compute lower and upper bounds on the worst-case voltage fluctuations of the P/G grid.

3.1 Vector of Lower Bounds

We will show that, for specific initial conditions, both $v^+(t)$ and $v^-(t)$ will be monotone non-decreasing functions of time t. We have found that the DC verification solution of the grid is a good initial condition, which satisfies the monotonicity property.

3.1.1 Non-zero initial conditions

We start by investigating the impact of starting the verification with different (non-zero) initial conditions on the worst-case voltage fluctuations. If we have the initial condition v_0 at time $t = 0$, then the voltage on the grid at any future time $p\Delta t$ can be expressed as:

$$v(p\Delta t) = D^p v_0 + \sum_{k=0}^{p-1} D^k H i((p-k)\Delta t) \tag{35}$$

Because the RC grid is a stable linear system and because the backward difference approximation used in (10) is absolutely stable [6], it follows that for $i(t) = 0, \forall t$ and any bounded initial condition v_0, equation (35) converges to 0 as $t \to \infty$. For $i(t) = 0, \forall t$, the voltage on the grid at any time $p\Delta t$ can be written as:

$$v(p\Delta t) = D^p v_0 \tag{36}$$

Writing (36) as $p \to \infty$, we get:

$$v(\infty) = \lim_{p \to \infty} v(p\Delta t) = \lim_{p \to \infty} D^p v_0 = 0 \tag{37}$$

Since (37) is valid for any bounded initial condition, it is clear that $D^p \to 0$ as $p \to \infty$. We will use the following theorem [7] to conclude that this actually means $\rho(D) < 1$, where $\rho(D)$ is the magnitude of the largest eigenvalue of D, also called the spectral radius of D.

THEOREM 1. *Let D be a square matrix. Then, the sequence D^k, for $k = 0, 1, \ldots,$ converges to zero if and only if $\rho(D) < 1$.*

It is easy to see that at steady state, i.e. as $p \to \infty$, choosing a different initial condition other than $v_0 = 0$ does not have any impact on $v(\infty)$, because of the fact that D^p converges to zero as $p \to \infty$. Therefore, (21) and (35) become equivalent as $p \to \infty$. Using the notation in the previous section and using the fact that $\mathcal{F}$ is the same for each time step, we can express the worst-case voltage fluctuations at time point $p\Delta t$ with the initial condition v_0 as:

$$v^+(p\Delta t) = D^p v_0 + \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i \right] \tag{38}$$

$$v^-(p\Delta t) = D^p v_0 - \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emin}} \left[D^k H i \right] \tag{39}$$

3.1.2 A monotone non-decreasing $v^+(t)$

Under the DC model of the P/G grid, (9) becomes:

$$Gv = \bar{i} \tag{40}$$

Now let $L = G^{-1}$ and let K to be a $n \times m$ matrix, which is obtained as the first m columns of L, such that:

$$K = [l_1, l_2, \ldots, l_m] \tag{41}$$

where l_i defines ith column of L. Using the matrix K, we can write $v = Ki$. Now assume that we have the DC solution of the system as the initial condition, leading to:

$$v_0 = Ki_0 \tag{42}$$

We define the voltage vector v_0 to be feasible, if it satisfies (42) for a current vector $i_0 \in \mathcal{F}$.

CLAIM 2. *If v_0 is feasible, then $v^+(p\Delta t)$ given in (38) is a monotone non-decreasing function of the time point p, for any integer $p \geq 1$.*

PROOF. The claim is true if we can show that $v^+(p\Delta t) \geq v^+((p-1)\Delta t)$, for any integer $p \geq 1$. Substituting v_0 from (42) into (38), we obtain:

$$v^+(p\Delta t) = D^p K i_0 + \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i\right] \tag{43}$$

Substituting $D^p K$ from (58) (see Appendix) into (43), we get:

$$v^+(p\Delta t) = (K - \sum_{k=0}^{p-1} D^k H) i_0 + \sum_{k=0}^{p-1} \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^k H i\right] \tag{44}$$

Taking the difference in two consecutive time steps, we have:

$$v^+(p\Delta t) - v^+((p-1)\Delta t) = \underset{\forall i \in \mathcal{F}}{\text{emax}} \left[D^{p-1} H i\right] - D^{p-1} H i_0 \tag{45}$$

We see in (45) that the emax$(\cdot)$ operator assigns the maximum value to the first term on the right-hand side over all $i \in \mathcal{F}$, whereas the second term on the right-hand side has the variables $i_0 \in \mathcal{F}$, which may not result in the optimal solution. Therefore, we conclude that if v_0 is feasible, then $v^+(p\Delta t) \geq v^+((p-1)\Delta t)$, for any integer $p \geq 1$. This completes the proof. $\square$

3.1.3 DC initial condition

Using the notation $(X)_j$ to define jth row of a matrix X and incorporating (38), (39) and (42), we can express the worst-case voltage fluctuation for the jth node at time $p\Delta t$ as:

$$v_j^+(p\Delta t) = (D^p K)_j i_0 + \sum_{k=0}^{p-1} \max_{\forall i \in \mathcal{F}} (D^k H)_j i \tag{46}$$

$$v_j^-(p\Delta t) = (D^p K)_j i_0 - \sum_{k=0}^{p-1} \min_{\forall i \in \mathcal{F}} (D^k H)_j i \tag{47}$$

A good choice of i_0 for a given node would be the current combination that leads to the worst-case voltage fluctuation for that node under the DC model of the P/G grid. The worst-case voltage fluctuation for the jth node under the DC model is given by:

$$v_j^+ = \max_{\forall i \in \mathcal{F}} (K)_j i \tag{48}$$

$$v_j^- = -\min_{\forall i \in \mathcal{F}} (K)_j i \tag{49}$$

Denote the optimal value of i of the maximization problem in (48) as i_j^+ ($m \times 1$ vector) and that of the minimization problem in (49) as i_j^- ($m \times 1$ vector). Since G is an M-matrix [5], its inverse consists of only non-negative elements, which means that K is a non-negative matrix. Therefore, the result of the minimization problem in (49) under non-negative current constraints will be 0, which leads to $i_j^- = 0$. Using i_j^+ and i_j^- as the initial current at $t = 0$ for the jth node, (46) and (47) become:

$$v_j^+(p\Delta t) = (D^p K)_j i_j^+ + \sum_{k=0}^{p-1} \max_{\forall i \in \mathcal{F}} (D^k H)_j i \tag{50}$$

$$v_j^-(p\Delta t) = -\sum_{k=0}^{p-1} \min_{\forall i \in \mathcal{F}} (D^k H)_j i \tag{51}$$

Algorithm 1 Lower Bound Algorithm

Inputs: $K, D, H, \mathcal{F}$ and ϵ
Outputs: $v_{lb}^+(\infty)$ and $v_{lb}^-(\infty)$

```
1: Compute $v^+ = \underset{\forall i \in \mathcal{F}}{\text{emax}}[Ki]$ and save $i_j^+$ for each node
2: Set $p = 0$ and stop_flag = false
3: while stop_flag = false do
4:     $p = p + 1$
5:     for $j = 1, \ldots, n$ do
6:         Compute $v_j^+(p\Delta t)$ using (50) and $v_j^-(p\Delta t)$ using (51)
7:     end for
8:     if $v^+(p\Delta t) - v^+((p-1)\Delta t) < \epsilon$ then
9:         if $v^-(p\Delta t) - v^-((p-1)\Delta t) < \epsilon$ then
10:            Set stop_flag = true
11:            Set $v_{lb}^+(\infty) = v^+(p\Delta t)$ and $v_{lb}^-(\infty) = v^-(p\Delta t)$
12:        end if
13:    end if
14: end while
```

3.1.4 Lower bound

Algorithm 1 describes the computation of the lower bound vector in detail, based on (50) and (51). Using claim 2 and claim 1, we can see that $v_j^+(p\Delta t)$ in (50) and $v_j^-(p\Delta t)$ in (51) are monotone non-decreasing functions of the time point p, for any integer $p \geq 1$. Since we stop the main loop of Algorithm 1 for a finite p, when $v^+(p\Delta t) - v^+((p-1)\Delta t)$ and $v^-(p\Delta t) - v^-((p-1)\Delta t)$ are less than a threshold ϵ, then $v_{lb}^+(\infty)$ and $v_{lb}^-(\infty)$ are clearly lower bounds on $v^+(\infty)$ and $v^-(\infty)$, respectively.

3.2 Vector of Upper Bounds

Following an approach similar to [8], we compute an upper bound on the worst-case voltage fluctuations of the P/G grid. Although the upper bound in [8] was derived for an RLC model of the power grid, it can be shown that it is also valid for the P/G grid model presented in this paper. Due to space considerations, and because it is mostly an adaptation of [8] to the P/G grid case, we will not show the background work associated with the derivation of the upper bound. Instead, we simply describe the computation of the upper bound on the worst-case voltage fluctuations in Algorithm 2, in which we use the notation $|X|$ to denote the matrix of the element-wise absolute values of the entries of a matrix X. Further details can be found in [8].

4. IMPLEMENTATION

4.1 Inverse Approximation Method

It is obvious from section 3 that the inverse of the system matrix A is needed for the computation of the lower and upper bounds, because $D = A^{-1}B$. For the lower bound, we need to invert the conductance matrix G as well as A. We now explain how this can be efficiently done.

Power distribution networks have a mesh structure, where a node has a small number of neighbouring nodes. Such a structure results in a matrix (i.e. A or G), that is sparse, symmetric, positive definite, and banded. In the literature, it is well known

Algorithm 2 Upper Bound Algorithm

```
Inputs: $D, H$ and $\mathcal{F}$
Outputs: $v_{ub}^{+}(\infty)$ and $v_{ub}^{-}(\infty)$
 1: Set $Q = I$, $s^{+} = 0$ and $s^{-} = 0$
 2: while $\|Q\|_1 \geq 1$, $\|Q\|_\infty \geq 1$ and $\|Q\|_F \geq 1$ do
 3:     $s^{+} := s^{+} + \underset{\forall i \in \mathcal{F}}{\text{emax}}\,[QHi]$
 4:     $s^{-} := s^{-} - \underset{\forall i \in \mathcal{F}}{\text{emin}}\,[QHi]$
 5:     $Q := QD$
 6: end while
 7: Set $R = \frac{1}{2}(|Q| - Q)$
 8: Set $P = \begin{bmatrix} Q+R & R \\ R & Q+R \end{bmatrix}$
 9: Compute $v_{ub}^{+}(\infty)$ and $v_{ub}^{-}(\infty)$ using:
10:         LU-factorize $(I-P)$: $(I-P) = L \cdot U$
11:         Forward solve: $Lw = \begin{bmatrix} s^{+} \\ s^{-} \end{bmatrix}$
12:         Backward solve: $U \begin{bmatrix} v_{ub}^{+}(\infty) \\ v_{ub}^{-}(\infty) \end{bmatrix} = w$
```

that the inverse of a non-singular sparse matrix is dense. Especially, a matrix that results from a mesh structure has the inverse that is almost full. However, it is also well known in the literature that the inverse of a sparse, symmetric positive definite and banded matrix has entries whose values decay exponentially as one moves away from the diagonal [9]. This fact is the main idea of constructing sparse approximate inverse preconditioners to precondition large sparse linear systems when using an iterative method such as conjugate gradient method. Sparse approximate inverse preconditioners try to find a good sparse approximate inverse M, such that $MA \approx I$, where I denotes the identity matrix.

There are numerous sparse approximate inverse techniques in the literature. One of them is SPAI [10], which is based on Frobenius norm minimization. SPAI starts with an arbitrary initial matrix M and iteratively refines its columns by minimizing the Frobenius norm $\|MA - I\|_F$. This technique has been applied to power delivery network verification in [5]. Another technique is called AINV [11], which is based on the conjugate Gram-Schmidt (or A-orthogonalization) process. AINV has the advantage of using the fact that the matrix whose inverse is to be approximated is symmetric positive definite, while SPAI is a general sparse approximate preconditioner for unsymmetric matrices.

In what follows, we will briefly explain the AINV method given in [11]. Assume that A is an $n \times n$ symmetric positive definite matrix. It is shown in [11] that the factorization of A^{-1} can be obtained from a set of conjugate directions $z_1, z_2, \ldots, z_n$ for A. By writing a set of conjugate directions in matrix form, we have:

$$Z = [z_1, z_2, \ldots, z_n] \tag{52}$$

where Z is the matrix whose ith column is z_i. Knowing a set of conjugate directions for A, we can write:

$$Z^T A Z = D = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix} \tag{53}$$

where $d_i = z_i^T A z_i$. To obtain a factorization of A^{-1}, we write:

$$A^{-1} = Z D^{-1} Z^T \tag{54}$$

Notice in (54) that the inverse of A can be obtained easily if we know a set of conjugate directions $z_1, z_2, \ldots, z_n$. In [11], a set of conjugate directions is constructed by means of a conjugate Gram-Schmidt (or A-orthogonalization) process applied to any set of linearly independent vectors $v_1, v_2, \ldots, v_n$. The Gram-Schmidt process is a method for orthogonalizing a set of vectors

Algorithm 3 AINV Inverse Factorization Algorithm

```
Inputs: $A$
Outputs: $A^{-1}$
 1: Let $z_i^{(0)} = e_i$ for $1 \leq i \leq n$
 2: for $i = 1, \ldots, n$ do
 3:     for $j = i, \ldots, n$ do
 4:         $d_j^{(i-1)} = (A)_j z_j^{(i-1)}$
 5:     end for
 6:     if $i = n$ go to step 11
 7:     for $j = i+1, \ldots, n$ do
 8:         $z_j^{(i)} = z_j^{(i-1)} - \frac{d_j^{(i-1)}}{d_i^{(i-1)}} z_i^{(i-1)}$
 9:     end for
10: end for
11: Let $z_i = z_i^{(i-1)}$ and $d_i = d_i^{(i-1)}$ for $1 \leq i \leq n$
12: Set $Z = [z_1, z_2, \ldots, z_n]$
13: Set $D^{-1} = \begin{bmatrix} 1/d_1 & 0 & \cdots & 0 \\ 0 & 1/d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/d_n \end{bmatrix}$
14: Set $A^{-1} = Z D^{-1} Z^T$
```

in an inner product space of size n. It takes a finite, linearly independent set of vectors $v_1, v_2, \ldots, v_n$ and generates an orthogonal set $u_1, u_2, \ldots, u_n$ that spans the same inner product space n. For further details on the Gram-Schmidt process, the reader is referred to [12].

Since the Gram-Schmidt process can be applied to any set of linearly independent vectors, it is convenient to choose $v_i = e_i$, where e_i is the ith unit vector. From the Cholesky factorization of A, we have:

$$A = LDL^T \tag{55}$$

where L is unit lower triangular, which leads to $Z = L^{-T}$. Since the inverse of a unit lower triangular matrix is a unit lower triangular matrix, it follows that Z is unit upper triangular.

The factorization algorithm is given in Algorithm 3, in which the ith row of A is denoted by $(A)_j$. For a dense matrix this algorithm requires roughly twice as much work as Cholesky factorization [11]. Although for a sparse matrix the cost can be significantly reduced, the method is still expensive because the resulting matrix Z tends to be dense. However, the sparsity can be preserved by reducing the amount of fill-in occurring in the computation of z-vectors. Reducing the amount of fill-in can be achieved by ignoring all fill-in outside selected positions in Z or by discarding fill-ins whose magnitude is less than a tolerance δ.

In [13], the authors propose several strategies and special data structures to implement Algorithm 3 efficiently. We have used many aspects of their implementation and we have made some modifications in their data structures to be able to access entries in a matrix which is in compressed column storage (CCS) format. Since we do not know the sparsity pattern of the inverse upfront, we have only used the strategy in which we discarded fill-in occurring in the computation of z-vectors whose magnitude was less than $\delta = e^{-6}$.

Experimental results [14] show that AINV is more effective and faster than the other approximate inverse methods. Therefore, we have adopted it for our implementation.

4.2 Network Simplex Method

We will show that the linear programs in our formulation can be efficiently solved with the help of a network simplex method. Using the notation given in the previous sections, we have the following linear program for each node:

Table 1: Runtime and accuracy of the proposed technique

Grid	Lower Bound Algorithm		Upper Bound Algorithm		
nodes	runtime	time steps	runtime	time steps	maximum absolute error
102	4.13 sec.	5	8.48 sec.	4	$1.75mV$
437	39.33 sec.	4	57.49 sec.	3	$2.52mV$
1052	1.98 min.	4	2.43 min.	5	$2.67mV$
4285	14.66 min.	5	16.73 min.	4	$2.59mV$
18,472	1.57 h.	6	2.14 h.	3	$2.65mV$
31,974	3.74 h.	7	4.73 h.	4	$1.45mV$
76,753	9.45 h.	6	10.68 h.	4	$2.72mV$
192,974	18.55 h.	3	23.16 h.	3	$1.39mV$

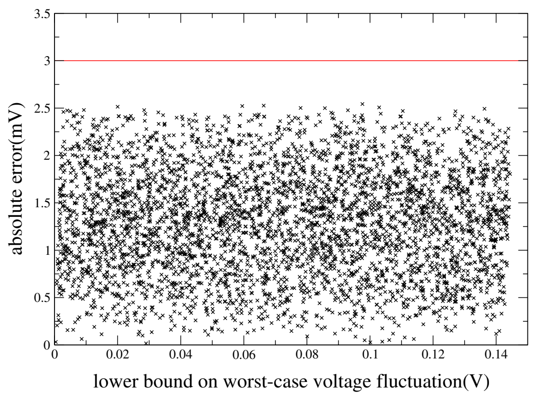

Figure 6: Absolute error comparison for all nodes of a 4285-node grid

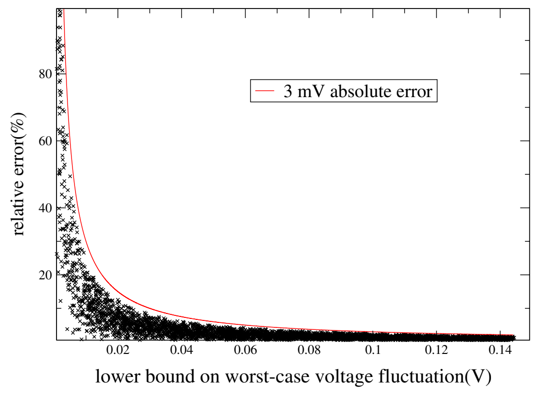

Figure 7: Accuracy of the proposed technique

$$\begin{aligned} \text{maximize / minimize: } & f(i) \\ \text{subject to: } & 0 \le Ui \le i_G \\ & Mi = 0 \\ & 0 \le i \le i_L \end{aligned} \tag{56}$$

where $f(i) : \mathbb{R}^n \rightarrow \mathbb{R}$ is the linear objective function of i. To simplify the notation, we augment the matrices U and M into the matrix T, and we augment the vectors i_G and the zero-vector of size γ into the vector a:

$$T = \begin{bmatrix} U \\ M \end{bmatrix}, \qquad a = \begin{bmatrix} i_G \\ 0 \end{bmatrix}$$

where T is a matrix of size $(\mu + \gamma) \times n$, and a is a vector of size $(\mu + \gamma) \times 1$. With this notation the linear program (56) can be rewritten as:

$$\begin{aligned} \text{maximize / minimize: } & f(i) \\ \text{subject to: } & 0 \le Ti \le a \\ & 0 \le i \le i_L \end{aligned} \tag{57}$$

The constraint matrix T consists of entries which are 1s, -1s or 0s. It resembles the node-arc incidence matrix (NAIM) of a network, in the sense that NAIM also has entries which are 1s, -1s or 0s. This is the key observation that allows us to formulate the optimization problem (57) as a network flow problem. In our optimization problem, the equality constraints define *flow conservation constraints* of the network flow problem, whereas local constraints define *capacity constraints* on the flow along the edges in the network. Furthermore, global constraints define *side constraints* on the sum of the flows along the edges. For a more detailed discussion of network flow problems, the reader is referred to [15].

Network flow problems can be efficiently solved with the help of the network simplex method. Empirical results have shown that the method is significantly faster than the standard simplex method, when applied to the same network problem [16]. Furthermore, it is shown in [17] that significant computational speed up can be achieved if closely related instances of network flow problems are solved sequentially. This observation is quite beneficial for our problem in the sense that the feasibility space of currents remains the same at each instance of the optimization problem. Thus, we have the same optimization problem for each node except different objective functions.

5. EXPERIMENTAL RESULTS

To test our method, we have implemented Algorithms 1, 2, and 3 in C++. We have set $\epsilon = e^{-5}$ to stop the main loop of Algorithm 1. To solve the required linear programs, Algorithms 1 and 2 use the network simplex method of the Mosek optimization package [18]. We have used the hot-start option of the Mosek network simplex solver for fast objective function switching. Several experiments were conducted on a set of test grids, which were generated from user specifications, which include grid dimensions, metal layers (M1-M9), pitch and width per layer, and C4 and current source distribution. Minimum spacing, and sheet and via resistances were specified according to a 65 nm technology. A global constraint is specified for each macroblock, and additional global constraints were specified covering the entirety of the grid area. The computations were carried on a 64-bit Linux machine with 8 GB memory.

Table 1 shows the speed and the accuracy of our proposed solution technique for the computation of the vectors of lower and upper bounds on the worst-case voltage fluctuations. The results are compared with each other and the maximum absolute difference between the upper and the lower bound vector is reported in column 6, where the absolute error is defined as $(v_{ub}(\infty) - v_{lb}(\infty))$. The results show that our solution technique resulted in a maximum absolute error of $2.72mV$ across all nodes of all test grids. The number of time steps shown in column 3 and 5 reports the number of time steps for which the lower and upper bound algorithms converge. The runtime for each one of two methods is also shown in the table.

Figure 6 shows a scatter plot with the lower bound on the worst-case voltage fluctuation on one axis and the absolute error on the other axis, for a 4285-node grid. The figure shows that the absolute error between the upper and the lower bound is very small, meaning that the proposed method is very accurate. For the same grid, Figure 7 shows a scatter plot with the lower bound on the worst-case voltage fluctuation on one axis and the relative error on the other axis, where the relative error is defined as $(v_{ub}(\infty) - v_{lb}(\infty))/v_{lb}(\infty)$. The figure also shows the curve corresponding to $3mV$ absolute error where a point on the curve represents a node that has $3mV$ difference between its upper and lower bound. Note that the relative error can be high, but only for small values of voltage fluctuations, and the absolute error does not exceed $3mV$.

Looking at the runtime results given in Table 1, we notice that the computation of the lower and upper bound vectors of a 18,472-node grid takes around 4 hours. Such a grid is of course small compared to full-chip grids containing millions of nodes. However, the proposed method is applicable for early grid verification, where the size of the grids is normally not as large. The power of our approach is that it finds tight upper and lower bounds on the worst-case voltage fluctuations under *all* feasible current combinations. It is a unique approach that offers this type of guarantee.

6. CONCLUSION

Voltage fluctuations of the power delivery network is a key concern for modern chip design. In this paper, we presented an early grid verification technique that takes both power and ground grids into account. Our proposed solution approach formulates both upper and lower bounds on the worst-case voltage fluctuations of the P/G grid under the framework of current constraints. Experimental results show that the proposed method has errors in the range of a few mV.

7. REFERENCES

[1] Semiconductor Industry Association. International technology roadmap for semiconductors, 2009.

[2] D. Kouroussis and F. N. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *ACM/IEEE Design Automation Conference (DAC'03)*, pages 99–104, Anaheim, CA, June 2-6 2003.

[3] M. Popovich, A. V. Mezhiba, and E. G. Friedman. *Power distribution networks with on-chip decoupling capacitors.* Springer, New York, USA, 2008.

[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*, 21(10):1148–1160, October 2002.

[5] N. H. Abdul Ghani and F. N. Najm. Fast vectorless power grid verification using an approximate inverse technique. In *ACM/IEEE Design Automation Conference (DAC'09)*, pages 184–189, San Francisco, CA, July 26-31 2009.

[6] J. D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem.* Jon Wiley & Sons Ltd., Chichester, UK, 1991.

[7] Y. Saad. *Iterative methods for sparse linear systems.* SIAM, Philadelphia, PA, 2003.

[8] N. H. Abdul Ghani and F. N. Najm. Fast vectorless power grid verification under an RLC model. Submitted to. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

[9] S. Demko, W. F. Moss, and P. W. Smith. Decay rates for inverses of band matrices. *Mathematics of Computation*, 43(168):491–499, October 1984.

[10] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, May 1997.

[11] M. Benzi, C. D. Meyer, and M. Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, September 1996.

[12] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.).* Johns Hopkins University Press, Baltimore, MD, 1996.

[13] J. Zhang. A sparse approximate inverse preconditioner for parallel preconditioning of general sparse matrices. *Applied Mathematics and Computation*, 130(11):63–85, July 2002.

[14] M. Benzi and M. Tuma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2-3):305–340, June 1999.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms and applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[16] G. Sierksma. *Linear and integer programming: theory and practice.* Marcel Dekker, New York, NY, 2001.

[17] A. Frangioni and A. Manca. A computational study of cost reoptimization for min-cost flow problems. *INFORMS Journal on Computing*, 18(1):61–70, 2003.

[18] Mosek: http://www.mosek.com.

[19] S. Lang. *Algebra.* Addison-Wesley, Menlo Park, CA, 1993.

APPENDIX

In the context of section 3.1, we will prove the following claim.

CLAIM 3. *For any integer $p \geq 1$,*

$$D^p K = K - \sum_{k=0}^{p-1} D^k H \tag{58}$$

PROOF. Define the matrix W as:

$$W = (I - D^p)^{-1} \sum_{k=0}^{p-1} D^k H \tag{59}$$

Denoting the set of all eigenvalues of D by $\sigma(D)$, we can write:

$$\max_{\forall \lambda \in \sigma(D)} |\lambda| < 1 \tag{60}$$

From the spectral mapping theorem [19], we know that if k is an integer, we have the following relationship:

$$\sigma(D^k) = \{\lambda^k \quad : \quad \lambda \in \sigma(D)\} \tag{61}$$

Thus, $\rho(D^p) < 1$. The series $\sum_{q=0}^{\infty} X^q$ for a square matrix X is known to converge [7] if and only if $\rho(X) < 1$, under which condition the series limit is $(I - X)^{-1}$. Substituting $(I - D^p)^{-1}$ from (59) by $\sum_{q=0}^{\infty}(D^p)^q$, we obtain:

$$W = \sum_{q=0}^{\infty} (D^p)^q \sum_{k=0}^{p-1} D^k H \tag{62}$$

Expanding the multiplication of the matrix series in (62) and using the fact that the series $\sum_{k=0}^{\infty} D^k$ converges to $(I - D)^{-1}$, we have:

$$W = \sum_{k=0}^{\infty} D^k H = \left(\sum_{k=0}^{\infty} D^k \right) H = (I - D)^{-1} H \tag{63}$$

Since $D = A^{-1}B$ and $B = A - G$, we have $D = A^{-1}(A - G) = I - A^{-1}G$, leading to $I - D = A^{-1}G$. Since K and H are the matrices obtained as the first m columns of G^{-1} and A^{-1}, respectively, we can write (63) as:

$$W = G^{-1}AH = K \tag{64}$$

meaning that:

$$K = (I - D^p)^{-1} \sum_{k=0}^{p-1} D^k H \tag{65}$$

Left multiplying both sides of (65) with $(I - D^p)$, we obtain:

$$(I - D^p) K = \sum_{k=0}^{p-1} D^k H \tag{66}$$

leading to (58), which completes the proof. □