Transient Noise Bounds for Large-scale Power Grids using Vectorless Verification

Naval Gupte, *Student Member*, *IEEE*, and Jia Wang, *Senior Member*, *IEEE*

***Abstract*—Increased demand for low voltage integrated circuits has made power grid analysis extremely critical and indispensable in modern design flows. Efficient validation of on-chip power distribution network is computationally demanding because of increasing grid sizes. Vectorless technique to grid verification estimates worst-case voltage noises without detailed enumeration of load current excitations. We study voltage noise assessment in RLC models of VDD and GND networks in integrated power grids. Abstract grid model is utilized to abbreviate runtime, while transient constraints capture transitory circuit behaviour. Heuristics are employed to extract constraints that restrict power consumption profiles to realistic scenarios. Multiple linear programming problems are formulated to evaluate bounds on voltage overshoots and undershoots. We propose ways to mitigate storage and computational requirements on processing resources, enabling users to deploy computations on economical Cloud Computing platforms. Recommended solution is parallelizable, thereby reducing the overall verification time. Data compression is applied to fully exploit the compute capabilities of contemporary processors for higher throughputs. Experimental results suggest that the proposed technique is practical and scalable for industrial grids.**

***Index Terms*—Vectorless verification, current constraints, steady-state voltage noise, Cloud computing.**

I. Introduction

Power grids (PGs) in integrated circuits (ICs) are designed to provide stable supply and ground references to circuit elements for reliable operation. Voltage violations are mainly attributed to *IR* drops, caused by resistivity of metal rails, and *Ldi/dt* noises, caused by parasitic inductance of rails and grid-package interconnections [1], [2]. With technology scaling, shrinking interconnect sizes lead to larger *IR* drops while higher operating frequencies result in substantial *Ldi/dt* noises. Decreasing supply voltages to reduce power consumption and lowering transistor threshold levels for improved performance lead to diminished noise margins at logic gates. Increasing circuit densities further make modern ICs increasingly susceptible to power supply noises. Poorly designed grids result in excessive fluctuations in voltage levels at underlying circuit elements affecting the overall speed of operation and risking the timing performance of ICs. PG verification refers to ensuring that power supply noises at the grid nodes are within an acceptable range for all possible runtime situations to avoid logic errors and timing violations. Verification should account for both voltage undershoots and overshoots, caused due to inductive components. Due to increasing complexity of modern ICs, PG verification is become highly challenging and extensive work has been done to explore efficient solutions.

Conventional PG verification techniques are based on simulation. PG is modelled as RC/RLC circuits with current sources, representing currents drawn by underlying circuits. Nodal voltages are evaluated using current waveforms, indicative of actual transistor switching. Given the large number of current sources with extensive combinations of current patterns, a comprehensive grid verification is computationally prohibitive. Algorithms to search the current space for worst-case patterns are proposed in [3], [4]. Further, simulation based techniques are predominantly restricted towards the end of IC design cycle as detailed current waveforms need to be extracted from circuit layout to yield discerning results. Early design verification is preferable to shorten grid modification times.

Vectorless schemes estimate worst-case supply noises without enumerating current inputs explicitly. Current excitations are characterized by *current constraints* that specify feasible range in which currents can vary during normal circuit operation. Supply voltage fluctuations are estimated by solving linear programming (LP) problems for worst-case scenarios subject to specified current constraints. Alternately, current constraints could be utilized as design guidelines in a specification-based design flow ensuring grid safety as IC design progresses. PG verification using vectorless approach has been widely investigated [5], [6], [7], [8], [9]. These studies are based on static constraints that do not consider load current transients, resulting in inordinately pessimistic noise predictions. Hierarchical current and power constraints are proposed in [10] for pragmatic bounds. This approach is further improved in [11] using model order reduction. Verification is confined to VDD networks that cannot be extended to integrated grids. Noise assessment under transient constraints in [12] is restricted to the duration for which constraints are specified. In practice, voltage fluctuations for several time slots need to be computed to ensure grid safety under all operating conditions. Verification in [13] computes steady-state solution of the system under static constraints. However, steady-state solution based on transient grid behaviour is desirable.

We consider an integrated RLC grid model with both VDD and GND networks. Verification methodology of [13] is extended to accommodate transient current constraints. To speed up verification, we utilize an abstract grid model that relies on transient constraints to reflect grid dynamics. Computations are re-organized for parallel execution and multiple processing nodes are deployed to reduce overall runtime. Storage and computational requirements of these compute nodes is kept manageable by having them perform redundant but undemanding computations and incorporating data compression mechanisms. For large-scale grids, computing noise bounds accurately is too time consuming. Noise bounds are computed under relaxed set of constraints to identify critical or excessively noisy PG nodes. These nodes are further analysed subject to realistic scenarios for segregating safe nodes from truly unsafe nodes. This paper is an extension of our previous work [14].

Rest of the paper is organized as follows. Section II introduces grid model, steady-state bound computation and current constraints. Our algorithmic solution is developed in Section III, followed by implementation details in Section IV. Experimental evaluation is presented in Section V. Limitations and future improvements are discussed in Section VI, and Section VII concludes our study.

II. Background

A. RLC PG Model

We consider an integrated RLC grid model so that each branch is represented by a resistor, an inductor, a capacitor or a resistor in series with an inductor. Nodes within RL-branches are referred to as *internal nodes*, while all other nodes are referred to as *external nodes*. Ideal voltage sources, representing supply or ground pads, are connected to some of the external nodes. Capacitive branches connect either two external nodes or connect an external node to ground. Some of these nodes have ideal current sources, representing currents drawn from supply network or injected into ground network.

Applying Kirchoff's current law at every node provides time-domain equations describing the circuit. For a grid with N nodes, system equation in terms of nodal voltage noises is expressed as

$$\boldsymbol{G}V(t) + \boldsymbol{C}\dot{V}(t) - \boldsymbol{M}I(t) = I_s(t) \tag{1}$$

where $V(t)$ is a $N\times 1$ vector representing voltage noises. $V_k(t)$ is the noise voltage at node k, related to nodal voltage $U_k(t)$ as

$$V_k(t) = \begin{cases} V_{dd} - U_k(t), & \text{when } k \text{ is a supply node,} \\ \quad\;\, - U_k(t), & \text{when } k \text{ is a ground node.} \end{cases} \tag{2}$$

I_s is a $N\times 1$ vector of outgoing currents, with elements

$$I_{s,k}(t) = \begin{cases} \widehat{I}_{s,k}(t), & \text{when } k \text{ is a supply node,} \\ -\widehat{I}_{s,k}(t), & \text{when } k \text{ is a ground node.} \end{cases} \tag{3}$$

where $\widehat{I}_{s,k}(t)$ is current source at k^{th} node, $k = 1, \ldots, N$. $\boldsymbol{G}$ is a $N\times N$ conductance matrix, $\boldsymbol{C}$ is a $N\times N$ capacitance matrix with 0 entries for internal and external nodes with no capacitances connected, and $\boldsymbol{M}$ is a $N\times\tilde{N}$ incidence matrix with elements ± 1 or 0 for a grid with $\tilde{N}$ inductors. I is a $\tilde{N}\times 1$ vector of inductive branch currents. Inductive branch currents are related to respective voltage drops across inductors as

$$\boldsymbol{M}^T V(t) + \boldsymbol{L}\dot{I}(t) = 0 \tag{4}$$

where $\boldsymbol{L}$ is a $\tilde{N}\times\tilde{N}$ diagonal matrix of inductor values and $\boldsymbol{M}^T$ is the transpose of incidence matrix. Using Backward Euler finite difference approximation ($\dot{V}(t) \approx \frac{V(t)-V(t-\Delta t)}{\Delta t}$), discrete versions for (1) and (4) are derived. Eliminating inductive branch current variables from these discrete-time equations lead to

$$V(t) = \boldsymbol{D}^{-1}\boldsymbol{E}V(t - \Delta t) + \boldsymbol{D}^{-1}I_s(t) \tag{5}$$

where matrices $\boldsymbol{D}$ and $\boldsymbol{E}$ are defined as

$$\begin{gathered} \boldsymbol{D} = \left(\boldsymbol{G} + \tfrac{\boldsymbol{C}}{\Delta t} + \boldsymbol{M}\left(\tfrac{\boldsymbol{L}}{\Delta t}\right)^{-1}\boldsymbol{M}^T\right) \\ \boldsymbol{E} = \left(\tfrac{\boldsymbol{C}}{\Delta t} + \boldsymbol{M}\widehat{\boldsymbol{G}}\right) \end{gathered} \tag{6}$$

$\boldsymbol{D}$ is a sparse symmetric positive-definite M-matrix and $\boldsymbol{E}$ is a sparse matrix with zero columns and rows. $\widehat{\boldsymbol{G}}$ is a $\tilde{N}\times N$ matrix that correspond to $\tilde{N}$ internal nodes and consists of rows either from $-\boldsymbol{G}$ or $\boldsymbol{G}$ depending on current assignment in every RL-branch. $\boldsymbol{D}$ is invertible and $\boldsymbol{D}^{-1}$ is symmetric.

B. Steady-State Upper and Lower Bounds

Steady-state noise bounds are derived in [13] and are briefly discussed next. With no stimulus $\forall t \leq 0$, we have $V(0) = 0$. Evaluating (5) at instants $t = \Delta t, 2\Delta t, \ldots, p\Delta t$ yields

$$V(p\Delta t) = \sum_{k=0}^{p-1} (\boldsymbol{D}^{-1}\boldsymbol{E})^k \boldsymbol{D}^{-1} I_s((p-k)\Delta t) \tag{7}$$

Current constraints characterize grid loading conditions and restrict input currents I_s to a feasible region $\mathfrak{F}$. Current region $\mathfrak{F}$ is same for each time step and the required steady-state solution can be obtained by evaluating the grid at $p \to \infty$ as

$$V_{opt}(\infty) = \lim_{p\to\infty} \sum_{k=0}^{p-1} \underset{I_s(t)\in\mathfrak{F}}{opt} \left[(\boldsymbol{D}^{-1}\boldsymbol{E})^k \boldsymbol{D}^{-1} I_s\right] \tag{8}$$

where *opt* refers to node-wise *maximization* and *minimization* for worst-case voltage overshoots and undershoots, respectively. Further, grouping every r consecutive terms in (7) together leads to

$$V(t) = \boldsymbol{N}V(t - r\Delta t) + S \tag{9}$$

$\forall t = r\Delta t, 2r\Delta t, \ldots, p\Delta t$, where $\boldsymbol{N}$ and S are defined as

$$\begin{gathered} \boldsymbol{N} = (\boldsymbol{D}^{-1}\boldsymbol{E})^r \\ S = \textstyle\sum_{q=0}^{r-1} (\boldsymbol{D}^{-1}\boldsymbol{E})^q \boldsymbol{D}^{-1} I_s \end{gathered} \tag{10}$$

Optimizations at r consecutive time steps are carried out and utilized to yield steady-state bounds at $p \to \infty$. Provided upper and lower noise bounds at time $(t - r\Delta t)$, noise bounds at time t can be computed using

$$Y(t) = \boldsymbol{R}Y(t - r\Delta t) + W \tag{11}$$

where $Y(t) = [V_{ub}(t)\ V_{lb}(t)]^T$, $W = [S^{(r)}_{max}\ S^{(r)}_{min}]^T$ and $\boldsymbol{R}$ is a $2N\times 2N$ matrix defined as

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{N}^+ & \boldsymbol{N}^- \\ \boldsymbol{N}^- & \boldsymbol{N}^+ \end{bmatrix} \tag{12}$$

$\boldsymbol{N}^+$ is a matrix of non-negative values in $\boldsymbol{N}$, $\boldsymbol{N}^-$ is a matrix of non-positive values in $\boldsymbol{N}$ and

$$\begin{bmatrix} S^{(r)}_{max} \\ S^{(r)}_{min} \end{bmatrix} = \begin{bmatrix} \sum_{q=0}^{r-1} \underset{I_s(t)\in\mathfrak{F}}{max} \left[(\boldsymbol{D}^{-1}\boldsymbol{E})^q \boldsymbol{D}^{-1} I_s\right] \\ \sum_{q=0}^{r-1} \underset{I_s(t)\in\mathfrak{F}}{min} \left[(\boldsymbol{D}^{-1}\boldsymbol{E})^q \boldsymbol{D}^{-1} I_s\right] \end{bmatrix} \tag{13}$$

Noise bounds at t are obtained as $V_{lb}(t) \leq V(t) \leq V_{ub}(t)$, $\forall t = r\Delta t, 2r\Delta t, \ldots, p\Delta t$. Evaluating (11) at $t = r\Delta t, 2r\Delta t, \ldots,$

$p\Delta t$ results in

$$Y(p\Delta t) = (\boldsymbol{R}^{p-1} + \ldots + \boldsymbol{R} + I)W \tag{14}$$

Convergence of noise bounds as $p \to \infty$ depend on convergence of matrix series ($\boldsymbol{R}^p$ + ...+ $\boldsymbol{R}$ + I), that is governed by *spectral radius* of matrix $\boldsymbol{N}$. Since $\boldsymbol{N}$ = $(\boldsymbol{D}^{-1}\boldsymbol{E})^r$, value of r for which convergence condition is satisfied is expressed as

$$r = r_0 \in \mathbb{N} : max(\| (\boldsymbol{D}^{-1}\boldsymbol{E})^{r_0} \|_\infty, \| (\boldsymbol{D}^{-1}\boldsymbol{E})^{r_0} \|_1) < 1 \tag{15}$$

where $\mathbb{N}$ is the set of integers. At convergence ($\boldsymbol{R}^p$ + ...+ $\boldsymbol{R}$ + I) $\to$ $(\boldsymbol{I} - \boldsymbol{R})^{-1}$ as $p \to \infty$ and steady-state bounds are computed as

$$\lim_{p\to\infty} Y(p\Delta t) = (\boldsymbol{I} - \boldsymbol{R})^{-1}W \tag{16}$$

$(\boldsymbol{I} - \boldsymbol{R})$ is non-singular and $\lim_{p\to\infty} \boldsymbol{R}^k = 0$. As a consequence, $(\boldsymbol{I} - \boldsymbol{R})^{-1}$ is close to $(\boldsymbol{I})^{-1} = \boldsymbol{I}$ and error is of the same order as $\|\boldsymbol{R}\|_p$ [16].

C. Current Constraints

Current waveforms can be modelled by four types of constraints: *local*, *global*, *equality* and *transient*. Local constraints define lower and upper bounds on individual current sources. Global constraints are introduced to limit currents drawn by groups of current sources that represent independent circuit blocks. Equality constraints ensure currents flowing out of the supply network equal currents flowing into the ground network. Transient constraints restrict total amount of current that each source can draw within an interval. Collectively, these constraints can be expressed as

$$\begin{array}{ll} Local & I_L^{lb} \le I_s(t) \le I_L^{ub} \\ Global & I_G^{lb} \le \boldsymbol{U}I_s(t) \le I_G^{ub} \\ Equality & \boldsymbol{E}I_s(t) = 0 \\ Transient & I_T^{lb} \le \sum_{q=1}^{N_{ts}} I_s(q\delta t) \le I_T^{ub} \end{array} \tag{17}$$

I_L^{lb} and I_L^{ub} are $N\times 1$ lower and upper bound vectors on current sources. For $\widehat{N}$ circuit blocks, $\boldsymbol{U}$ is a $\widehat{N}\times N$ 0/1 matrix indicating assignment of current sources to circuit blocks while $\boldsymbol{E}$ is a $\frac{N}{2}\times N$ matrix of ± 1 or 0 that combines sources in supply and ground networks. I_G^{lb} and I_G^{ub} are $\widehat{N}\times 1$ lower and upper global bound vectors, respectively. I_T^{lb} and I_T^{ub} are $N\times 1$ transient bound vectors that restrict currents during an interval $N_{ts}\delta t$. *Max delta constraints*, introduced in [17] bound the change in current between successive time instants, while [18] utilizes *current slope constraints* to bound minimum current transition time. Combination of several such constraint settings can be employed to improve the feasible current characteristics for vectorless grid verification.

III. Proposed Solution

A. Transient PG Model

Current constraints can be classified broadly into temporal and spatial constraints. Transient constraints represent temporal constraints, while local, global and equality constraints form spatial constraints. Discretizing each time-step Δt further into N_{ts} instants, result in vectors

$$I_s^0 = \begin{bmatrix} I_{s,1}(\delta t) \\ \vdots \\ I_{s,N}(\delta t) \end{bmatrix}, \quad \ldots, \quad I_s^{N_{ts}} = \begin{bmatrix} I_{s,1}(N_{ts}\delta t) \\ \vdots \\ I_{s,N}(N_{ts}\delta t) \end{bmatrix} \tag{18}$$

such that Δt = $N_{ts}\delta t$. Currents associated with each time instant δt apart, are treated as distinct source variables when formulating problems with temporal constraints [19]. This results in $N\times N_{ts}$ distinct current variables in the transient grid model. Current vector I_s can be expressed as

$$I_s = \begin{bmatrix} I_s^0 & I_s^1 & \ldots & I_s^{N_{ts}} \end{bmatrix}^T \tag{19}$$

Constraints on current variables of (19), in terms of transient constraints on current sources, can be written as

$$\begin{array}{ccccc} I_{T,1}^{lb} & \le & I_{s,1}(\delta t) + \ldots + I_{s,1}(N_{ts}\delta t) & \le & I_{T,1}^{ub} \\ \vdots & & \vdots & & \vdots \\ I_{T,N}^{lb} & \le & I_{s,N}(\delta t) + \ldots + I_{s,N}(N_{ts}\delta t) & \le & I_{T,N}^{ub} \end{array} \tag{20}$$

Global and equality constraints for transient grid variables can be expressed as

$$\begin{array}{ll} Global & I_G^{lb} \le \boldsymbol{U}I_s^0 \le I_G^{ub}, \ldots, I_G^{lb} \le \boldsymbol{U}I_s^{N_{ts}} \le I_G^{ub} \\ Equality & \boldsymbol{E}I_s^0 = 0, \ldots, \boldsymbol{E}I_s^{N_{ts}} = 0 \end{array} \tag{21}$$

Local bounds on elements of (19) can be expressed, in terms of bounds on current sources, as

$$\begin{array}{ccc} I_{L,1}^{lb} \le I_{s,1}(\delta t) \le I_{L,1}^{ub}, & \ldots, & I_{L,1}^{lb} \le I_{s,1}(N_{ts}\delta t) \le I_{L,1}^{ub} \\ \vdots & & \vdots \\ I_{L,N}^{lb} \le I_{s,N}(\delta t) \le I_{L,N}^{ub}, & \ldots, & I_{L,N}^{lb} \le I_{s,N}(N_{ts}\delta t) \le I_{L,N}^{ub} \end{array} \tag{22}$$

Constraints for the transient model can be combined as

$$\begin{bmatrix} I_T^{lb} \\ I_{GE}^{lb} \\ \vdots \\ I_{GE}^{lb} \end{bmatrix} \le \begin{bmatrix} \boldsymbol{I} & \ldots & \boldsymbol{I} \\ \boldsymbol{C}_{GE} & & \\ & \ddots & \\ & & \boldsymbol{C}_{GE} \end{bmatrix} I_s \le \begin{bmatrix} I_T^{ub} \\ I_{GE}^{ub} \\ \vdots \\ I_{GE}^{ub} \end{bmatrix} \tag{23}$$

where $\boldsymbol{C}_{GE}$ = $[\boldsymbol{U}\ \boldsymbol{E}]^T$, I_{GE}^{lb} = $[I_G^{lb}\ 0]^T$, I_{GE}^{ub} = $[I_G^{ub}\ 0]^T$ and $\boldsymbol{I}$ is a $N\times N$ identity matrix. Bounds for each of the current variables are static and do not change with time. Verification problem can be formulated as finding a solution to (11) so that

$$\begin{bmatrix} S_{max}^{(r)} \\ S_{min}^{(r)} \end{bmatrix} = \begin{bmatrix} \sum_{q=0}^{r-1} \underset{I_s(t)\in\mathfrak{F}}{max} \left[(\boldsymbol{D}^{-1}\boldsymbol{E})^q \boldsymbol{D}^{-1} \left(I_s^0 + \ldots + I_s^{N_{ts}}\right)\right] \\ \sum_{q=0}^{r-1} \underset{I_s(t)\in\mathfrak{F}}{min} \left[(\boldsymbol{D}^{-1}\boldsymbol{E})^q \boldsymbol{D}^{-1} \left(I_s^0 + \ldots + I_s^{N_{ts}}\right)\right] \end{bmatrix} \tag{24}$$

subject to constraints in (23) and variable bounds in (22). This formulation results in N LP problems in $N\times N_{ts}$ variables subject to N + $[\widehat{N}$ + $\frac{N}{2}]\times N_{ts}$ constraints as depicted in Fig. 1. Current constraints are derived based on the availability of pulse characteristics of current sources and is discussed in [14]. When these characteristics are not available, PG analysis encompassing gate/cell switching activities can be employed to procure these data [20], [21], [22].

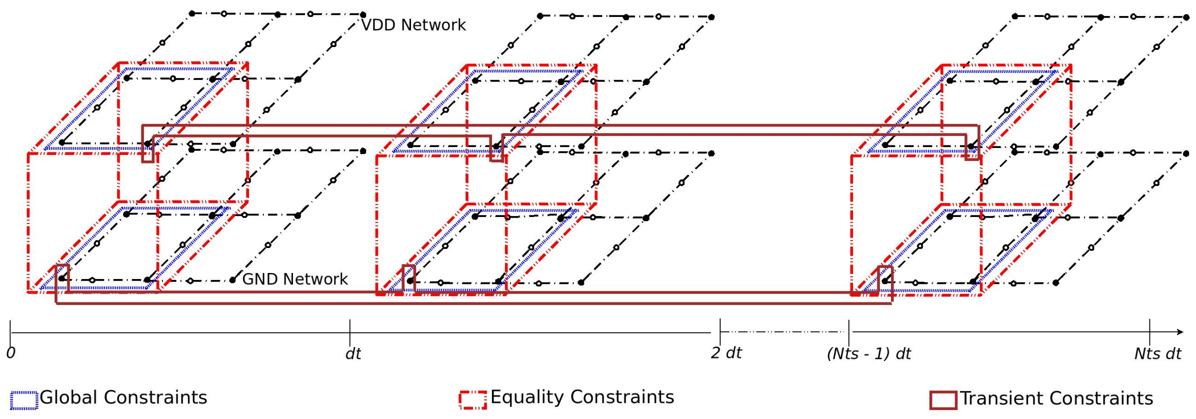

Fig. 1. Current Constraints for Transient PG Model

B. Model Abstraction

Convergence for $(\boldsymbol{R}^p + \ldots + \boldsymbol{R} + I) \rightarrow (\boldsymbol{I} - \boldsymbol{R})^{-1}$ for $p \rightarrow \infty$ is regulated by the spectral radius of $(\boldsymbol{D}^{-1}\boldsymbol{E})$. Therefore, in addition to the RLC component values within the grid, choice of time step Δt impacts the rate of convergence of the verification algorithm. Small values for Δt lead to relatively large number of iterations to satisfy convergence criterion of (15). This is in accordance to the observation in [15], that convergence rate for realistic problems can be extremely small. Therefore, a higher value of Δt is desirable to accelerate convergence. Further, worst-case noise estimates also depend on the choice of Δt. Currents are allowed to change arbitrarily in the feasible region $\mathfrak{F}$ within this duration. Currents can switch from their minimum to their maximum values or vice-versa in a single time step, leading to overly pessimistic bounds for a small Δt. Too small Δt values might introduce round-off errors in computations and render them useless. However, current switching activities are dictated by the underlying logic circuits. Therefore, Δt should be small enough to capture true transient behaviour of the node voltages for maximum switching activities.

To accommodate these conflicting requirements, an abstract grid model is derived by choosing $\Delta t \geq 10ns$. Clearly, this value for Δt is much larger than the fastest switching activities observed in modern ICs. However, by choosing N_{ts} appropriately, close to actual grid dynamics can be captured. Transitory grid behaviour within each Δt duration is captured via transient current constraints. Since $\delta t = \Delta t / N_{ts}$, current sources are effectively restricted to switch between their minimum and maximum bounds within a duration of δt, while concurrently maintaining their dynamic power consumption within each time step Δt. Global constraints are also defined so as to limit power consumption exhibited by groups of current sources, during the period Δt, to realistic scenarios. This ensures faster solution convergence while providing reasonable transient bounds on voltage noises.

IV. Implementation Details

A. Computational Strategies

Our algorithm takes SPICE description of a grid as input and outputs bounds on maximum overshoots and minimum undershoots, at every grid node. Matrices $\boldsymbol{D}$ and $\boldsymbol{E}$ are constructed, followed by a constraint generation routine. $\boldsymbol{D}^{-1}$ needs to be computed and stored for further computations. However, $\boldsymbol{D}^{-1}$ may not be sparse, mandating high storage requirements on the computing machine. For a practical verification framework, proposed scheme must be parallelizable, while minimizing storage and communication overheads associated with deploying multiple compute nodes. To address these design objectives, n computing nodes (CN) are instantiated, where each CN may be a processor core. Each CN has its copy of sparse matrices $\boldsymbol{D}$ and $\boldsymbol{E}$ along with the derived constraints, and is responsible for computations pertaining to $k = \lceil N/n \rceil$ PG nodes. Every CN performs LU factorization of $\boldsymbol{D}$ and computes k column vectors of $\boldsymbol{D}^{-1}$, expressed as

$$\boldsymbol{D}\boldsymbol{D}^{-1}_{(k)} = (\boldsymbol{L}_D\boldsymbol{U}_D)\boldsymbol{D}^{-1}_{(k)} = \boldsymbol{I}_{(k)} \tag{25}$$

where $\boldsymbol{L}_D$, $\boldsymbol{U}_D$ are the LU factors of $\boldsymbol{D}$ and $\boldsymbol{D}^{-1}_{(k)}$, $\boldsymbol{I}_{(k)}$ represent k vectors of $\boldsymbol{D}^{-1}$ and identity matrix $\boldsymbol{I}$, respectively. Since $\boldsymbol{D}^{-1}$ is symmetric, k optimization operations can be initiated in parallel as

$$\begin{aligned} W_{0,max(k)} &= \max_{I_s(t)\in\mathfrak{F}} (\boldsymbol{D}^{-1})^T_{(k)} \boldsymbol{I}_s \\ W_{0,min(k)} &= \min_{I_s(t)\in\mathfrak{F}} (\boldsymbol{D}^{-1})^T_{(k)} \boldsymbol{I}_s \end{aligned} \tag{26}$$

where $(\boldsymbol{D}^{-1})^T_{(k)}$ depict k rows of $\boldsymbol{D}^{-1}$. Column vectors of $\boldsymbol{D}^{-1}$ are discarded after handing them over to the optimization process. Computation of $\boldsymbol{N} = \boldsymbol{D}^{-1}\boldsymbol{E}$ is executed as

$$(\boldsymbol{L}_D\boldsymbol{U}_D)\boldsymbol{N}_{(k)} = \boldsymbol{E}_{(k)} \tag{27}$$

where $\boldsymbol{N}_{(k)}$ and $\boldsymbol{E}_{(k)}$ correspond to k column vectors of $\boldsymbol{N}$ and $\boldsymbol{E}$, respectively. Vectors $\boldsymbol{N}_{(k)}$ are retained by CNs in file system memory for subsequent processing. Following computations are repeated until convergence. Computation of $\boldsymbol{P} = \boldsymbol{N}^r\boldsymbol{D}^{-1}$ proceeds as follows

$$\boldsymbol{D}^T\boldsymbol{P}^T_{(k)} = (\boldsymbol{N}^r)^T_{(k)} \Rightarrow (\boldsymbol{L}_D\boldsymbol{U}_D)\boldsymbol{P}^T_{(k)} = (\boldsymbol{N}^r)^T_{(k)} \tag{28}$$

where $(\boldsymbol{N}^r)^T_{(k)}$ represent k rows of $\boldsymbol{N}^r$ expressed as column vectors. Row-wise data for $\boldsymbol{N}^r$ can be collected by requesting data concerning k rows from other processing nodes. Alternately, network file system can be employed so that every CN has access to file memory of other CNs. LP solves initiated by CNs in each iteration are given by

$$\begin{aligned} W_{r,max(k)} &= \max_{I_s(t)\in\mathfrak{F}} \boldsymbol{P}^T_{(k)}\boldsymbol{I}_s \\ W_{r,min(k)} &= \min_{I_s(t)\in\mathfrak{F}} \boldsymbol{P}^T_{(k)}\boldsymbol{I}_s \end{aligned} \tag{29}$$

where $\boldsymbol{P}^T_{(k)}$ denote k rows of $\boldsymbol{P}$. Essentially, (28) computes k rows of $\boldsymbol{P}$, necessitated by the optimization engine for computing (29). $\boldsymbol{N}^{r+1}$ is computed as

$$\boldsymbol{D}\boldsymbol{N}^{r+1} = \boldsymbol{E}\times\boldsymbol{N}^{r} \Rightarrow (\boldsymbol{L}_D\boldsymbol{U}_D)\boldsymbol{N}^{r+1}_{(k)} = \boldsymbol{E}\boldsymbol{N}^{r}_{(k)} \tag{30}$$

Since $\boldsymbol{E}$ is sparse, computing $\boldsymbol{E}\boldsymbol{N}^{r}_{(k)}$ is relatively inexpensive. Vectors $\boldsymbol{N}^{r+1}_{(k)}$ replace the previous vectors in the file memory, thereby confining overall storage requirements for CNs to k column vectors of lengths N and additional vectors for (26) or (29) of lengths k. Row vectors of $(\boldsymbol{N}^r)^T_{(k)}$ are stored for faster access. $S^{(r)}_{max}$ and $S^{(r)}_{min}$ are computed on convergence. As a final step, $\boldsymbol{R}$ needs to be constructed and (16) solved to obtain the steady-state noise bounds. However, $(\boldsymbol{I} - \boldsymbol{R})$ is a $2N\times 2N$ matrix severely inflating the memory requirements and stipulating much larger computational runtime for matrix decomposition. (11) can be revised as

$$\begin{aligned} V_{ub}(t) &= \boldsymbol{N}^{+}V_{ub}(t - r\Delta t) + \boldsymbol{N}^{-}V_{lb}(t - r\Delta t) + S^{(r)}_{max} \\ V_{lb}(t) &= \boldsymbol{N}^{-}V_{ub}(t - r\Delta t) + \boldsymbol{N}^{+}V_{lb}(t - r\Delta t) + S^{(r)}_{min} \end{aligned} \tag{31}$$

Adding and subtracting the two equations results in

$$\begin{aligned} V_{sum}(t) &= \boldsymbol{N}V_{sum}(t - r\Delta t) + W_{sum} \\ V_{dif}(t) &= \boldsymbol{N}_{abs}V_{dif}(t - r\Delta t) + W_{dif} \end{aligned} \tag{32}$$

where $V_{sum}(t)$ = $[V_{ub}(t) + V_{lb}(t)]$, $V_{dif}(t)$ = $[V_{ub}(t) - V_{lb}(t)]$, W_{sum} = $[S^{(r)}_{max} + S^{(r)}_{min}]$ and W_{dif} = $[S^{(r)}_{max} - S^{(r)}_{min}]$. Matrix summation $(\boldsymbol{N}^{+} + \boldsymbol{N}^{-})$ yields the matrix $\boldsymbol{N}$, while difference $(\boldsymbol{N}^{+} - \boldsymbol{N}^{-})$ leads to a matrix that consists of element-wise absolute values in $\boldsymbol{N}$, denoted by $\boldsymbol{N}_{abs}$. (32) is analogous to (11), and (16) can be accordingly solved as

$$\begin{aligned} V_{sum}(\infty) &= (\boldsymbol{I} - \boldsymbol{N})^{-1}W_{sum} \\ V_{dif}(\infty) &= (\boldsymbol{I} - \boldsymbol{N}_{abs})^{-1}W_{dif} \end{aligned} \tag{33}$$

Eqn. (33) facilitates solution for two $N\times N$ system and its implications on computational expense will become clear when discussing experimental results. The required steady-state bounds are computed as

$$\begin{aligned} V_{ub}(\infty) &= [V_{sum}(\infty) + V_{dif}(\infty)]/2 \\ V_{lb}(\infty) &= [V_{sum}(\infty) - V_{dif}(\infty)]/2 \end{aligned} \tag{34}$$

Fig. 2 shows control and data flow graph for the outlined procedure.

B. Verification Methodology

Estimation of voltage noise bounds on any node requires solutions to two LP problems for all other grid nodes at every previous time step. This leads to prohibitively prolonged verification runtimes. To mitigate this challenge, bounds on voltage noises are estimated using relaxed set of static constraints and variable bounds. Noise bounds are derived subject to global constraints while dropping the equality constraints i.e. $2N$ LP problems in $N\times N_{ts}$ variables are solved subject to $\widehat{N}\times N_{ts}$ constraints. This has little effect on the accuracy of the solution when compared to bounds computed under global and equality constraints. Under global constraints alone, each term in (24) constitutes an identical sub-problem solved under synonymous set of constraints. Therefore it is adequate to solve one sub-problem and scale the disturbance vectors W by factor of N_{ts}. We refer to solutions corresponding to each

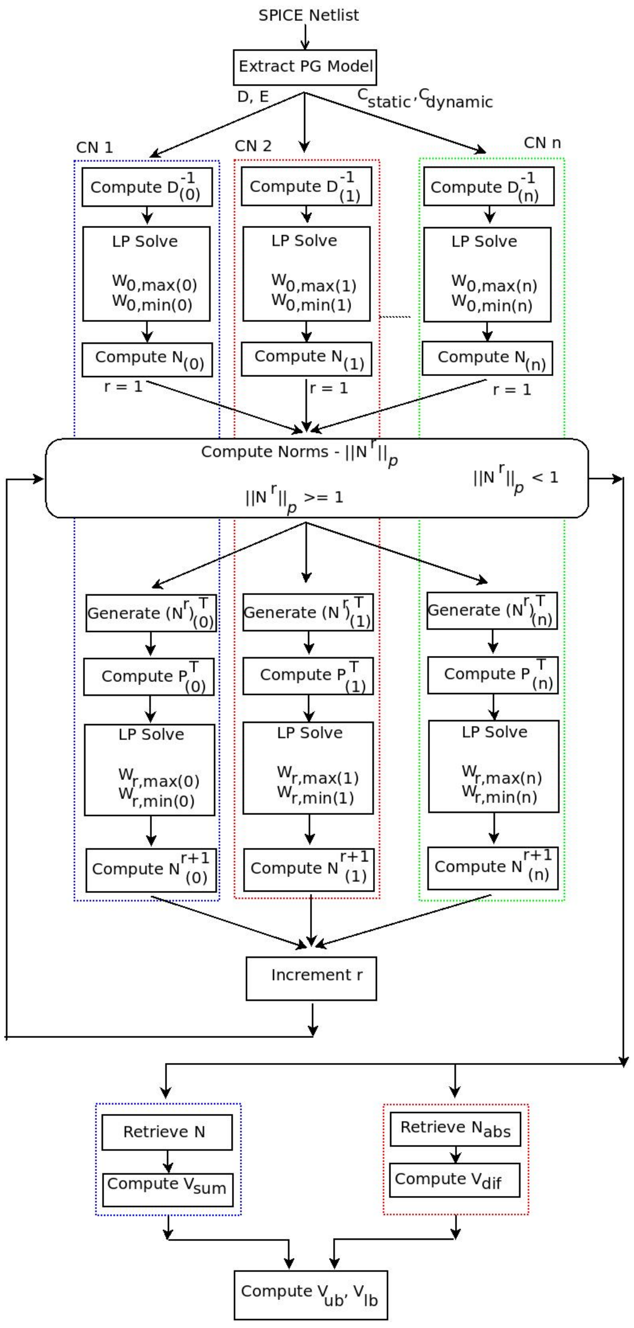

Fig. 2. Control & Data Flow Graph

time step, (26) and (29), as W^*. Nodes that exhibit excessive overshoots or undershoots are identified as critical nodes. For these nodes, elements of (26) and (29) are re-computed under transient constraints and updated in vectors W^*. Updated vectors are denoted by $\tilde{W}$. Steady-state noise bounds are re-calculated. Thus, false-positives can be determined and tighter noise bounds obtained incrementally.

Overall verification procedure is summarized by Algorithm 1. A master process computes the system matrices $\boldsymbol{D}$, $\boldsymbol{E}$, and the grid constraints. Several slave processes are instantiated to handle a fixed number of PG nodes, iteratively. Slaves address optimization problems under global constraints for these nodes. Matrix norms are computed at each iteration. Master consolidates the norms and terminates iterations when convergence criterion is met. Noise bounds under global constraints are determined. Nodes exhibiting excessive voltage fluctuations are identified and the outlined procedure is repeated for these nodes under transient constraints.

Algorithm 1

Input: SPICE Description of PG
Output: Transient Bounds on Critical Nodes of PG

```
 1: Construct $\boldsymbol{D}$, $\boldsymbol{E}$
 2: Extract Constraints - $\boldsymbol{C}_{global}$, $\boldsymbol{C}_{transient}$
 3: Instantiate $n$ Processes
 4:     a. Compute $\boldsymbol{D}^{-1}_{(k)}$, $\boldsymbol{N}_{(k)}$
 5:     b. Initiate Optimization($(\boldsymbol{D}^{-1})^T_{(k)}$, $\boldsymbol{C}_{global}$)
 6: Set $r$ = 1
 7: while $min(\| \boldsymbol{N}^r \|_\infty, \| \boldsymbol{N}^r \|_1) \geq 1$ do
 8:     Instantiate $n$ Processes
 9:         a. Read/Request $\boldsymbol{N}^T_{(k)}$
10:         b. Compute $\boldsymbol{P}^T_{(k)}$, $\boldsymbol{N}^{r+1}_{(k)}$
11:         c. Initiate Optimization($\boldsymbol{P}^T_{(k)}$, $\boldsymbol{C}_{global}$)
12:     Increment $r$
13: end while
14: Compute $W^*_{max}$, $W^*_{min}$, $W^*_{sum}$, $W^*_{dif}$
15: Solve for $V^*_{sum}$, $V^*_{dif}$
16: Compute $V^*_{ub}$, $V^*_{lb}$
17: Determine critical nodes - $N_{critical}$
18: Repeat (4)-(5) & (9)-(11) for $N_{critical}$ s.t. $\boldsymbol{C}_{transient}$
19: Update $W^*_{max} \rightarrow \tilde{W}_{max}$, $W^*_{min} \rightarrow \tilde{W}_{min}$
20: Repeat (14)-(16) for $\tilde{V}_{ub}$, $\tilde{V}_{lb}$
```

C. System Considerations

Modern processors with multiple processing units are referred to as multi-core processors. Additionally, these processors support multi-threading enabling the processors to execute the same program that shares the code and most of the address space. With improved processor performance, memory access becomes a major bottleneck in several applications. Computing systems rely on a hierarchical memory organization to match processor speed with memory bandwidth at the lowest level. Memory hierarchy is organized so that the fastest memory resides at the highest level and the slowest memory occupies the lowest level [23]. Cache modules provide fast access to data and instructions currently in use. Each core has its private L1 data and instruction cache and a larger L2 cache for data and instructions. An even larger *last level cache* L3 is shared across the multiple cores [24]. Cache miss in a higher level initiates request to a lower level. In case of a cache miss at every level, main memory is accessed to fetch the requested block of information. Portions of fetched block is stored in higher cache levels and target bytes are transmitted to the processing unit.

Shared cache resources are dynamically allocated to multiple cores and there is no duplication of shared data. This leads to efficient utilization of cache space. However, data sets of different cores may interfere with each other leading to a poor quality-of-service. In addition, a single large shared cache results in relatively longer access times. To overcome these limitations, shared caches may be distributed physically on the processor die. Cache memories are partitioned into banks that are placed in close proximity of few processor cores. Banked cache organizations support non-uniform access times that are referred to as *non-uniform cache access* (NUCA) architectures. Banks are connected with an on-chip network, and physical network distances and contention affect cache access latencies.

External DRAM memory modules constitute the system's main memory that is inexpensive and large compared to the cache, and fast compared to the disks. Each core has prioritized connection to some local memory modules. Processor cores and memory modules are connected via high-speed interconnects enabling cores to issue requests to its local memory as well as to memories of other cores. However, remote requests are subject to longer wire delays resulting in non-uniform memory access latencies. These are referred to as *non-uniform memory access* (NUMA) architectures. At the lowest level, disks provide permanent storage at lowest cost per bit.

Mismatch between data access patterns of applications and data mapping to memory incur high overheads as remote accesses have higher latency and lower throughput than local accesses [25], [26]. Impact of data placement on performance of multi-core processors with distributed shared caches is emphasized in [27], [28], [29]. Assigning independent working datasets to individual CNs has dual advantages. Data segregation can aid operating system algorithms to jointly place processes and data in same memory domain of shared caches and local external memory modules, thereby mitigating remote accesses and bus contentions. Multiple processor cores in a memory domain accessing the same shared resource, do not compete with other processing elements for data. Each processing unit can modify its data set in private cache regions without triggering cache coherence mechanisms, reducing main memory transactions.

D. Data Compression for Higher Throughputs

Memory demands on each CN is dominated by the storage requirements for upto k columns of $\boldsymbol{N}$. For larger grids even these requirements exceed the capabilities of general computing platforms. For *ibmpg6t* with 1531324 nodes and 20 CNs, this amounts to 470 GB of data per CN for a 4-byte representation. Compression strategies are mandated to handle storage restrictions and aid data transfers among CNs. Additionally, higher throughputs are desirable for rapid verification runtime.

In addition to alleviating the storage requirements, our goal is to improve application performance by lowering network latencies due to main memory and disk memory accesses and effectively harness the on-chip cache resources. To achieve this, data is compressed and decompressed at the application layer on-the-fly. Primary data consists of streams of floating-point column vectors of $\boldsymbol{N}$ that can be processed independently. The application can benefit by reading blocks of compressed column vectors of $\boldsymbol{N}$ for computations, and writing blocks of compressed column vectors after processing. Compression of floating-point data in scientific computing to achieve high compression ratios and throughputs are explored in [33], [34]. We utilize the LZ4 package for our application due to its speed and flexibility to be incorporated into other compression frameworks [35], [36], [37], [38].

Data compression reduces network load, thereby lowering network latencies and improving power consumption. Large shared caches constructed using smaller cache banks, are typically connected through a packet-based communication fabric. Effects of data compression on cache interconnect performance and power consumption are studied in [30], [31], [32]. These proposals focus on compression schemes that compress data either prior to sending it to the shared cache or before injecting it in the external network.

We consider two scenarios in which the application can benefit from data compression -

1) Computation of $\boldsymbol{P}^T$: This computation proceeds in two stages. Row vectors $(\boldsymbol{N}^r)^T$ are assembled by performing disk reads to retrieve all column vectors of $\boldsymbol{N}^r$ that are held in main memory. Compression aids in faster disk reads and permits main memory usage for capturing retrieved data. It also facilitates quicker data transfers between the memory hierarchy and processing cores. Once the required row vectors are extracted, compressed data is written to disk. With availability of $(\boldsymbol{N}^r)^T$, row vectors of $\boldsymbol{P}$ can be readily computed. These are utilized by optimization routines and later discarded. Smaller data footprint may assist in better cache utilization by ensuring that row vectors of $\boldsymbol{N}^r$ are always present at some cache level, thereby avoiding main memory transactions.

2) Computation of $\boldsymbol{N}^{r+1}$: Disk read is performed to recover previously computed columns of $\boldsymbol{N}^r$. Columns are decompressed and multiplied with $\boldsymbol{E}$. Computed vectors of the matrix product $\boldsymbol{EN}^r$ are then utilized to compute vectors corresponding to $\boldsymbol{N}^{r+1}$. Resultant vectors are compressed and can be flushed to low level memories without performance penalty. Data in higher memory hierarchies are flushed out only when resources are exhausted. Disk write is carried out when all column computations are complete. Similar to previous case, low level memory accesses are avoided whenever possible.

The key to effective memory utilization is the size of working set. Applications with compressed datasets can benefit from hierarchical non-uniform memory structures since data can reside closer to processor cores. Memory transactions are not explicitly triggered by the application but are performed under supervision of the operating system. Compression at application layer serves a dual purpose. Applications can execute faster when data is located close to processor cores. It reduces network load enabling lower memory latencies, higher available bandwidth and reduced power consumption.

V. Experimental Results

A. Experimental Setup

The proposed verification algorithm is implemented in C++. LU factorizations are performed using NICSLU [39], Eigen package is employed for matrix multiplications [40], and LP problems are solved by Mosek optimization software [41]. To evaluate the performance of our approach, we utilize IBM PG benchmarks for transient analysis [42]. Experiments are carried out on two 64-bit Linux servers; server one (S1) with two 2.67 GHz Intel X5650 six-core processors with 32 KB L1 data and instruction caches, 256 KB L2 cache, 12 MB L3 cache and 64 GB memory, and server two (S2) with two 2.4 GHz Intel Xeon E5-2620 six-core processors with 32 KB L1 data and instruction caches, 256 KB L2 cache, 15 MB L3 cache and 132 GB memory. However, server S1 is utilized by several users with varying load patterns. To fully utilize the CPU capabilities 12 processes are initiated in parallel. To determine economical feasibility, application is deployed on Amazon Cloud platform and performance is observed on 64-bit m3.xlarge EC2 instances [43]. These are equipped with one 2.5 GHz Intel Xeon E5-2670 two-core processor with 32 KB L1 data and instruction caches, 256 KB L2 cache, 25 MB L3 cache and 15 GB memory. In multiprogrammed environments every process gets to use the processor for a time slice in round-robin fashion until it terminates. To observe the effects of multiprogramming, 4 processes are instantiated on AWS instances. System solution, concerning step 15 of Algorithm 1, mandate larger memory-optimized r3.8xlarge EC2 instances with 244 GB memory. 20 global constraints and 10 equality constraints are specified for each of the test circuits. Time-step for the PG model is set to Δt = 10ns and number of discrete time instants for which transient constraints are specified is set to N_{ts} = 10. In effect, currents are allowed to vary arbitrarily between their minimum and maximum values within a duration of δt = 1ns. This is commensurate to transitions specified for pulse excitations in the transient benchmarks. Experiments are conducted considering n = 20 CNs. Verification time is inversely proportional to the number of CNs which represent either independent or time-multiplexed processes. *ibmpg1t* and *ibmpg2t* are solved using server S1, while S2 is utilized to solve *ibmpg3t* and *ibmpg4t*. m3.xlarge EC2 instances are used to solve LP problems for *ibmpg5t* and *ibmpg6t*. r3.8xlarge EC2 instances are utilized for final system solution for *ibmpg2t* - *ibmpg5t*. Main memory requirements for system solution of *ibmpg6t* exceed RAM resources on largest EC2 instances. Iterative methods to solving linear systems can be employed as these require less memory compared to direct techniques [15]. Alternately, swap files on disk drives can be utilized to hold data temporarily but result in slower memory accesses [46].

B. Performance Evaluation

Convergence criterion of (15) is satisfied for r = 1, limiting the number of matrix operations and LP problems. To verify the effectiveness of our constraint generation procedure, *ibmpg1t* benchmark is verified under transient conditions while ignoring spatial switching profile with α = 1, and then under transient conditions with α = 0.21 as determined by the

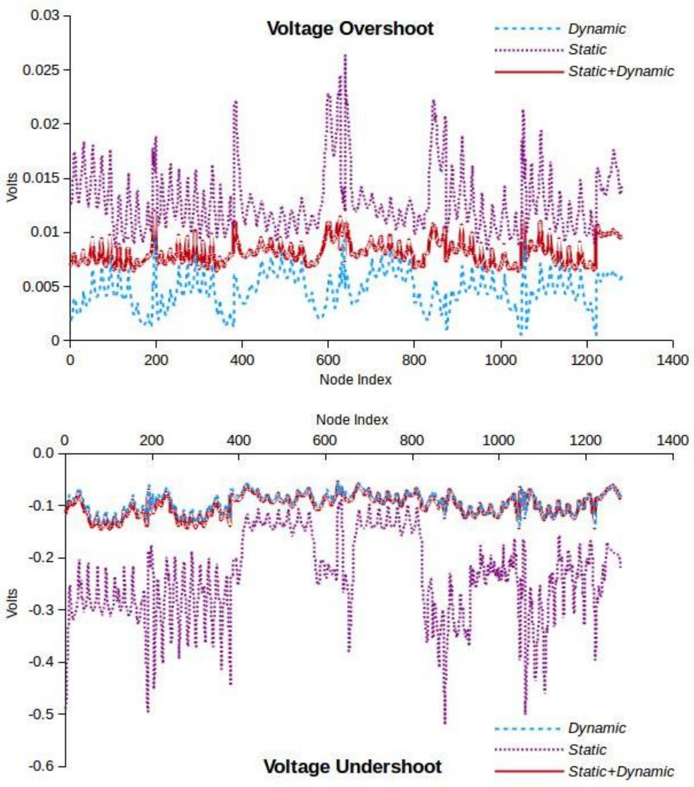

Fig. 3. Static & Dynamic Noise Bounds for *ibmpg1t*

constraint generation procedure. Average worst-case voltage noises reported for $\alpha = 1$ are about 139% higher than that for $\alpha = 0.21$. Likewise, verification under static constraints ignoring the transient current profiles lead to overestimation in bounds. *ibmpg1t* is verified subject to static as well as transient constraints. To assert the potency of verification procedure of Algorithm 1, disturbances in 5% of the PG nodes located in a single neighbourhood are computed under transient conditions. These are combined with results from static optimization for rest of the nodes to compute the overall noise bounds. Fig. 3 depicts the noise estimates for these PG nodes under *Static* (**S**), *Dynamic* (**D**) and *Static+5%Dynamic* (**S**+5%**D**) scenarios. Horizontal axis represents node indices and vertical axis depicts voltage undershoot/overshoot under different constraint settings. It is apparent that the proposed scheme can be utilized for faster noise estimates on larger grids. Table I reports the average voltage noises for 5% arbitrary nodes. *ibmpg1t* is verified using compressed as well as uncompressed datasets. Inter-process communications are improved by a factor of two, while minor gains are observed for other computations. Performance incentives could be higher for larger benchmarks.

Table II provides node counts in each circuit in column 2 while constraint counts are reported in column 3. α values are tabulated in column 4. Runtime breakdown for each CN is tabulated next. Duration for LU factorization is reported in column 5, and column 6 indicates file read times required to get the row vectors of $\boldsymbol{N}$. Multiplication time for computing the right-hand side matrix in (30) is reported in column 7 while

TABLE I. Average Voltage Noises

IBM PG	Overshoot (mV)		Undershoot (mV)	
	S	**S**+5%**D**	**S**	**S**+5%**D**
ibmpg1t	12.8	8.1	-231.3	-101.1
ibmpg2t	7.4	5.7	-289.2	-116.9
ibmpg3t	5.6	4.2	-209.4	-50.6
ibmpg4t	0.3	0.1	-5.1	-0.8
ibmpg5t	4.9	3.0	-62.0	-17.6

duration for system solve is indicated in column 8. Majority of this duration is utilized in assembling matrix $\boldsymbol{N}$ from column vectors $\boldsymbol{N}_{(k)}$ spread across multiple CNs. Static and dynamic optimization durations per node are listed in columns 9 and 10, respectively. Optimization durations are reported taking into account the computational overheads due to data decompression. Static verification time is presented in column 11. Dynamic runtime with 5% nodes re-computed is presented in column 12. Column 13 lists overall storage requirements for each CN and column 14 indicates **S**+5%**D** verification costs when carried out on Amazon Cloud instances. Fig. 4 depicts the noise estimates from static and **S**+5%**D** verification for 1000 arbitrary nodes for two largest IBM benchmarks verified.

Proposed verification methodology exhibits comparable overall performance to transient approach of [12] for like sized circuits. However, their algorithm cannot be extended to compute steady-state noise bounds. Additionally, transient constraints are specified for groups of current sources. Our approach specifies transients for each current source allowing better characterization of current loading effects. Variable reduction approach is employed to generate reduced-size LP problems. User-specified error tolerance value is used to restrict variable removal. However, a single optimization run is employed to estimate voltage noises for the duration of the transients. Our approach requires multiple optimization runs per node. It is, therefore, challenging to distribute user-specified error tolerances across multiple LP problem executions.

Per node optimization duration of our approach is larger than [13]. Increase in runtime can be attributed to inclusion of transient constraints. Matrix operation dominate their algorithm runtime, rendering it infeasible for larger circuits. Proposed scheme re-organizes computations so that complex processing is realized via elementary sparse matrix factorization. Data compression is employed to meet the storage requirements. Compression is equally favourable from cache and memory utilization perspective and for inter-process communication. Cloud deployment costs are significantly lower than initial estimates given in [14].

VI. Limitations & Future Work

Noise estimation under transient constraints is a major bottleneck in the overall verification flow. Runtime can be significantly improved by reducing the dynamic optimization duration. Constraint matrix of (23) has a block-angular structure. Dantzig-Wolfe decomposition can be exploited to accelerate LP problem solution [44]. p distinct solutions to the relaxed sub-problems of (24) are generated using arbitrary

TABLE II. RUNTIME & COMPUTATIONAL REQUIREMENT ANALYSIS

IBM PG	Node Count	Constr. Count	α	Runtime Breakdown (sec) t_{lu}	$t_{tr(N)}$	t_{mm}	t_{sys}	t_{sta}	t_{dyn}	**S**tatic (hrs)	5%**D**yn (hrs)	Storage (MB)	Cost ($) **S**+5%**D**
ibmpg1t	25649	25949	0.21	0.05	60	5	95	0.01	3.5	0.1	0.4	14	1
ibmpg2t	164237	164537	0.11	80	383	64	2700	0.6	35.4	6.0	22.3	303	61
ibmpg3t	1041534	1041834	0.09	45	3600	1800	14400	2.3	76.0	143.7	199.2	2216	540
ibmpg4t	1212364	1212664	0.21	80	12600	3103	25200	3.4	169.6	244.2	387.0	1678	1048
ibmpg5t	1013974	1014274	0.20	4	3600	7455	32400	6.2	287.8	372.6	575.2	1422	1554
ibmpg6t	1531324	1531624	0.22	10	46800	15549	-	12.8	695.0	-	-	3402	≈5207

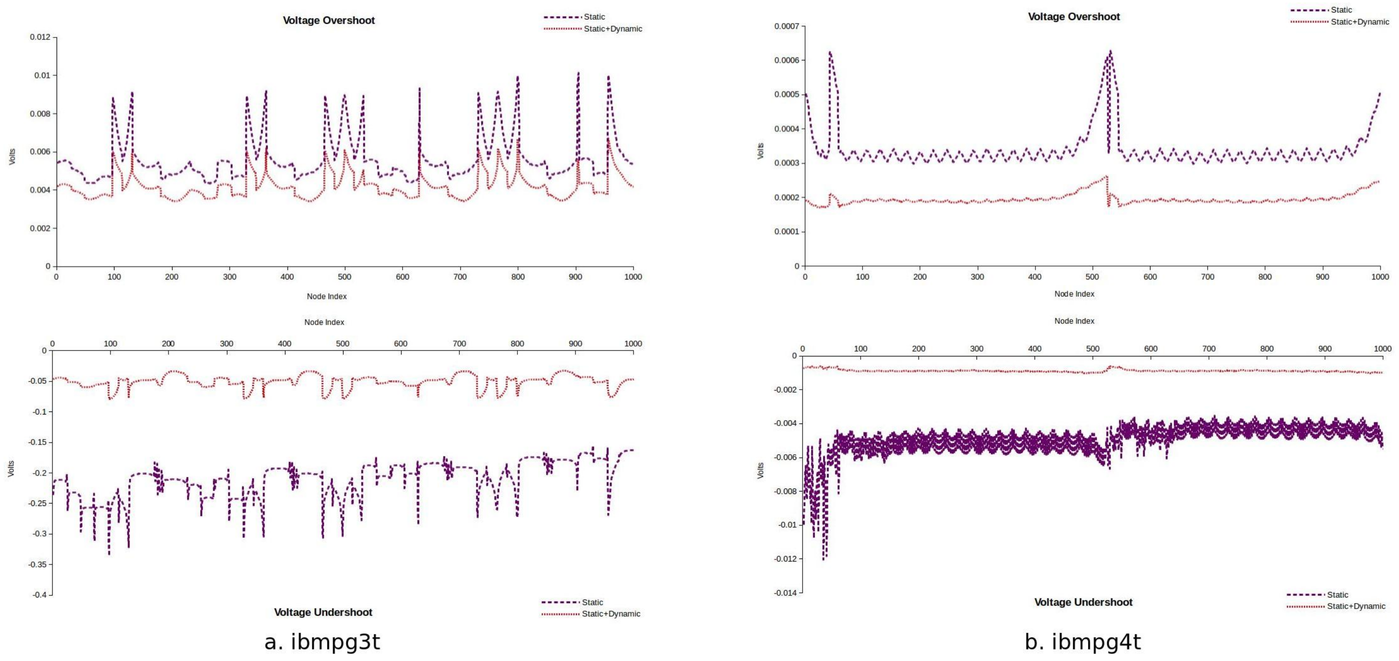

Fig. 4. Transient Noise Bounds

cost coefficients. A master problem, that assigns weights to these solutions optimally, is then formulated. Complicating constraints $I_t^{lb} \le \boldsymbol{II}_s^0 + \ldots + \boldsymbol{II}_s^{N_{ts}} \le I_t^{ub}$ are enforced, ensuring that the solution to the master problem is a solution to the original problem. New objective functions for each sub-problem are formulated and solved. The master problem incorporates the solutions to the modified sub-problems depending on their ability to improve the current objective value. This iterative procedure terminates when current solution can no longer be improved. Efficient mechanisms for generating feasible solutions to sub-problems need to be investigated. Model order reduction may be examined for faster verification [45].

Iterative procedures can be exploited for larger systems as these warrant lower main memory than direct matrix decomposition methods. Model time step Δt is tightly coupled with the transient constraint specification. Ways to define Δt independent of the constraint duration will be explored.

VII. CONCLUSION

Steady-state vectorless approach is extended to estimate bounds on transient voltage noises. Abstract grid model is utilized to speed up convergence while load current dynamics are captured via transient current constraints. Heuristics are presented to capture grid behaviour effectively. Methodologies to parallelize computations and expedite verification by re-using conservative bounds is proposed. Data compression techniques are exploited to meet storage requirements for large-scale grids, that can further aid in effective utilization of memory hierarchies in modern computing systems. Proposed approach is scalable for large circuits. Critical PG nodes are identified and can be subjected to extensive verification.

REFERENCES

[1] A. Muramatsu, M. Hashimoto, H. Onodera, "Effects of on-chip inductance on power distribution grid," in *Proc. Intl. Symp. Physical Design (ISPD)*, 2005, pp. 63-69.

[2] N. Srivastava, X. Qi, K. Banerjee, "Impact of on-chip inductance on power distribution network design for nanometer scale integrated circuits," in *Proc. Intl. Symp. Quality Electronic Design (ISQED)*, 2005, pp. 346-351.

[3] H. Kriplani, F. N. Najm, I. N. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: Algorithms, signal correlation, and their resolution," in *IEEE Trans. Computer-Aided Design*, vol. 14, no. 8, pp. 998-1012, Aug 1995.

[4] S. Pant, D. Blaauw, S. Sundareswaran, R. Panda, ”A stochastic approach to power grid analysis,” in *Proc. Design Automation Conf. (DAC)*, 2004, pp. 171-176.

[5] D. Kouroussis, F. N. Najm, ”A static pattern-independent technique for power grid voltage integrity verification,” in *Proc. Design Automation Conf. (DAC)*, 2003, pp. 99-104.

[6] I. A. Ferzli, F. N. Najm, L. Kruse, ”A geometric approach for early power grid verification using current constraints,” in *Proc. Intl. Conf. Computer-Aided Design (ICCAD)*, 2007, pp. 40-47.

[7] N. H. A. Ghani, F. N. Najm, ”Handling inductance in early power grid verification,” in *Proc. Intl. Conf. Computer-Aided Design (ICCAD)*, 2006, pp. 127-134.

[8] X. Xiong, J. Wang, ”Dual algorithms for vectorless power grid verification under linear current constraints,” in *IEEE Trans. Computer-Aided Design*, vol. 30, no. 10, pp. 1469-1482, Oct 2011.

[9] M. Avci, F. N. Najm, ”Early P/G grid voltage integrity verification,” in *Proc. Intl. Conf. Computer-Aided Design (ICCAD)*, 2010, pp. 816-823.

[10] C. K. Cheng, P. Du, A. B. Kahng, G. K. H. Pang, Y. Wang, N. Wong, ”More realistic power grid verification based on hierarchical current and power constraints,” in *Proc. Intl. Symp. Physical Design (ISPD)*, 2011, pp. 159-166.

[11] Y. Wang, X. Hu, C. K. Cheng, G. K. H. Pang, N. Wong, ”A realistic early-stage power grid verification algorithm based on hierarchical constraints,” in *IEEE Trans. Computer-Aided Design*, vol. 31, no. 1, pp. 109-120, Jan 2012.

[12] X. Xiong, J. Wang, ”Verifying RLC power grids with transient current constraints,” in *IEEE Trans. Computer-Aided Design*, vol. 32, no. 7 pp. 1059-1071, July 2013.

[13] N. H. A. Ghani, F. N. Najm, ”Fast vectorless power grid verification under an RLC model,” in *IEEE Trans. Computer-Aided Design*, vol. 30, no. 5 pp. 691-703, May 2011.

[14] N. Gupte, J. Wang, ”Transient noise bounds using vectorless power grid verification,” in *Proc. Intl. Conf. Computer-Aided Design (ICCAD)*, 2015, pp. 713-720.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003.

[16] G. W. Stewart, *Matrix Algorithms Vol. I: Basic Decompositions*, SIAM, 1998.

[17] I. A. Ferzli, E. Chiprout, F. N. Najm, ”Verification and codesign of the package and die power delivery system using wavelets,” in *IEEE Trans. Computer-Aided Design*, vol. 29, no. 1 pp. 92-102, Jan 2010.

[18] P. Du, X. Hu, S. H. Weng, A. Shayan, X. Chen, A. E. Engin, C. K. Cheng, ”Worst-case noise prediction with non-zero current transition times for early power distribution system verification,” in *Proc. Intl. Symp. Quality Electronic Design (ISQED)*, 2010, pp. 624-631.

[19] D. G. Luenberger, Y. Ye, *Linear and Nonlinear Programming*, 3rd ed., Springer, 2008.

[20] P. M. Morgado, P. F. Flores, L. M. Silveira, ”Generating realistic stimuli for accurate power grid analysis,” in *IEEE Comp. Soc. Annual Symp. VLSI (ISVLSI)*, 2007, pp. 233-238.

[21] P. M. Morgado, P. F. Flores, J. C. Monteiro, L. M. Silveira, ”Generating worst-case stimuli for accurate power grid analysis,” in *Integrated Circuit and System Design-Power and Timing Modelling, Optimization and Simulation*, Springer-Verlag, 2009, pp. 247-257.

[22] H. Qian, S. R. Nassif, S. S. Sapatnekar, ”Early-stage power grid analysis for uncertain working modes,” in *IEEE Trans. Computer-Aided Design*, vol. 24, no. 5, pp. 676-682, May 2005.

[23] K. Hwang, F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, 1984.

[24] R. Balasubramonium, N. P. Jouppi, N. Muralimanohar, *Multi-Core Cache Hierarchies*, Morgan & Claypool, 2011.

[25] Z. Majo, T. R. Gross, ”Memory matching access patterns and data placement for NUMA systems,” in *Proc. Intl. Symp. Code Generation & Optimization (CGO)*, 2012, pp. 230-241.

[26] M. Diener, E. H. M. Cruz, P. O. A. Navaux, ”Locality vs. balance: Exploring data mapping policies on NUMA systems,” in *Euromicro Intl. Conf. Parallel, Distributed and Network-based Processing (PDP)*, 2015, pp. 9-16.

[27] N. Hardavellas, M. Ferdman, B. Falsafi, A. Ailamaki, ”Near-optimal cache block placement with reactive nonuniform cache architectures,” in *IEEE Micro*, vol. 30, no. 1, pp. 20-28, Jan 2010.

[28] G. S. Sachdev, K. Sudan, M. W. Hall, R. Balasubramonium, ”Understanding the behavior of Pthread applications on non-uniform cache architectures,” in *Intl. Conf. Parallel Architectures & Compilation Techniques (PACT)*, 2011, pp. 175-176.

[29] N. Beckmann, Po-Ann Tsai, D. Sanchez, ”Scaling distributed cache hierarchies through computation and data co-scheduling,” in *IEEE Intl. Symp. High Performance Computer Architecture (HPCA)*, 2015, pp. 538-550.

[30] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, ”Performance and power optimization through data compression in Network-on-Chip architectures,” in *IEEE Intl. Symp. High Performance Computer Architecture (HPCA)*, 2008, pp. 215-225.

[31] Y. Jin, K. H. Yum, E. J. Kim, ”Adaptive data compression for high-performance low-power on-chip networks,” in *IEEE/ACM Intl. Symp. Microarchitecture*, 2008, pp. 354-363.

[32] J. Zhan, M. Poremba, Y. Xu, Y. Xie, ”NoΔ: Leveraging delta compression for end-to-end memory access in NoC based multicors,” in *Proc. Asia South Pacific Design Automation Conf (ASP-DAC)*, 2014, pp. 586-591.

[33] P. Ratanwoprabhan, J. Ke, M. Burtscher, ”Fast lossless compression for scientific floating-point data,” in *Proc. Data Compression Conference (DCC)*, 2006, pp. 133-142.

[34] A. Padyana, C. D. Sudheer, P. K. Baruah, A. Srinivasan, ”High throughput compression of floating point numbers on graphical processing units,” in *IEEE Intl. Conf. Parallel Distributed & Grid Computing (PDGC)*, 2012, pp. 313-318.

[35] Y. Collet, *Realtime data compression* [Online]. Available: http://fastcompression.blogspot.in/2011/05/lz4-explained.html

[36] LZ4 [Online]. Available: https://github.com/Cyan4973/lz4

[37] S. Liu, X. Huang, Y. Ni, H. Fu, G. Yang, ”A versatile compression method for floating-point data stream,” in *Intl. Conf. Networking & Distributed Computing (ICNDC)*, 2013, pp. 141-145.

[38] D. Harnik, E. Khaitzin, D. Sotnikov, S. Taharlev, ”A fast implementation of defalte,” in *Proc. Data Compression Conference (DCC)*, 2014, pp. 223-232.

[39] X. Chen, Y. Wang, H. Yang, ”NICSLU: An adaptive sparse matrix solver for parallel circuit simulation”, in *IEEE Trans. Computer-Aided Design*, vol. 32, no. 2, pp. 261-274, Feb 2013.

[40] Eigen [Online]. Available: http://eigen.tuxfamily.org

[41] Mosek [Onine]. Available: http://www.mosek.com

[42] IBM Benchmarks For Transient Analysis [Online]. Available: http://dropzone.tamu.edu/ pli/PGBench/

[43] AWS Documentation [Online].
Available: http://aws.amazon.com/documentation/

[44] A. J. Conejo, E. Castillo, R. Minguez, R. Garcia-Bertrand, *Decomposition Techniques in Mathematical Programming*, Springer-Verlag, 2006.

[45] P. Benner, M. Hinze, E. Jan W. ter Maten, *Model Reduction for Circuit Simulation*, Springer, 2011.

[46] SwapFaq [Online]. Available: https://help.ubuntu.com/community/SwapFaq