VERIFICATION OF LARGE-SCALE ON-CHIP POWER GRIDS

BY
XUANXING XIONG

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering in the Graduate College of the Illinois Institute of Technology

Approved
Advisor

Chicago, Illinois
May 2013
ACKNOWLEDGMENT

First and foremost, I would like to thank my Ph.D. advisor Prof. Jia Wang. Without his guidance and support throughout my graduate study, this dissertation would not have been accomplished. I appreciate all his research directions, technical criticisms, as well as valuable feedbacks from innumerable office discussions. His intelligence, wisdom and diligence are contagious and motivational for me. I will always be in his debt and admire him for the great person he is. I also would like to thank my thesis committee: Prof. Jafar Saniie, Prof. Erdal Oruklu, Prof. Zuyi Li and Prof. Zhiling Lan for their time, interest, suggestions and comments. Last, but not least, I would like to thank my wife Jingjin and our parents for their support at all times, and I hope that I can make them proud.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>x</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xi</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Power Grid Verification</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Contributions</td>
<td>7</td>
</tr>
<tr>
<td>1.3. Thesis Organization</td>
<td>12</td>
</tr>
<tr>
<td>2. PRELIMINARIES</td>
<td>13</td>
</tr>
<tr>
<td>2.1. Power Grid Simulation</td>
<td>13</td>
</tr>
<tr>
<td>2.2. Vectorless Verification</td>
<td>15</td>
</tr>
<tr>
<td>3. PARALLEL TRANSIENT SIMULATION</td>
<td>18</td>
</tr>
<tr>
<td>3.1. Background</td>
<td>18</td>
</tr>
<tr>
<td>3.2. Parallel Forward and Back Substitution</td>
<td>19</td>
</tr>
<tr>
<td>3.3. Transient Simulator – IITPGS</td>
<td>24</td>
</tr>
<tr>
<td>3.4. Experimental Results</td>
<td>26</td>
</tr>
<tr>
<td>4. STEADY-STATE VERIFICATION</td>
<td>32</td>
</tr>
<tr>
<td>4.1. Background</td>
<td>32</td>
</tr>
<tr>
<td>4.2. Dual Bound Algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.3. Hierarchical Matrix Inversion</td>
<td>54</td>
</tr>
<tr>
<td>4.4. Constraint Abstraction</td>
<td>72</td>
</tr>
<tr>
<td>5. TRANSIENT VERIFICATION</td>
<td>86</td>
</tr>
<tr>
<td>5.1. Background</td>
<td>86</td>
</tr>
<tr>
<td>5.2. Vectorless Verification with Transient Current Constraints</td>
<td>89</td>
</tr>
<tr>
<td>5.3. Proposed Methodology</td>
<td>96</td>
</tr>
<tr>
<td>5.4. Variable Reduction</td>
<td>105</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Solution Accuracy of IITPGS (Direct Solver). $E_{\text{max}}$: maximum error in volt; $E_{\text{avg}}$: average error in volt.</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Simulation Runtime of IITPGS (1 thread). Nodes: the number of nodes after merging nodes connected by shorts; DC: DC Analysis; Fact.: factorization; Comp. $b(t)$: compute $b(t)$; FBS: forward and back substitution; nnz: the number of non-zeros; PCG: the PCG method for solving the system equation (2.4); runtime is in seconds.</td>
<td>28</td>
</tr>
<tr>
<td>4.1 Accuracy Settings of Different Algorithms</td>
<td>45</td>
</tr>
<tr>
<td>4.2 Runtime Comparison of DirectVN and DualVN with 4 Global Constraints</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Runtime Comparison of DirectVN and VNBound with 40 Global Constraints</td>
<td>51</td>
</tr>
<tr>
<td>4.4 Runtime Comparison of HierarchicalVN and DualVD with 4 Global Constraints</td>
<td>71</td>
</tr>
<tr>
<td>4.5 Performance Results of the Proposed Constraint Abstraction Approach. $n$: the number of nodes after merging the nodes connected by shorts; $v_{\text{max}}$: the maximum voltage noise across the grid in mV; $rss$: rough subgrid size; $n_0$: the number of global nodes; $E_{\text{max}}/E_{\text{avg}}$: the maximum/average overestimation of voltage noises in mV; the runtime units “s”, “m”, “h” and “d” represent seconds, minutes, hours, and days, respectively.</td>
<td>84</td>
</tr>
<tr>
<td>5.1 Average Runtime per Node for Vectorless Verification</td>
<td>115</td>
</tr>
<tr>
<td>5.2 Worst-case Voltage Noises of a Random Node With and Without Transient Constraints</td>
<td>116</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Part of a typical RC power grid model.</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>A lower triangular matrix $\mathbf{L}$, the associated elimination tree, and</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>the mapping of tasks to two threads.</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>The node ordering algorithm based on nested dissection for stochastic</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>preconditioning. All the nodes can be numbered from 1 to $n$ by a call to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Node Ordering($\mathcal{N}$, 1).</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>An example of the non-zero pattern of matrix $\mathbf{L}$ and the</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>corresponding dependency acyclic graph.</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Simulation flow of IITPGS.</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>Performance scaling of IITPGS (direct solver).</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>Performance scaling of IITPGS (PCG solver).</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>The piecewise linear function $f_k(x)$ and its piecewise linear</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>approximation function $b_k(x)$.</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>The VNBound algorithm.</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Number of variables in the reduced-size LP problem of VNBound for two</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>synthetic power grids pg4000 (90643 nodes) and pg10000 (562363 nodes) with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 and 40 global constraints.</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Cumulative distribution function of overestimation for synthetic power</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>grid pg4000 (90643 nodes) with 40 global constraints, $\text{gap} = 2000$</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>99% overestimation of VNBound for two synthetic power grids pg4000 (90643</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>nodes) and pg10000 (562363 nodes) with 4 and 40 global constraints.</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Speedup of DualVN relative to DirectVN in solving the LP problem.</td>
<td>52</td>
</tr>
<tr>
<td>4.7</td>
<td>Speedup of VNBound relative to DirectVN in solving the LP problem.</td>
<td>53</td>
</tr>
<tr>
<td>4.8</td>
<td>Speedup of VNBound relative to DualVN in solving the LP problem.</td>
<td>54</td>
</tr>
<tr>
<td>4.9</td>
<td>A node $l$ in the power grid.</td>
<td>55</td>
</tr>
<tr>
<td>4.10</td>
<td>A subset of nodes in the power grid.</td>
<td>57</td>
</tr>
<tr>
<td>4.11</td>
<td>Part of a partitioned power grid.</td>
<td>61</td>
</tr>
</tbody>
</table>
5.10 The variable reduction algorithm.  

5.11 Runtime break down of noise optimization using the proposed variable reduction algorithm. “Setup” denotes the runtime of the variable reduction procedure; “Solve” represents the runtime for solving the reduced-size LP problems.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$n \times n$ power grid matrix including R/L/C components</td>
</tr>
<tr>
<td>$c_l$</td>
<td>$n \times 1$ vector for the $l$th non-VDD/GND node</td>
</tr>
<tr>
<td>$C$</td>
<td>$n \times n$ diagonal capacitance matrix</td>
</tr>
<tr>
<td>$G$</td>
<td>$n \times n$ conductance matrix</td>
</tr>
<tr>
<td>$i$</td>
<td>$n \times 1$ current source vector</td>
</tr>
<tr>
<td>$i(t)$</td>
<td>$n \times 1$ time-varying current source vector</td>
</tr>
<tr>
<td>$I_G$</td>
<td>$m \times 1$ upper-bound vector of global constraints</td>
</tr>
<tr>
<td>$I_L$</td>
<td>$n \times 1$ upper-bound vector of local constraints</td>
</tr>
<tr>
<td>$m$</td>
<td>number of global constraints</td>
</tr>
<tr>
<td>$n$</td>
<td>number of non-VDD/GND nodes in the power grid</td>
</tr>
<tr>
<td>$U$</td>
<td>$m \times n$ 0/1 matrix indicating the assignment of current sources to global constraints</td>
</tr>
<tr>
<td>$v$</td>
<td>$n \times 1$ voltage noise vector</td>
</tr>
<tr>
<td>$v(t)$</td>
<td>$n \times 1$ time-varying voltage noise vector</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>time step size</td>
</tr>
</tbody>
</table>
ABSTRACT

As technology scaling continues, the performance and reliability of integrated circuits become increasingly susceptible to power supply noises, such as IR drops and Ldi/dt noises in the on-chip power grids. Reduced supply voltage levels in the grid can increase the gate delay, leading to timing violations and logic failures. In order to ensure a reliable chip design, it is critical to verify that the power grid is robust, i.e., the power supply noises are acceptable for all possible runtime situations. Hence, power grid verification has become an indispensable step in modern design flow of integrated circuits.

Nowadays, it is common practice to verify power grids by simulation. Typically, an equivalent RC/RLC circuit model of the grid is extracted from the layout, and designers perform simulations to evaluate the power supply noises based on the current waveforms drawn by the circuit. As power grid simulation can only be performed after the circuit design is done, vectorless power grid verification has been introduced to enable early power grid verification with incomplete current specifications, so that the power grid design can be better tuned and optimized at early design stages, thus reducing the design time. Due to the increasing complexity of modern chips, power grid verification has become very challenging.

The broad goal of this dissertation is to explore efficient algorithms for verifying large-scale on-chip power grids. Specifically, we study parallel power grid transient simulation, vectorless steady-state verification and vectorless transient verification. Parallel forward and back substitution algorithms are designed for efficient transient simulation; a set of novel algorithms are developed to incrementally improve the runtime efficiency of vectorless steady-state verification; and an efficient approach is proposed for vectorless transient verification with novel constraint setting.
1.1 Power Grid Verification

On-chip power grids [56] deliver power to gates and cells in very-large-scale integrated (VLSI) circuits. They are also known as power distribution networks, power supply networks, power networks, and power rails. Ideally, the power grids should provide perfect voltage levels representing logic ‘0’ and ‘1’, i.e., VDD and GND, to each gate and cell. However, because of the intrinsic resistances, inductances and capacitances within the grids, there are power supply noises in the form of imperfect supply voltage levels at gates and cells, usually known as IR drops, Ldi/dt noises, and capacitively-coupled noises. Such voltage noises may cause longer gate delays [37], resulting in timing violations and logic errors. Thus have to be controlled within some acceptable range in order to guarantee correct functionality of the circuit. While power supply noises were not a major concern of circuit robustness in the past, they become increasingly important in modern chip designs using deep sub-micron technologies [29, 22], since the shrinking interconnect sizes lead to large IR drops, and the high clock frequency results in substantial amount of Ldi/dt noises. Moreover, as supply voltages are lowered to reduce power consumption while subthreshold voltages are decreased for better performance, the circuit becomes more vulnerable to power supply noises than ever before. Therefore, it is indispensable to verify that the power grid is “safe”, which means that the power supply noise at each node is acceptable for all possible runtime situations, whereas the increasing complexity of modern integrated circuits (ICs) has made the power grid verification a challenging task.

1.1.1 Power Grid Simulation. Nowadays, power grids are usually verified by simulating the grids to evaluate the voltage noise at each node. As the full-chip
transistor level simulation is prohibitively expensive, several power grid modeling
techniques [12, 13, 55] have been proposed for power grid simulation. An integrated
chip-and-package model typically consists of package model, on-chip power bus model,
and current model. Generally, the power grid is modeled as an RC/RLC circuit with
ideal voltage sources and time-varying current sources, which represent the current
drawn by the underlying circuitry. Given the current waveforms, one can simulate
the power grid using DC analysis (also known as steady-state analysis) and transient
analysis to evaluate power supply noises. Conventional SPICE-like general-purpose
circuit simulation tools [51, 71] may be employed for power grid analysis. However,
they suffer from long runtime and high memory usage, which make the simulation of
large-scale power grids prohibitive.

To assist power grid simulation, lots of algorithms for large-scale power grid
analysis have been proposed. Most of these algorithms fall under the category of time-
domain simulation. Since the power grid analysis is intrinsically to solve the linear
system representing the relationships between nodal voltages and current excitations,
matrix computation algorithms [27] for solving linear systems have been explored for
simulating the power grid, including LU factorization, Cholesky factorization [15],
and Krylov-subspace iterative methods, especially the preconditioned conjugate gra-
dient (PCG) algorithm, which was first studied in [14]. Although direct factorization
methods may provide promising performance as evaluated in [87], the factorization
process takes long runtime and they requires a large amount of memory to store the
LU matrices. On the contrary, the PCG algorithm usually consumes much less mem-
ory in comparison with direct factorization methods, but its convergence rate is highly
dependent on the quality of the preconditioner. In order to reduce the PCG runtime,
several different preconditioning schemes have been investigated, such as incomplete
Cholesky factorization [14], power grid structure based preconditioning [19], random
walk based stochastic preconditioning [62], deterministic random walk preconditioner
[74], support-graph preconditioner [7, 90], and fast transform-based preconditioners [21].

Except for the aforementioned linear algebra methods, multigrid [10] algorithms have been developed for power grid analysis. [53] first proposed to apply the multigrid technique for the simulation of power grids, and this method was further improved in [42, 43]. Different from the regular iterative multigrid method, this method is not iterative, and thus being a direct approach. Basically, it reduces the power grid to a coarser grid for fast analysis, and then maps the solution back to the original grid. Significant speedups have been achieved for both DC and transient analysis with reasonable errors. Besides, [66] proposed a power grid reduction scheme based on algebraic multigrid (AMG), while [34] studied an aggregation-based AMG method. In order to control the accuracy of power grid analysis, [91] designed an AMG PCG method. As shown by the TAU power grid simulation contests [45, 46], the AMG PCG method [83] is the most efficient for steady-state analysis, while the direct matrix decomposition method (LU/Cholesky factorization) [85, 84, 80] achieves the best performance for transient simulation if we have sufficient amount of memory.

Moreover, many other algorithms also have been proposed. An hierarchical methodology for full-chip power grid verification was introduced in [65], hierarchical power grid analysis with macromodeling was proposed in [88, 89], and hierarchical analysis with the IEKS (Improved Extended Krylov Subspace) method [73] was studied in [44]. Random walk was first employed for power grid analysis in [57], hierarchical random walk method was designed in [58, 61], and incremental solutions for power grid analysis using random walks were studied in [63, 8]. [68] and [86] investigated domain decomposition algorithms, [64] proposed an hierarchical matrix [30] approach, and [11] applied a sparse approximate inverse technique [31] for power grid simulation. In addition, frequency domain methods are alternative solutions
for power grid analysis. [33] employed discrete Fourier transform (DFT) to simulate the RLC power grid in frequency-domain, and then converted the frequency-domain voltage response back to time-domain.

To ensure power grid robustness, the power grid simulation must target the worst-case scenario, where gates and cells draw peak current from the grid because of simultaneously switching activities. As it is often difficult to determine the exact worst-case condition from the grid perspective, the choice of current excitations for grid simulation has a critical effect on the analysis results. Using safe but unrealistic simulation stimuli could result in pessimistic estimation of power supply noises and costly overdesigns of the grid. In order to generate realistic current waveforms for power grid analysis, [49] and [50] proposed an approach to approximate the worst-case condition using timing and spatial information from the circuit netlist and placement. It is shown that only a fraction of gates and cells switch in any timing window, and the resulting stimuli leads to more realistic noise estimation.

Although grid simulation is the main stream technique for sign-off verification of power grids, it has a few limitations. First, as there are too many current sources with different patterns, it is either intractable or computationally prohibitive to enumerate all the possible waveform combinations. In practice, some “typical” current waveforms are used for verification. Such current waveforms are either too pessimistic, or realistic but we are taking the risk of missing some important corner case. Second, it needs specific circuit implementation to provide detailed input current waveforms, and can only be employed after the circuit design is done. However, in practice, most power grid design is performed at the early design stage when the circuit details are not available, and early power grid verification is preferable for ease of grid modification. After the circuit design is completed, only small changes can be made to optimize the power grid in most cases. Hence, a verification technique,
which is not dependent on simulation, i.e. a vectorless approach, is highly desirable.

1.1.2 Vectorless Verification. To enable early power grid verification, vectorless verification approaches have been proposed [40], [25], [1]. The principle is to define the feasible set of all possible current waveforms using available knowledge of the underlying circuitry, and then solve optimization problems to compute the worst-case power supply noises. For example, at the early design stage, some circuit modules like high speed I/O components, SRAM (Static Random Access Memory) blocks and other hardware intellectual property (IP) cores may be available. There may also exist some reference designs which can be used to predict the characteristics of the ongoing design. Using such information, designers can estimate the full-chip power dissipation based on engineering judgements, and then initiate the power grid design. The estimated power consumption of individual circuit blocks can be transformed into corresponding current specifications for vectorless verification, and it may also serve as guidelines or even design specifications for circuit implementation. The bottom-line is that we must know some knowledge about the design so that we can verify the power grid.

As the feasible set of current excitations is defined by incomplete current specifications, the vectorless verification is intrinsically pessimistic. In order to get realistic power supply noises with small amount of overestimations, the current waveforms must be restricted by some realistic and tight constraints. Moreover, the vectorless verification is more challenging than power grid analysis due to the fact that it is difficult to solve the optimization problems for large-scale power grids. Several studies have been done to explore efficient vectorless techniques. In [40, 25, 1, 2, 76], feasible input current waveforms were characterized into linear current constraints to cover all possible current excitations, such that the worst-case voltage noises can be obtained after formulating and solving linear programming (LP) problems. The initial
vectorless approach [40] considered the DC analysis model, and it was extended to handle RC and RLC power grids in [25] and [1], respectively. To solve the LP problems efficiently, [2] used an approximate inverse technique to generate a reduced-size LP problem for each node, while we proposed a convex dual algorithm in previous work [76] to compute the worst-case power supply noises. [4] utilized the dominance relations among nodal voltage drops to reduce the number of LP problems to be solved, [41] and [5] used macromodeling technique to simplify the optimization problems for efficient incremental verification. Instead of solving the exact voltage noises, [3] proposed a fast approach to compute conservative bounds of voltage noises for RLC power grids.

In [59, 60], the current constraints were further refined by introducing integer variables to model the uncertain working modes of circuit blocks, and integer linear programming (ILP) problems were formulated and solved. [6] proposed to perform integrated verification of RC power grids, because their voltage noises have mutual effect through the decoupling capacitors. Additional current conservation constraints were introduced to model the current flows. As it is more convenient to specify current constraints in terms of power bounds, [18] proposed hierarchical power constraints for more realistic RLC power grid verification, and developed a sorting-deleting algorithm to solve the LP problem by exploiting the hierarchy of constraints. This approach was further improved in [75] by using model order reduction. Besides, [26] used max delta constraints to restrict the current change between successive time units, and employed wavelet analysis to characterize current excitations in order to identify the worst-case voltage fluctuations. [23] proposed current slope constraints to bound the minimum current transition time, and results show that assuming a zero transition time results in too pessimistic voltage noise prediction.

In summary, vectorless verification under current constraints is an important
alternative approach for verifying the power grid, especially at an early design stage. As linear constraints are often used to model the current excitations, vectorless verification is often formulated as LP problems, which are difficult to solve in a timely manner. Although several algorithms have been proposed, vectorless approaches usually take a large amount of runtime to compute voltage noises with high accuracy, or only noise bounds can be computed. Moreover, for a given power grid design, the worst-case voltage noises are dependent on the current constraints. It is critical to use realistic and tight constraints to model the current waveforms. Therefore, it is of great interest to develop novel solutions to evaluate realistic worst-case voltage noises efficiently, such that the vectorless verification of large-scale on-chip power grids can be practical.

1.2 Contributions

In this thesis work, we study parallel power grid transient simulation and vectorless verification. Since the vectorless verification can be performed in both steady state model and transient model, we have studied both models as discussed in the following subsections.

1.2.1 Parallel Transient Simulation. We employ LU decomposition and the PCG method for transient simulation of power grids, and design a parallel power grid simulator [80], which won the third place of TAU 2012 power grid simulation contest [46]. As forward and back substitution is a dominating routine of transient simulation and it is difficult to parallelize, we focus on investigating efficient parallelization techniques for it. Two approaches for parallel forward and back substitution have been proposed. The first approach exploits the parallelism by levelizing unknown variables according to the elimination tree [47] and calculating independent variables within the same level in parallel. The second approach uses a node ordering produced by nested dissection (ND) [39], so that the computed matrix \( L \) favors efficient parallelization.
of forward and back substitution. These two approaches are implemented in a direct solver with LU decomposition and a PCG solver with stochastic preconditioning [62], respectively. Experimental results with IBM power grid benchmarks [35, 54] show that the second approach is more efficient.

1.2.2 Steady-State Verification. For vectorless steady-state verification, we consider RC power grids, and continue to develop more efficient algorithms based on the dual algorithm proposed in our prior work [76]. We follow [2] to formulate the LP problem for vectorless power grid verification under linear current constraints, then decompose it into two orthogonal sub-problems. The first sub-problem is a power grid analysis problem, and can be solved efficiently by existing power grid analysis techniques. The second sub-problem is still an LP problem where the voltage noise is represented as an affine function of current sources. We solve these two sub-problems independently with user-specified error tolerances, and use a preconditioned conjugate gradient (PCG) power grid analyzer based on [62] to solve the power grid analysis problem.

Specifically, we propose a dual bound algorithm [78] for computing upper bound of voltage noise, a hierarchical matrix inversion algorithm [77] for computing the inverse of the power grid matrix, and constraint abstraction approach [81] to check the grid safety fast.

1.2.2.1 Dual Bound Algorithm. As the performance of our previous dual algorithm [76] degrades when more current constraints are specified, we design a dual bound algorithm [78] for solving the LP problem with relatively large number of current constraints. We exploit the structure of the dual problem, and formulate a reduced-size LP problem for each node by using piecewise linear approximation technique. Note that our reduced-size LP problem is different from that of [2], which is based on the approximate inverse technique. As the reduced-size LP problem has
small amount of decision variables, it can be efficiently solved by standard LP solver in small amount of runtime. The dual bound algorithm can serve as an alternative approach to check grid safety. Experimental results show that both the dual algorithm and the dual bound algorithm achieve significant speedup in solving the LP problem when there are relatively small number of current constraints, and the dual bound algorithm is able to maintain the performance gain when more current constraints are specified, while introducing small amount of overestimation for voltage noises.

1.2.2.2 Hierarchical Matrix Inversion. As the second sub-problem, i.e., the LP problem, can be solved efficiently by the dual algorithm or the dual bound algorithm, the first sub-problem, i.e., the power grid analysis problem, becomes the major performance bottleneck. For each node, the power grid analysis in the first step computes the corresponding row/column in the inverse of the power grid matrix. Therefore, for all the nodes, essentially the whole inverse of the power grid matrix is computed. In comparison with [76, 78] where the rows are computed independently by the preconditioned conjugate-gradient (PCG) method [27, 14], we propose a hierarchical algorithm to speed up the matrix inversion by exploiting the fact that there are dependencies among the rows/columns in the inverse of the matrix.

Different from the conventional hierarchical matrix approaches [30], where the inverse matrix is computed by using the inversions of sub-matrices, our hierarchical matrix inversion algorithm partitions the power grid into several clusters and a set of external neighbor nodes, performs partial inversions on each cluster directly, computes the rows/columns corresponding to these external neighbor nodes with the PCG method at first, then combines partial inversions and computed rows to generate the rows corresponding to the nodes within each cluster. Our algorithm also differs from the hierarchical power grid analysis [89], which uses macro-modeling to reduce the problem size for solving a single voltage noise vector (e.g. a row in the inverse)
and does not exploit the row dependencies. The proposed hierarchical algorithm can be viewed as a combination of the direct method for solving linear system, e.g. LU factorization, and the iterative method, e.g. the PCG method. The advantages of both are seen in our algorithm – while small clusters are solved efficiently by direct methods, extremely large power grids that cannot be usually handled by direct methods and require iterative solvers can be verified with our algorithm. We integrate the proposed hierarchical matrix inversion algorithm with the dual approach in [76] to solve the vectorless power grid verification problem for RC power grids, the resulting verification algorithm further reduces the runtime, thus making the vectorless verification more efficient.

1.2.2.3 Constraint Abstraction. Although the dual bound algorithm and the hierarchical matrix inversion algorithm significantly improve the runtime-efficiency, the computation cost of steady-state verification remains much higher than that of power grid simulation. Typically, for full-chip power grid verification, both the number of linear programs to be solved and the size of each linear program are proportional to the size of the grid. As a result, the computation cost increases dramatically when the grid size becomes larger.

To further improve the runtime-efficiency, propose to reduce the computation cost by constraint abstraction. Since a localized region of the power grid can be verified based on the boundary condition (i.e., the power supply noises at the neighboring nodes), we propose novel boundary constraints to define a partial specification of boundary condition, and perform full-chip grid verification in a divide-and-conquer manner to compute conservative bounds of power supply noises. The boundary constraints provide a high-level abstraction of the grid environment for the region of interest, and enables dramatic performance gain by significantly reducing the computation cost for verification. Experimental results with both synthetic power grids and
IBM power grid benchmarks show that the proposed approach achieves up to about 17X speedup over the prior art [78], and the overestimation of power supply noises is reasonably small. In particular, a power grid with 562K nodes can be verified in about 1 hour.

1.2.3 Transient Verification. For vectorless transient verification, we consider an integrated RLC power grid model with both VDD and GND networks, and propose to perform sign-off verification with novel transient current constraints. The main contributions are as follows.

1. Novel transient current constraints are proposed to restrict the current waveforms for more realistic scenarios, leading to less pessimistic voltage noise predictions.

2. A general methodology to verify RLC power grids by transient simulation and noise optimization is developed, so that efficient power grid analysis algorithms can be leveraged for vectorless verification. This methodology eliminates the verification difficulty introduced by inductors, since inductors can be properly handled during transient simulation. We rigorously prove that the voltage noise at a node, either at a particular time point or cumulatively over a time interval, can be represented as an affine function of current excitations, which enables us to decompose the vectorless verification problem into two orthogonal subproblems. The first sub-problem is a power grid transient analysis problem that computes the affine function, which can be solved efficiently by existing power grid analysis algorithms. The second sub-problem is a linear programming (LP) problem that optimizes the affine function under current constraints, which is difficult to solve for practical power grids.

3. A variable reduction algorithm is proposed for solving the LP problem. It
removes insignificant current variables according to a user-specified error tolerance, so that the resulting reduced-size LP problem can be efficiently solved by standard LP solvers to obtain conservative bounds of voltage noises. Results show that the proposed algorithm significantly speeds up solving the LP problem for vectorless verification.

Different from previous works [3] and [6], which compute bounds of voltage noises based on DC current constraints (i.e. local, global and equality constraints detailed in Section 5.1.2) only, the proposed approach solves the exact worst-case voltage noises under both DC and transient current constraints. In other words, we study the exact approach for vectorless verification of RLC power grids with more realistic constraint settings. The verification techniques of [3] and [6] cannot be extended to handle transient current constraints, because they iteratively compute bounds of voltage noises at advancing time steps till convergence. The approximate inverse technique [2] may be applied for verifying RLC power grids, but it suffers the major limitation that there is no well-defined accuracy guarantee. Hence, we propose to perform transient simulation to setup the LP problem, so that the computed voltage noise estimations are accurate. Prior works [18] and [75] investigate similar problem but limit the current/power constraints to hierarchical structure for efficient solution of the LP problem, while the proposed approach targets more general constraint settings.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we briefly introduce power grid simulation and vectorless verification. Chapter 3 presents parallel power grid transient simulation. Vectorless steady-state verification and vectorless transient verification are presented in Chapters 4 and 5, respectively. Finally, Chapter 6 presents the concluding remarks and discusses future research topics.
In this chapter, we briefly introduce power grid simulation and vectorless verification.

2.1 Power Grid Simulation

Consider an RC power grid model as illustrated in Figure 2.1. The grid consists of resistors, capacitors, VDD pads, and current sources which represent the currents drawn by the underlying circuit. Let $\mathbf{v}(t)$ be the vector of voltage drops at the nodes that are not VDD pads, $\mathbf{i}(t)$ be the vector of all the current sources attached to the aforementioned non-VDD nodes, $G$ be the conductance matrix, and $C$ be the matrix introduced by capacitors. According to [40], we have the following system of equations,

$$G\mathbf{v}(t) + C\dot{\mathbf{v}}(t) = \mathbf{i}(t).$$

Note that if we assume that there are $n$ non-VDD nodes, then both $\mathbf{v}(t)$ and $\mathbf{i}(t)$ are
$n \times 1$ vectors, $G$ is an $n \times n$ symmetric positive definite M-matrix, and $C$ is an $n \times n$ diagonal matrix.

The aforementioned power grid is a VDD network, while practical power grids also have GND networks. In the GND network, there are GND pads instead of VDD pads, and the current sources flow into the nodes. Let $\mathbf{v}(t)$ be the vector of ground bounces at the nodes that are not GND pads, and $\mathbf{i}(t)$ be the vector of current sources flowing into these non-GND nodes. Then equation (2.1) still holds. Therefore, for either VDD network or GND network, the relationship between voltage noises $\mathbf{v}(t)$ and current sources $\mathbf{i}(t)$ always satisfies equation (2.1).

Typically, power grid simulation is performed under DC analysis (also known as steady-state analysis) model and transient analysis model. For DC analysis, the system equation (2.1) is reduced to

$$G\mathbf{v} = \mathbf{i}, \quad (2.2)$$

where $\mathbf{v}$ and $\mathbf{i}$ are the vectors of voltage drops and current sources, respectively. For transient analysis, equation (2.1) can be discretized by using the Backward Euler method or the Trapezoidal Rule with a time step $\Delta t$. For example, with the backward Euler method, equation (2.1) can be reduced to

$$\left(G + \frac{C}{\Delta t}\right)\mathbf{v}(t) = \mathbf{i}(t) + \frac{C}{\Delta t}\mathbf{v}(t - \Delta t). \quad (2.3)$$

Obviously, both (2.2) and (2.3) are systems of linear equations, and their left-hand-side matrices are symmetric positive definite. In fact, this property also holds for RLC power grid model as studied in [14]. Generally, for either RC or RLC power grids, the system equation of transient analysis can be formulated in nodal analysis.
form

\[ \mathbf{A} \mathbf{v}(t) = \mathbf{b}(t), \]  

(2.4)

where \( \mathbf{A} \) is an \( n \times n \) matrix stamped from the power grid, and \( \mathbf{b}(t) \) is an \( n \times 1 \) vector computed by using the nodal voltages at previous time steps and current excitations. Note that \( \mathbf{A} \) is a symmetric positive definite \( \mathcal{M} \)-matrix.

In order to solve the DC analysis equation (2.2) and transient analysis equation (2.4) efficiently, both direct matrix decomposition methods and iterative methods have been explored for power grid simulation as discussed in Section 1.1.1. However, due to the increasing complexity of on-chip power grids, power grid analysis has become very challenging for excessive runtime and large memory consumption. Fortunately, the massively parallelism of modern computers enables us to explore parallel power grid simulation methods.

2.2 Vectorless Verification

Different from power grid simulation with given current waveforms, vectorless verification approaches employ current constraints to define a feasible set of all possible current excitations, and make worst-case voltage noise predictions accordingly. There are two kinds of commonly-used current constraints: local constraints and global constraints. The local constraints define an upper bound for every current source,

\[ 0 \leq \mathbf{i}(t) \leq \mathbf{I}_L, \forall t, \]

where \( \mathbf{I}_L \geq 0 \) is an \( n \times 1 \) upper bound vector. The global constraints define upper bounds for groups of current sources,

\[ U_i(t) \leq \mathbf{I}_G, \forall t. \]
Here we assume that there are $m$ current source groups and $m$ is much smaller than $n$, $U$ is an $m \times n$ 0/1 matrix indicating the assignments of current sources to groups, and $I_G \geq 0$ is an $m \times 1$ upper-bound vector.

Except for local and global constraints, some other constraints have also been proposed to characterize current waveforms. For example, current conservation constraints were proposed in [6] for the verification of integrated RC power grids. In [26], the authors introduced max delta constraints to bound the change in current between successive time units. Moreover, [23] used current slope constraints to bound the minimum current transition time. Both max delta constraints and current slope constraints restrict the transition characteristics of current sources. In addition, hierarchical power constraints were proposed in [18] to bound the power consumption of circuit blocks.

Theoretically, a combination of all kinds of constraints can better characterize the feasible current excitations for vectorless power grid verification. However, in practice, it may not be possible to verify the grid with all of these constraints, since it is often too computationally expensive or some constraints are not available. Hence, different constraint settings are employed for different applications.

Let $F(v(t), i(t)) = 0$ denote the system equation of the power grid (e.g., equation (2.1)), and $\mathcal{I}_F$ be the feasible set of current excitations defined by the current constraints. Then the vectorless verification problem can be represented as following optimization problem. For each node $1 \leq j \leq n$,

$$\begin{align*}
\text{Maximize/Minimize } v_j(t), \\
\text{subject to } F(v(t), i(t)) = 0, \ i(t) \in \mathcal{I}_F,
\end{align*}$$

(2.5)

where $v_j(t)$ is the voltage noise of node $j$ at time $t$. It is to be noted that both
the maximum and the minimum of voltage noises need to be computed for transient verification, since the nodal voltage may fluctuate in both directions. Problem (2.5) aims to compute the worst-case voltage noises over all time $t$. In practice, it is very difficult to calculate the exact optimal values of (2.5) given limited computing resources, and problem (2.5) is of theoretical interest only. Practical vectorless verification methods typically compute the worst-case voltage noises within a time interval (e.g., from $t = 0$ to 1ns or 10ns), or calculate upper bounds of the worst-case voltage noises.

Similar to power grid simulation, vectorless verification can also be performed under steady-state analysis model and transient analysis model, which will be detailed in Chapters 4 and 5, respectively. As current constraints define a superset of possible current excitations, vectorless approaches are intrinsically pessimistic, i.e., the worst-case voltage noises computed by solving problem (2.5) will be upper bounds of real voltage noises in the power grid. It is important to use realistic and tight current constraints for vectorless verification, so that the voltage noise estimations are reliable. Moreover, vectorless verification is much more compute-intensive than power grid simulation. The computation cost of existing approaches is often too high to enable time-efficient verification of large-scale power grids. Hence, it is of great interest to investigate more efficient vectorless verification techniques.
Parallelizing power grid transient simulation with factorization-based direct or preconditioned iterative methods is a challenging task due to the data dependency among forward and back substitution. In this thesis work, we propose two approaches to parallelize forward and back substitution. The first approach exploits the parallelism by computing independent variables in parallel; the second approach resolves the data dependency by leveraging a node ordering produced by nested dissection. According to the requirements of TAU 2012 power grid simulation contest, a parallel power grid transient simulator is developed and tested with IBM power grid benchmarks. Results show that the second approach with the nested-dissection-based node ordering is more efficient.

3.1 Background

As discussed in Section 2.1, the key problem of power grid transient analysis is to solve the system equation $A\mathbf{v}(t) = \mathbf{b}(t)$ at different time steps. Direct matrix decomposition methods [27], e.g., LU and Cholesky factorization algorithms, can be employed for fast power grid analysis. However, these direct methods consume a large amount of memory because of the large number of fill-ins generated during the factorization process, especially for large-scale power grids. To achieve memory efficiency, several iterative algorithms [14, 83, 90, 19, 74] have been proposed for power grid analysis. These algorithms often use the preconditioned conjugate gradient (PCG) method [27], and build a high quality preconditioner matrix $M$ by exploiting the structure of the power grid for fast convergence.

The factorization or preconditioner calculation only needs to be done once if a fixed time step size is employed. At each time step, direct methods solve the
system equation by using a forward substitution followed by a back substitution, while iterative methods often solve it through PCG iterations, each of which involves solving the linear system $Mz_k = r_k$ through forward and back substitution, computing a matrix vector multiplication, and a few vector operations. To parallelize these direct and iterative methods, it is a must to parallelize forward and back substitution, since it is a critical routine for transient analysis.

3.2 Parallel Forward and Back Substitution

3.2.1 Levelized Parallelization. In forward and back substitution, the computation of unknown variables must follow specific order because of the data dependency between variables. Such dependency can be represented as elimination tree(s) [47] according to the structure of L/U matrices. For parallel implementation, a straightforward approach is to perform dynamic task scheduling guided by the elimination tree, where each task computes a single variable. However, the parallel performance will suffer because the scheduling overhead often dominates the runtime. Previous

---

$M$ is the $n \times n$ preconditioner matrix, $z_k$ is an $n \times 1$ vector of the $k$th iteration, and $r_k$ is the $n \times 1$ residue vector of the $k$th iteration. Additional details of the PCG method can be found in [27].
work [32] parallelizes forward and back substitution based on supernodes, which are groups of task nodes within the elimination tree. Unfortunately, it is difficult to form supernodes for power grid analysis due to the high sparsity of power grid matrices.

To achieve parallel performance, we propose a levelized approach for parallel implementation of forward and back substitution. The level of each task node in the elimination tree is defined as the length of the longest path from any leaf node to the node itself. All task nodes are categorized into different levels, so that the tasks within the same level are independent and can be executed in parallel. Figure 3.1 shows an example of lower triangular matrix $L$, its elimination tree and task mapping to threads. Forward substitution (for solving $Lx = b$) is performed from level 0 to the maximum level, while back substitution (for solving $L^T x = b$) is executed from the maximum level to level 0. This approach is implemented in a direct solver with LU decomposition for power grid transient simulation. In order to reduce the parallel overhead, we only parallelize the levels, which have at least 10000 task nodes.

3.2.2 ND-based Parallelization. The aforementioned levelized parallelization is based on given L/U matrices, and only limited parallelism can be exploited. For better parallel efficiency, we propose to use an elimination ordering generated by nested dissection (ND) [39] during the factorization or preconditioning process, so that the computed matrix $L$ have parallelizable structure. To evaluate the performance, we implement this approach in a PCG solver with a random-walk-based stochastic preconditioner [62]. By using random walks with proper node ordering, one can obtain an incomplete $L^TDL$ factorization, whose matrix $L$ has desired non-zero patterns for parallel forward and back substitution.

In this section, we first review the principle of stochastic preconditioning, then present the proposed node ordering algorithm and the corresponding parallel implementation of forward and back substitution.
3.2.2.1 Stochastic Preconditioning. Consider a diagonally dominant SPD $M$-matrix $A$, which represents an un-directed network, e.g., the power grid. Assume that the network is connected and the nodes are numbered from 1 to $n$. Let $a_{ij}$ be the $(i, j)$ entry of matrix $A$. Random walks are defined on the network using a transfer probability for a step from node $i$ to $j$ of

$$p_{i\rightarrow j} = -\frac{a_{ij}}{a_{ii}}, \forall 1 \leq i \neq j \leq n,$$

and from node $i$ to itself and an absorbing node $0$,

$$p_{i\rightarrow i} = 0, \quad p_{i\rightarrow 0} = 1 - \sum_{j=1}^{n} p_{i\rightarrow j} = \sum_{j=1}^{n} \frac{a_{ij}}{a_{ii}}, \quad \forall 1 \leq i \leq n.$$

The stochastic preconditioning is based on the random walks that originate from node $i \neq 0$ and terminate when it arrives at a node $0 \leq j < i$. Let $P_{i\rightarrow j}$ be the probability of all such random walks from node $i$ to $j$ and let $E_i$ be the expectation of the number of times that one such random walk originating from node $i$ passes itself. Consider an $n \times n$ unit lower triangular matrix $L$ where its $(i, j)$ entry

$$l_{ij} = -P_{i\rightarrow j}, \forall 1 \leq j < i \leq n,$$

and an $n \times n$ diagonal matrix $D$ where its diagonal entry

$$d_{ii} = \frac{a_{ii}}{E_i}, \forall 1 \leq i \leq n.$$

It has been proved [62] that $A = L^TDL$.

In practice, by running sufficiently large number of random walks originating from each node to estimate $P_{i\rightarrow j}$ and $E_i$, we can compute an incomplete $L^TDL$
factorization, which serves as the preconditioner matrix \( M \) for the PCG method. Then in each PCG iteration, \( Mz_k = r_k \) can be solved by a backward substitution, a vector operation and a forward elimination.

### 3.2.2.2 Node Ordering and Parallel Implementation.

Figure 3.2 presents the node ordering algorithm for stochastic preconditioning. It uses nested dissection to number the nodes in a divide-and-conquer manner. This algorithm proceeds by finding a set of nodes \( S \) which separate the nodes of interest \( N \) into two disconnected subsets \( P_1 \) and \( P_2 \), assigning the smallest numbers to the separating nodes in \( S \), and numbering the nodes in \( P_1 \) and \( P_2 \) recursively until the node subsets are sufficiently small. In order to achieve cache efficiency, the reverse Cuthill-McKee (RCM) ordering [20] is employed for the small subsets of nodes and the sets of separating nodes. This recursive algorithm for node ordering results in node subsets split by sets of separating nodes. Although the primal objective of general nested dissection is to reduce the number of fill-ins, we utilize it to generate the node ordering for stochastic preconditioning, so that the computed matrix \( L \) favors parallel forward and back substitution.

As the nodes in \( P_1 \) and \( P_2 \) are separated by the nodes in \( S \), we must have the \((i, j)\) entry of \( L \) being 0 for any node \( i \in P_2 \) and \( j \in P_1 \), because the random walks starting from the nodes in \( P_2 \) will terminate at the separating nodes in \( S \) before reaching any node in \( P_1 \). Hence, the unknown variables for \( P_1 \) and \( P_2 \) can be computed in parallel. A dependency acyclic graph (DAG) for forward and back substitution can be built according to the relation between sets of separating nodes and node subsets. Figure 3.3 shows an example of the non-zero pattern of matrix \( L \) and the corresponding DAG. The nodes are separated into four subsets P1 to P4 by three sets of separating nodes S1 to S3 (through two levels of recursive node

\(^2 \)\( S \) is often called graph separator [39].
**Algorithm** Node Ordering

**Input:** The network associated with matrix $A$.

**Output:** A node ordering for stochastic preconditioning.

```
1    Node_Ordering($\mathcal{N}, k$)
2    Set_of_Nodes $\mathcal{N}$;
3    integer $k$;
4    {
5      if ($|\mathcal{N}|$ is sufficiently small) {
6        Number the nodes in $\mathcal{N}$ from $k$ to $k + |\mathcal{N}| - 1$
7            using the RCM ordering;
8      }
9      else {
10         Find a set of nodes $\mathcal{S}$, which separates $\mathcal{N}$ into
11            two parts $\mathcal{P}_1$ and $\mathcal{P}_2$;
12         Number the nodes in $\mathcal{S}$ from $k$ to $k + |\mathcal{S}| - 1$
13            using the RCM ordering;
14         Node_Ordering($\mathcal{P}_1, k + |\mathcal{S}|$);
15         Node_Ordering($\mathcal{P}_2, k + |\mathcal{S}| + |\mathcal{P}_1|$);
16      }
17    }
```

Figure 3.2. The node ordering algorithm based on nested dissection for stochastic preconditioning. All the nodes can be numbered from 1 to $n$ by a call to `Node_Ordering($\mathcal{N}, 1$)`.
ordering). In forward substitution, a top-down approach on the DAG should be used, i.e., the unknown variables for the sets of separating nodes must be solved first. In back elimination, a bottom-up approach on the DAG should be employed, i.e., the unknown variables for node subsets must be computed before the calculation of unknown variables for separating nodes.

Forward and back substitution can be parallelized by using dynamic task scheduling according to the DAG, where each task solves the variables for a set of separating nodes or a node subset. However, as there are only relatively small amount of separating nodes, this dynamic approach is less efficient than the static approach, which only executes the tasks for node subsets in parallel. Therefore, the static approach is employed in our implementation.

3.3 Transient Simulator – IITPGS

A power grid transient simulator called IITPGS has been developed in C++ with Intel Threading Building Blocks (TBB) [36] for parallelization. It supports both backward Euler and trapezoidal rule [52] for transient simulation, and incorporates
Figure 3.4. Simulation flow of IITPGS.
a direct solver using LU factorization and a PCG solver using stochastic preconditioning. The direct solver with the backward Euler method was submitted to TAU 2012 power grid simulation contest [46]. As shown in Figure 3.4, our simulator first parses the circuit and builds matrices, then performs DC analysis to calculate the DC operating point as the initial condition for transient simulation. After that, it performs factorization or preconditioning, and then simulates transient time steps. For each time step, the simulator first computes $b(t)$ in the system equation (2.4) by calculating current source values and performing some matrix vector multiplications and vector operations, and then solves (2.4) by using forward and back substitution or the PCG method.

We employ the parallel LU factorization tool NICSLU [16, 17] to perform LU decomposition for the direct solver as well as DC analysis. Forward and back substitution is parallelized as detailed in Section 3.2. In the proposed node ordering algorithm, the sets of separating nodes are identified by partitioning a hypergraph model of the power grid [77] with hMETIS [69]. For efficient calculation of $b(t)$, the current source values are evaluated in parallel. Moreover, all the matrix vector multiplications and vector operations (including those in the PCG method) have been parallelized, resulting in a highly parallel simulator, except that parsing circuit, building matrices and stochastic preconditioning are serial.

3.4 Experimental Results

We test our transient simulator with IBM power grid benchmarks [35, 54]. All experiments are carried out on a 64-bit Linux server with 2.67GHz Intel X5650 processor and 64GB memory. As the server has 12 cores, we run transient simulation using up to 12 threads. Table 3.1 shows the solution accuracy of the direct solver. The accuracy results of the PCG solver are not listed, because they are approximately the same as that of the direct solver. It can be seen that trapezoidal rule provides much
Table 3.1. Solution Accuracy of IITPGS (Direct Solver). $E_{max}$: maximum error in volt; $E_{avg}$: average error in volt.

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>Backward Euler $E_{max}$</th>
<th>Backward Euler $E_{avg}$</th>
<th>Trapezoidal Rule $E_{max}$</th>
<th>Trapezoidal Rule $E_{avg}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibmpg1t</td>
<td>1.08e-03</td>
<td>1.61e-04</td>
<td>5.30e-05</td>
<td>4.27e-06</td>
</tr>
<tr>
<td>ibmpg2t</td>
<td>8.92e-04</td>
<td>1.53e-04</td>
<td>4.57e-05</td>
<td>4.34e-06</td>
</tr>
<tr>
<td>ibmpg3t</td>
<td>8.73e-04</td>
<td>1.31e-04</td>
<td>4.70e-05</td>
<td>4.40e-06</td>
</tr>
<tr>
<td>ibmpg4t</td>
<td>2.11e-03</td>
<td>1.63e-04</td>
<td>1.31e-04</td>
<td>8.80e-06</td>
</tr>
<tr>
<td>ibmpg5t</td>
<td>5.63e-04</td>
<td>9.08e-05</td>
<td>3.10e-05</td>
<td>3.20e-06</td>
</tr>
<tr>
<td>ibmpg6t</td>
<td>6.47e-04</td>
<td>1.11e-04</td>
<td>3.30e-05</td>
<td>3.68e-06</td>
</tr>
</tbody>
</table>

higher accuracy than backward Euler. For performance tests, we use the backward Euler method in order to be consistent with the contest results.

The runtime of our simulator with a single thread is illustrated in Table 3.2. The branches with resistance being $< 10^{-6}$ are treated as shorts during simulation. For the PCG solver, we report the runtime of forward and back substitution (FBS) as well as the total PCG runtime, since the other runtime components are comparable to that of the direct solver. Clearly, the direct solver is much more efficient than the PCG solver in terms of runtime. As shown in Figure 3.5(a), we can only achieve minor FBS runtime reduction by using the levelized parallelization proposed in Section 3.2.1, because relatively a small portion of unknown variables are computed in parallel, and there is a high memory access to computation ratio. The total runtime of the direct solver (Figure 3.5(b)) shows a slightly better scaling due to efficient parallelization of LU factorization and computing $b(t)$. The parallel direct solver consumes more memory than the serial version (Figure 3.5(c)) because of the high memory usage of parallel LU factorization with NICSLU. Figure 3.6 shows the performance scaling of the PCG solver. We can obtain about 2X speedup with 4 threads for forward and back substitution and the overall PCG method. It can be seen in Figure 3.5(a)
Table 3.2. Simulation Runtime of IITPGS (1 thread). Nodes: the number of nodes after merging nodes connected by shorts; DC: DC Analysis; Fact.: factorization; Comp. $b(t)$: compute $b(t)$; FBS: forward and back substitution; nnz: the number of non-zeros; PCG: the PCG method for solving the system equation (2.4); runtime is in seconds.

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>Nodes</th>
<th>Parse</th>
<th>DC</th>
<th>Fact.</th>
<th>Comp. $b(t)$</th>
<th>FBS</th>
<th>Total</th>
<th>nnz in $L&amp;U$</th>
<th>FBS</th>
<th>PCG</th>
<th>nnz in $L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibmpg1t</td>
<td>25372</td>
<td>0.61</td>
<td>0.08</td>
<td>0.07</td>
<td>1.10</td>
<td>1.12</td>
<td>3.06</td>
<td>375606</td>
<td>16.38</td>
<td>23.65</td>
<td>158466</td>
</tr>
<tr>
<td>ibmpg2t</td>
<td>163907</td>
<td>1.90</td>
<td>2.24</td>
<td>2.07</td>
<td>6.04</td>
<td>25.16</td>
<td>37.87</td>
<td>8916576</td>
<td>159.78</td>
<td>214.92</td>
<td>1610429</td>
</tr>
<tr>
<td>ibmpg3t</td>
<td>475297</td>
<td>12.43</td>
<td>4.02</td>
<td>3.90</td>
<td>31.66</td>
<td>54.59</td>
<td>108.19</td>
<td>18168500</td>
<td>828.87</td>
<td>1028.23</td>
<td>8508006</td>
</tr>
<tr>
<td>ibmpg4t</td>
<td>1211402</td>
<td>14.77</td>
<td>120.06</td>
<td>152.60</td>
<td>51.86</td>
<td>353.13</td>
<td>696.93</td>
<td>166379518</td>
<td>3134.98</td>
<td>3813.42</td>
<td>23917043</td>
</tr>
<tr>
<td>ibmpg5t</td>
<td>1013697</td>
<td>19.15</td>
<td>12.14</td>
<td>11.61</td>
<td>67.06</td>
<td>111.08</td>
<td>224.78</td>
<td>37836900</td>
<td>6265.30</td>
<td>7327.84</td>
<td>25147593</td>
</tr>
<tr>
<td>ibmpg6t</td>
<td>1530943</td>
<td>31.09</td>
<td>10.62</td>
<td>9.82</td>
<td>113.43</td>
<td>152.69</td>
<td>323.40</td>
<td>47505850</td>
<td>7188.15</td>
<td>8393.85</td>
<td>37886570</td>
</tr>
</tbody>
</table>
Figure 3.5. Performance scaling of IITPGS (direct solver).
Figure 3.6. Performance scaling of IITPGS (PCG solver).
and 3.6(a) that the ND-based parallelization (Section 3.2.2) is more efficient than the levelized parallelization (Section 3.2.1). Generally, the parallel solvers only achieves about 2X speedup due to the high memory access to computation ratio of sparse matrix computation routines.
CHAPTER 4
STEADY-STATE VERIFICATION

Vectorless steady-state verification approaches evaluate the worst-case IR drops in the grid, and can also provide bounds for the worst-case voltage noises in RC power grids. Earlier works are limited to small grid sizes since it is very difficult to solve the optimization problem of steady-state verification efficiently. In this chapter, we consider RC power grids, and propose three highly efficient algorithms for steady-state verification, including a dual bound algorithm to compute upper bound of voltage noise, a hierarchical matrix inversion algorithm to compute the inverse of the power grid matrix, and a constraint abstraction approach for fast estimation of power supply noises across the chip.

4.1 Background

4.1.1 Problem Formulation. As summarized in [2], vectorless verification of RC power grids can be performed under two power grid analysis models, i.e., the DC analysis model and the transient analysis model with a time step $\Delta t$. For DC model, only a resistive grid is considered, the worst-case voltage noises can be obtained by solving optimization problems; for transient model, an upper bound vector of exact worst-case voltage noises can be derived by solving similar optimization problems and calculating a matrix vector multiplication. Conventional “steady state verification” only refers to the verification using DC model. However, in this thesis, we generalize it to also include the verification using transient model, because only steady state analysis is required for computing the voltage noise upper bounds. For both models, the key problem can be formulated as the following $\text{maxVN-LCC} \ (\text{maximum voltage noise under linear current constraints})$ problem.

Problem 1 (maxVN-LCC) Assume there are $n$ non-VDD nodes and $m$ global cur-
rent constraints. Let $A = G$ if the DC analysis model is of concern or $A = G + \frac{C}{\Delta t}$ if the transient analysis model is of concern. Let $I_L \geq 0$ and $I_G \geq 0$ be the upper bound vectors for local and global current constraints respectively. Let $U$ be a 0/1 matrix. Suppose that $v$ and $i$ are the vectors of decision variables for voltage noises and current sources respectively, and let $v_l$ be the $l$’th component of $v$ for $1 \leq l \leq n$. Solve the following optimization problem for every $1 \leq l \leq n$,

\[
\text{Maximize } v_l \quad \text{s.t.} \\
A v = i, \quad 0 \leq i \leq I_L, U i \leq I_G.
\]  

(4.1)

For every $1 \leq l \leq n$, the optimization problem in (4.1) is a linear programming (LP) problem since both the objective function and the constraints are linear functions of the decision variables. As there are $n$ optimization problems and $n$ is usually large for any practical power grid, such LP problems have to be solved very efficiently.

Intuitively, instead of sending the whole (4.1) to an LP solver, one would decompose the optimization problem first utilizing the fact that $A$ is an M-matrix and thus it is invertible. Let $e_l$ be an $n \times 1$ vector of all 0’s except the $l$’th component being 1. Let $c_l \triangleq A^{-1} e_l$, which can be computed by solving $A x = e_l$ using any power grid analysis algorithm. Because $A v = i$ and $A$ is symmetric, we have $v_l = c_l^T i$. Then the original $\text{maxVN-LCC}$ problem is decomposed into the following two sub-problems.

I: Compute $c_l$ by solving $A x = e_l$,  
(4.2)

II: Maximize $v_l = c_l^T i$ \quad s.t.  
(4.3)

\[
0 \leq i \leq I_L, U i \leq I_G.
\]

The first sub-problem (4.2) is a power grid analysis problem. The second sub-problem (4.3) is still an LP problem but it is much easier to solve in comparison with (4.1),
because the voltage noise is represented as a linear function of current sources.

4.1.2 Previous Approaches. The aforementioned problem decomposition has been explored in both [2] and our previous work [76]. As there are still a substantial number of decision variables and constraints in the LP problem (4.3), these two works take different approaches to solve both sub-problems efficiently. In [2], it is proposed to compute an approximated $c_l$ with a small number of non-zero components, so that most decision variables and constraints can be dropped from the LP problem, which can then be solved efficiently by any LP solver. However, as the number of non-zero components depends on the accuracy of the approximation, when a higher level of accuracy is desired, the reduction of the decision variables and constraints diminishes and so does the efficiency to solve the LP problem. Moreover, to generate the approximated $c_l$ introduces significant running time overhead compared to efficient power grid analysis techniques as demonstrated by [76].

Different from [2], [76] adopts an approach whose running time is not strongly dependent on the accuracy of the solution. The accuracy of the overall solution is established from those of the two sub-problems. Let $\delta_{\text{inv}}$ and $\delta_{\text{lp}}$ be two user-specified error-tolerances for the two sub-problems respectively. For the first sub-problem, let the residual vector $r \triangleq e_l - Ac_l$. Let $r_j$ be the $j$'th component of $r$ and $\Delta$ be the maximum component of the solutions of the linear equations $Ax = I_L$. The computed $c_l$ should satisfy that,

$$||r||_1 \triangleq \sum_{j=1}^{n} |r_j| \leq \frac{\delta_{\text{inv}}}{\Delta}. \quad (4.4)$$

For the second sub-problem, let $\hat{v}_l$ be the optimal value of the LP problem (4.3) with the computed $c_l$, any algorithm that solves (4.3) can be terminated with a solution $v_l^+$ when

$$v_l^+ - \delta_{\text{lp}} \leq \hat{v}_l \leq v_l^+. \quad (4.5)$$
Let $v_l^*$ be the optimal value of the LP problem (4.3) for the exact $c_l$. It is proved in [76] that the following theorem must hold.

**Theorem 1** Suppose both (4.4) and (4.5) hold. Let $\overline{v_l} \triangleq v_l^* + \Delta \sum_{j=1}^n \max(r_j, 0)$ be the conservative bound of the worst-case voltage noise. Then,

$$\overline{v_l} - \delta_{inv} - \delta_{lp} \leq v_l^* \leq \overline{v_l}.$$ 

A dual algorithm called DualVD is proposed in [76] to solve the LP problem (4.3) efficiently. It is shown that (4.3) is equivalent to the following simplified dual problem,

$$\text{Minimize } D(\gamma) \text{ s.t. } \gamma \geq 0, \quad (4.6)$$

where $D(\gamma) \triangleq I_L^T \gamma + \sum_{j=1}^n I_{L,j} \max(0, c_{l,j} - u_j^T \gamma)$. Here $\gamma$ are the $m$ decision variables corresponding to the Lagrangian multipliers of the global constraints, $I_{L,j}$ is the $j$’th component of $I_L$, $c_{l,j}$ is the $j$’th component of $c_l$, and $u_j$ is the $j$’th column of $U$. It is further shown that $D(\gamma)$ is a convex function of $\gamma$ and thus (4.6) is a convex programming problem that is then solved by Kelley’s cutting-plane method [38]. Since (4.6) only has $m$ decision variables and the constraints are extremely simple, solving (4.6) is much more efficient than solving (4.3) directly. In addition, it is proposed in [76] that the preconditioned conjugate gradient (PCG) method [27, 14] can be employed to compute $c_l$ with a proper choice of preconditioner. The stochastic preconditioning technique proposed in [62] is applied in [76] to generate the preconditioner using random walks [61].

The PCG method, combined with the dual algorithm to solve the LP problem (4.3), allows [76] to achieve significant speedups over previous works including [2] with-
out sacrificing the solution accuracy. However, as the number of global constraints increases, the performance of the dual algorithm degrades since the cutting-plane method becomes less efficient. Hence, we exploit the structure of the dual problem to design a dual bound algorithm for solving the LP problem (4.3), especially for large number of global constraints.

4.2 Dual Bound Algorithm

This section presents the dual bound algorithm, VNBound, to compute upper bound of voltage noise by formulating a reduced-size LP problem for each node. At first, Section 4.2.1 and 4.2.2 exploit the structure of the simplified dual problem by piecewise linear approximation. Then, Section 4.2.3 presents how to formulate the reduced-size LP problem, and Section 4.2.4 summarizes the overall algorithm. Finally, Section 4.2.5 presents experimental results.

4.2.1 An Equivalent Optimization Problem. Let $\mathcal{U}$ be the set of the distinct columns of $U$, i.e., $\mathcal{U} \triangleq \bigcup_{j=1}^{n}\{u_j\}$. Denote the elements of $\mathcal{U}$ as $\overline{u}_1$, $\overline{u}_2$, $\ldots$, and $\overline{u}_{|\mathcal{U}|}$. The duplicated columns of $U$ can be identified by introducing $|\mathcal{U}|$ index sets defined as $\mathcal{J}_k \triangleq \{j : u_j = \overline{u}_k\}, \forall 1 \leq k \leq |\mathcal{U}|$. Note that the sets $\mathcal{J}_k$ are a partition of $\{1, 2, \ldots, n\}$,

$$\mathcal{J}_k \cap \mathcal{J}_{k'} = \emptyset, \forall 1 \leq k \neq k' \leq |\mathcal{U}|, \text{ and } \bigcup_{k=1}^{|\mathcal{U}|} \mathcal{J}_k = \{1, 2, \ldots, n\}.$$

Then the simplified dual function $D(\gamma)$ can be formulated as follows.

$$D(\gamma) = \mathbf{I}_G^T \gamma + \sum_{j=1}^{n} I_{L,j} \max(0, c_{l,j} - u_j^T \gamma)$$

$$= \mathbf{I}_G^T \gamma + \sum_{k=1}^{|\mathcal{U}|} \sum_{j \in \mathcal{J}_k} I_{L,j} \max(0, c_{l,j} - \overline{u}_k^T \gamma).$$
Define a series of functions $f_k(x), \forall 1 \leq k \leq |\mathcal{U}|$ as

$$f_k(x) \triangleq \sum_{j \in \mathcal{J}_k} I_{L,j} \max(0, c_{l,j} - x), \forall x \geq 0, \quad (4.7)$$

then the optimization problem (4.6) can be rewritten as

$$\text{Minimize } D(\gamma) = \mathbf{1}_c^T \gamma + \sum_{k=1}^{\mathcal{U}} f_k(\mathbf{u}_k^T \gamma) \quad (4.8)$$

s.t. $\gamma \geq 0$.

### 4.2.2 Piecewise Linear Approximation of $f_k(x)$. Obviously, $f_k(x)$ is a continuous piecewise linear function with different gradients in different intervals. The gradients are determined by $I_{L,j}$’s, while the intervals are determined by $c_{l,j}$’s. According to the proof in Section IV-B of [78], the “max” operation is convex, so $f_k(x)$ is convex. A simple example of $f_k(x)$ is shown in Figure 4.1, where $x_{\text{min}}$ and $x_{\text{max}}$ are defined as

$$x_{\text{min}} \triangleq \min\{c_{l,j}, \forall j \in \mathcal{J}_k\}, \quad (4.9)$$

$$x_{\text{max}} \triangleq \max\{c_{l,j}, \forall j \in \mathcal{J}_k\}, \quad (4.10)$$

for any given $f_k(x)$. It has the maximum value at $x = 0$, and the minimum value 0 when $x \geq x_{\text{max}}$. The function value is monotonically decreasing in $[0, x_{\text{max}}]$. Although this function is straightforward, for real power grids, the number of its pieces is equal to the total number of nodes included in the same set of global constraints. Then it is difficult to solve either (4.6) or (4.8).

Fortunately, $f_k(x)$ can be approximated by a similar piecewise linear function with much smaller number of pieces. Let’s select $p + 1$ points $x_0, x_1, x_2, \ldots, x_p$ in $[0, x_{\text{max}}]$, and let $x_0 = 0, x_1 = x_{\text{min}}$ and $x_p = x_{\text{max}}$ such that these boundary points
Let \( b_k(x) \) be the piecewise linear approximation function of \( f_k(x) \). Its function value can be computed as

\[
b_k(x) = \begin{cases} 
  \frac{f_k(x_i)-f_k(x_{i-1})}{x_i-x_{i-1}} \times (x-x_{i-1}) + f_k(x_{i-1}), & \forall x \in [x_{i-1}, x_i), \forall 1 \leq i \leq p; \\
  0, & \forall x \geq x_p.
\end{cases}
\]

Let \( \tilde{g}_i, \forall 1 \leq i \leq p \) be the gradient of \( b_k(x) \) in \([x_{i-1}, x_i)\), and let \( \tilde{g}_{p+1} = 0 \) denote the gradient of \( b_k(x) \) when \( x \geq x_p \). Define \( B_i \) as

\[
B_i = \frac{\Delta}{\tilde{g}_i} - \frac{\Delta}{\tilde{g}_{i+1}}, \forall 1 \leq i \leq p.
\] (4.11)

Then \( b_k(x) \) can equivalently be formulated as

\[
b_k(x) = \sum_{i=1}^{p} B_i \max(0, x_i - x), \forall x \geq 0.
\] (4.12)
Clearly, this formulation of \( b_k(x) \) is similar to that of \( f_k(x) \) in (4.7), but has much smaller number of pieces if \( p \ll |J_k| \). The property of \( b_k(x) \) is summarized in the following lemma.

**Lemma 1** The piecewise linear approximation function \( b_k(x) \) is convex, and it satisfies

\[
b_k(x) \geq f_k(x), \quad \forall x \geq 0.
\]

**Proof 1** According to (4.12), and the fact that the “max” operation is convex, \( b_k(x) \) must be convex.

Consider \( b_k(x) \) in \([x_{i-1}, x_i], \forall 1 \leq i \leq p\), we have

\[
b_k(x) = \frac{f_k(x_i) - f_k(x_{i-1})}{x_i - x_{i-1}}(x - x_{i-1}) + f_k(x_{i-1})
\]

\[
= \frac{x - x_{i-1}}{x_i - x_{i-1}} f_k(x_i) + (1 - \frac{x - x_{i-1}}{x_i - x_{i-1}}) f_k(x_{i-1})
\]

\[
= tf_k(x_i) + (1 - t)f_k(x_{i-1}), \quad \text{where } t = \frac{x - x_{i-1}}{x_i - x_{i-1}}.
\]

Recall that \( f_k(x) \) is convex, so

\[
f_k(tx_i + (1 - t)x_{i-1}) \leq tf_k(x_i) + (1 - t)f_k(x_{i-1}) = b_k(x).
\]

As \( tx_i + (1 - t)x_{i-1} = x \), we get \( b_k(x) \geq f_k(x) \). When \( x \geq x_{\text{max}} \), \( b_k(x) = f_k(x) = 0 \). Therefore, \( b_k(x) \) is larger than or equal to \( f_k(x) \) due to the convexity of \( f_k(x) \).

**4.2.3 Reduced-size LP Problem.** Substituting \( b_k(x) \) for \( f_k(x) \) in (4.8) leads to the following optimization problem.

Minimize \( D(\gamma) = I_c^T \gamma + \sum_{k=1}^{|\mathcal{U}|} b_k(u_k^T \gamma) \) \hspace{1cm} (4.13)

s.t. \( \gamma \geq 0 \).
Theorem 2  The optimization problem (4.13) has an optimal solution, and its optimal value is no less than that of (4.8).

Proof 2  Because of the property of $b_k(x)$ as indicated by Lemma 1, Theorem 2 must hold.

Basically, (4.13) has the same structure as (4.8), since both $b_k(x)$ and $f_k(x)$ are piecewise linear functions with similar formulations. The only difference is that $b_k(x)$ has smaller number of pieces than $f_k(x)$. Recall that (4.8) has the same optimal value as the original LP problem (4.3). Similarly, one can create an LP problem corresponding to (4.13), and this LP problem has the same optimal value as (4.13). Details follow.

As shown in (4.12), for a given $1 \leq k \leq |\mathcal{U}|$, $b_k(x)$ is uniquely defined by three groups of parameters: $p$, $B_i$’s and $x_i$’s, where $1 \leq i \leq p$. For a given $b_k(x)$, denote its parameter $p$ by $p_k$, and let $B_k$ and $x_k$ be the vector of its $B_i$’s and $x_i$’s, respectively. Let $U_k$ be an $m \times p_k$ matrix, and each of its column is equal to $\mathbf{u}_k$. Define

$$\hat{n} \triangleq \sum_{k=1}^{\lfloor |\mathcal{U}| \rfloor} p_k, \quad (4.14)$$

$$\hat{\mathbf{B}} \triangleq \left( \mathbf{B}_1^T, \mathbf{B}_2^T, \cdots, \mathbf{B}_{\lfloor |\mathcal{U}| \rfloor}^T \right)^T, \quad (4.15)$$

$$\hat{\mathbf{c}} \triangleq \left( \mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_{\lfloor |\mathcal{U}| \rfloor}^T \right)^T, \quad (4.16)$$

$$\hat{\mathbf{U}} \triangleq \left( \mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_{\lfloor |\mathcal{U}| \rfloor} \right), \quad (4.17)$$

where $\hat{\mathbf{B}}$ and $\hat{\mathbf{c}}$ are $\hat{n} \times 1$ vectors, $\hat{\mathbf{U}}$ is an $m \times \hat{n}$ matrix. Then, the LP problem...
corresponding to (4.13) can be formulated as,

\[ \text{Maximize } \mathbf{c}^T \mathbf{i} \quad \text{s.t.} \]
\[ 0 \leq \mathbf{i} \leq \mathbf{B}, \mathbf{U} \mathbf{i} \leq \mathbf{I}_G. \]

where \( \mathbf{i} \) is an \( \hat{n} \times 1 \) vector of decision variables.

**Theorem 3** The LP problem (4.18) has an optimal solution. Its optimal value is the same as that of the optimization problem (4.13), and no less than that of (4.3) and (4.8).

**Proof 3** According to Theorem 2, and the fact that (4.8) and (4.13) are the dual problem of (4.3) and (4.18) respectively, Theorem 3 must hold.

The optimal value of the LP problem (4.18) is actually upper bound of voltage noise. The benefit of the formulation in (4.18), in comparison with the original LP problem (4.3), is that the number of decision variables can be significantly reduced by the piecewise linear approximation of \( f_k(x) \). With proper approximation, we can have \( \hat{n} \ll n \), thus solving (4.18) would take much less runtime than solving (4.3). (4.18) is called the reduced-size LP problem. As it only has a small number of decision variables, it can be solved efficiently by any LP solver.

**4.2.4 Algorithm Flow.** We design the VNBound algorithm as shown in Figure 4.2 to compute upper bound of voltage noise for the maxVN-LCC problem. For each node, this algorithm adopts the PCG method to compute \( c_i \), and formulates the reduced-size LP problem (4.18) to calculate voltage noise upper bound. The preprocessing steps on line 1 to 2 generate necessary data for the PCG method. Line 3 prepares necessary data for formulating the reduced-size LP problem. The loop on line 4 computes upper bound of voltage noise at each node.
Algorithm VNBound

Inputs
\( \delta_{inv} \): user-specified error tolerances.
\( gap \): user-specified parameter to control the problem size of (4.18).

Outputs
Upper bound of voltage noise at each node.

1. Compute the preconditioner \( M \) according to [62]
2. Solve \( Ax = I_L \) to obtain \( \Delta \)
3. Compute \( \mathcal{U} \) and \( \mathcal{J}_k, \forall 1 \leq k \leq |\mathcal{U}| \)
4. for \( l = 1 \) to \( n \)
5. Apply PCG method to compute \( c_l \) using the termination condition in (4.4)
6. Compute \( \hat{c} \) and \( \hat{B} \) to formulate (4.18)
7. Solve (4.18) using any LP solver
8. Report the upper bound of voltage noise at node \( l \) to be \( \hat{v}_l + v_l^+ \)

Figure 4.2. The VNBound algorithm.
**Theorem 4** Let $\hat{v}_l$ be the optimal value of (4.18) for node $l$. Then for every $1 \leq l \leq n$, $\overline{v}_l \leq \hat{v}_l + v_l^+.$

**Proof 4** According to Theorems 1 and 3, we have $\overline{v}_l \leq v_l^+ + v_l^+$, and $v_l^+ \leq \hat{v}_l$. Then Theorem 4 must hold.

We define the difference between the upper bound $\hat{v}_l + v_l^+$ and the exact voltage noise $\overline{v}_l$ as *overestimation*. As VNBound has no theoretical accuracy bound, its overestimation will be discussed in Section 4.2.5.2.

In our implementation, we adopt a user-specified parameter $gap$ to control the number of selected points for linear approximation, and $p_k$ is computed as

$$p_k = \left\lceil \frac{|J_k| - 1}{gap + 1} \right\rceil + 1, \forall 1 \leq k \leq |U|.$$  

Then $\hat{U}$ can be obtained according to (4.17). Recall that the number of variables is the summation of all $p_k$’s as defined in (4.14). $gap$ determines the problem size of the reduced-size LP problem (4.18). Typically, larger $gap$ leads to smaller problem size, but the number of variables will not decrease indefinitely. When $gap \geq |J_k| - 2$, we have $p_k = 2$ if $|J_k| > 1$; or $p_k = 1$ if $|J_k| = 1$. Therefore, there exists a minimum problem size of the reduced-size LP, which is dependent on $|J_k|$.

For a given $f_k(x)$, the points selected for linear approximation is computed as

$$x_i = \frac{i - 1}{p_k - 1} \times (x_{\text{max}} - x_{\text{min}}) + x_{\text{min}}, \forall 1 \leq i \leq p_k,$$

where $x_{\text{min}}$ and $x_{\text{max}}$ are defined in (4.9) and (4.10), respectively. Note that $x_0 = 0$. After all the points for approximation of $f_k(x)$, $\forall 1 \leq k \leq |U|$, have been derived, $\hat{c}$ is computed by (4.16). To calculate $B_k$’s using (4.11), one has to compute the gradient of $b_k(x)$ in different intervals, and the major computation is attributable to
the calculation of \( f_k(x_i) \) for all \( 0 \leq i \leq p_k \). Similar to the approach presented in [78] to compute \( D(\gamma) \) fast, one can sort the elements in \( J_k \) for a given \( c_l \) in order to compute \( f_k(x_i) \). As we only need to map the \( c_{l,j} \)'s to different intervals \([x_{i-1}, x_i]\) for computation, we apply bucket sort for fast evaluation of \( f_k(x_i) \).

4.2.5 Experiments.

4.2.5.1 Experimental Setup. The VNBound algorithm has been implemented in C++. For performance comparison, we also improve the implementation of the DualVD algorithm [76] and call it DualVN (Dual Voltage Noise). The LP problems \( \text{CP}(\mathcal{S}) \) in the cutting-plane method and the reduced-size LP problems in the VNBound Algorithm are solved by MOSEK [70], a general optimizing engine. For comparison, we replace the dual approach in the DualVN algorithm by MOSEK to solve (4.3) directly, and call this algorithm DirectVN hereafter. As MOSEK allows to choose between the simplex method and the interior point method to solve LP problems, we experiment with both options but only report the results using the simplex method as it is slightly faster. All the experiments are performed on a 64-bit Linux server with 2.4GHz Intel Q6600 processor and 8GB memory. Note that although the processor has multiple cores, only one core is used for experiments.

We generate 7 power grids using a setting of 4 metal layers, 1.2V VDD, and various C4 bumps/chip sizes/power consumptions. Moreover, we also experiment with IBM power grid benchmarks [54, 35]. The supply voltage of these IBM power grids is 1.8V. We only verify the ground network of these IBM power grids in our experiments, as their VDD networks can be decomposed into several non-connected partitions and verified separately. Note that these IBM power grids have some shorts, which are zero-value resistors and voltage sources. Before the power grid verification, these shorts are preprocessed at first, such that the shorted nodes are treated as a single node during verification process. We carry out four sets of experiments with 4,
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\delta_{\text{inv}}$</th>
<th>$\delta_{\text{lp}}$</th>
<th>Algorithm Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectVN</td>
<td>0.1mV</td>
<td>NA</td>
<td>0.1mV</td>
</tr>
<tr>
<td>DualVN</td>
<td>0.1mV</td>
<td>0.1mV</td>
<td>0.2mV</td>
</tr>
<tr>
<td>VNBound</td>
<td>0.1mV</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

**Figure 4.3.** Number of variables in the reduced-size LP problem of VNBound for two synthetic power grids pg4000 (90643 nodes) and pg10000 (562363 nodes) with 4 and 40 global constraints.

10, 20 and 40 global constraints, respectively.

**4.2.5.2 Algorithm Accuracy.** In the experiments, both error tolerances $\delta_{\text{inv}}$ and $\delta_{\text{lp}}$ are set to $1e^{-4}$, i.e. 0.1mV. Table 4.1 summarizes the accuracy settings of these algorithms. $\delta_{\text{lp}}$ is not applicable (NA) for DirectVN and VNBound because their LP problems (4.3) and (4.18) are directly solved by MOSEK. According to Theorem 1, the accuracy of DirectVN is given by $\delta_{\text{inv}}$, DualVN has an overall accuracy of $\delta_{\text{inv}} + \delta_{\text{lp}} = 0.2\text{mV}$, while VNBound has no theoretical accuracy bound.

Recall that the number of variables in the reduced-size LP problem of VNBound is dependent on the user-specified parameter $\text{gap}$. A set of experiments with
Figure 4.4. Cumulative distribution function of overestimation for synthetic power grid pg4000 (90643 nodes) with 40 global constraints, \( gap = 2000 \).

\( gap = 50, 100, 200, 500, 1000, 1500, 2000, 2500 \) and 3000 are conducted to evaluate the performance of VNBound for different power grids with various global constraint settings. Figure 4.3 shows the number of variables in the reduced-size LP problem of VNBound for four test cases of synthetic power grids. As \( gap \) increases, the number of variables decreases significantly at the beginning, and then gradually approaches the minimum problem size. Larger number of global constraints would typically result in larger problem sizes for a particular \( gap \), so the curves with 40 global constraints are above their counterparts with 4 global constraints.

A larger \( gap \) results in smaller number of variables in the reduced-size LP problem, thus achieving more speedups. However, a larger \( gap \) would lead to larger overestimation of the voltage noise. In practice, it is desired that the overestimation is sufficiently small, or at least small enough for most nodes in the power grid. Therefore, to evaluate the accuracy of VNBound, we define 99\% overestimation as

\[
\text{Probability}\left(\text{overest. of any node} \leq 99\% \text{ overest.}\right) = 99\% ,
\]
where “overest.” denotes overestimation. Figure 4.4 shows the cumulative distribution function of overestimation for a synthetic power grid with 90643 nodes and 40 global constraints using $gap = 2000$. The overestimation for most nodes is within in $5 \text{mV}$, and the 99% overestimation is much less than the maximum overestimation.

Figure 4.5 presents the 99% overestimation of VNBound for four test cases. Since verifying all the nodes in power grid pg10000 is too time-consuming, the results of pg10000 are estimated by verifying 10000 random nodes. Obviously, the 99% overestimation increases when $gap$ becomes larger. Note that the overestimation increases slowly when $gap$ is sufficiently large, that is due to the fact that the number of variables in the reduced-size LP problem gradually approaches the minimum problem size as illustrated in Figure 4.3. Except for $gap$, the 99% overestimation also depends on the power grid and constraints. A particular $gap$ leads to different 99% overestimation for different power grids and constraint settings.

In summary, increasing $gap$ reduces the LP problem size of VNBound, but also increases the overestimation of voltage noises. In practice, one need to select a
proper gap in order to explore the tradeoff between performance and accuracy, such that the reduced-size LP problem can be solved efficiently and the overestimation is acceptable. A number of experiments have been carried out to evaluate the proper gap for different power grids with various global constraint settings. It is observed that different test cases have different proper gap values. For example, as shown in Figure 4.5, if we want the 99% overestimation to be within 5mV, the proper gap of pg4000 should be smaller than 500, and that of pg10000 with 40 global constraints is about 1500. Since the 99% overestimation for pg10000 with 4 constraints is much smaller than 1mV, the proper gap can be even larger than 3000. In order to evaluate the performance of VNBound, we use a gap of 2000 for all the performance tests detailed in the next subsection. Although 2000 may not be the proper gap value of some test cases, it has reasonable overestimation for most power grids. This setup is employed to simplify the experimental settings.

4.2.5.3 Performance Results. For all the test cases, the voltage noises computed by DirectVN and DualVN are within its corresponding accuracy bound of the exact voltage noises. The voltage noise upper bounds computed by VNBound are no less than the exact voltage noises, and the overestimations are reasonably small.

The runtime comparison of DirectVN and DualVN with 4 global constraints is presented in Table 4.2. For IBM power grids, the reported number of nodes are the nodes in ground networks after preprocessing of shorts. In the experiments with 4 global constraints, the runtime of VNBound is approximately the same as that of DualVN, so the results of VNBound are excluded from the table. The runtime can be generally partitioned into two parts: the runtime to compute $c_l$ and the runtime to solve the LP problem (4.3) (or more precisely (4.6) for the DualVN algorithm, and (4.18) for the VNBound algorithm). These three algorithms have approximately the same runtime for the former as they share the same code to compute $c_l$. Since the lat-
ter is our major contribution, we report the runtime for the latter under the columns “LP”, and report the total runtime under the columns “Total”. The time units are abbreviated such that “s”, “m”, “h” and “d” denote “seconds”, “minutes”, “hours”, and “days” respectively. The speedups of DualVN in comparison with DirectVN are shown in the last two columns. It can be seen that the proposed dual approach drastically reduces the runtime to solve the LP problem such that the DualVN algorithm can achieve significant speedups over the DirectVN algorithm. As some test cases are too time-consuming, the reported results with ≈’s are estimations from the results of 10000 nodes chosen randomly.

Table 4.3 shows the runtime comparison of DirectVN and VNBound with 40 global constraints. In this set of experiments, although the proposed dual approach still accelerates solving the LP problem for large power grids, the performance of DualVN is inferior in comparison with that of VNBound. The number of variables in the reduce-size LP problem (4.18) of the VNBound Algorithm is shown under the column “Var.”. Obviously, VNBound significantly reduces the number of variables in the LP problem, and largely accelerates the overall power grid verification. The 99% overestimation is much less than the maximum voltage noise of the power grid, so VNBound is relatively accurate for fast evaluation of voltage noises. Note that the speedups in solving the LP problem are much less than the times of variable reduction because formulating the reduced-size LP problem by piecewise linear approximation takes extra runtime.

Figure 4.6 and 4.7 show the speedup of DualVN and VNBound in solving the LP problem, respectively. Note that these figures include the results of both synthetic power grids and IBM power grid benchmarks. The X-axis shows the approximated number of nodes in the power grid. Clearly, when there are only 4 global constraints, the speedup of DualVN is approximately the same as that of VNBound. As the num-
Table 4.2. Runtime Comparison of DirectVN and DualVN with 4 Global Constraints

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>DirectVN ($\delta_{\text{inv}} = 0.1\text{mV}$)</th>
<th>DualVN ($\delta_{\text{inv}} = \delta_{\text{lp}} = 0.1\text{mV}$)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Max Voltage Noise (mV)</td>
<td>LP</td>
<td>Total</td>
</tr>
<tr>
<td>pg1000</td>
<td>5875</td>
<td>46.31</td>
<td>2.77 m</td>
</tr>
<tr>
<td>pg2000</td>
<td>22939</td>
<td>39.91</td>
<td>45.95 m</td>
</tr>
<tr>
<td>pg2500</td>
<td>35668</td>
<td>28.81</td>
<td>1.99 h</td>
</tr>
<tr>
<td>pg3000</td>
<td>51195</td>
<td>43.66</td>
<td>4.15 h</td>
</tr>
<tr>
<td>pg4000</td>
<td>90643</td>
<td>54.45</td>
<td>11.46 h</td>
</tr>
<tr>
<td>pg5000</td>
<td>141283</td>
<td>46.11</td>
<td>$\approx 1.06$ d</td>
</tr>
<tr>
<td>pg10000</td>
<td>562363</td>
<td>24.56</td>
<td>$\approx 12.30$ d</td>
</tr>
<tr>
<td>ibmpg1</td>
<td>10242</td>
<td>677.67</td>
<td>5.73 m</td>
</tr>
<tr>
<td>ibmpg2</td>
<td>65228</td>
<td>357.93</td>
<td>4.77 h</td>
</tr>
<tr>
<td>ibmpg3</td>
<td>150687</td>
<td>180.55</td>
<td>$\approx 1.14$ d</td>
</tr>
<tr>
<td>ibmpg4</td>
<td>478094</td>
<td>3.51</td>
<td>$\approx 9.40$ d</td>
</tr>
<tr>
<td>ibmpg5</td>
<td>291382</td>
<td>42.70</td>
<td>$\approx 8.77$ d</td>
</tr>
<tr>
<td>ibmpg6</td>
<td>430337</td>
<td>110.00</td>
<td>$\approx 14.11$ d</td>
</tr>
</tbody>
</table>
Table 4.3. Runtime Comparison of DirectVN and VNBound with 40 Global Constraints

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>DirectVN ($\delta_{\text{inv}} = 0.1\text{mV}$)</th>
<th>VNBound ($\delta_{\text{inv}} = 0.1\text{mV}$, $gap = 2000$)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LP</td>
<td>Total</td>
<td>Var.</td>
</tr>
<tr>
<td>pg1000</td>
<td>5875</td>
<td>52.65</td>
<td>3.39 m</td>
</tr>
<tr>
<td>pg2000</td>
<td>22939</td>
<td>38.95</td>
<td>1.32 h</td>
</tr>
<tr>
<td>pg2500</td>
<td>35668</td>
<td>28.06</td>
<td>2.84 h</td>
</tr>
<tr>
<td>pg3000</td>
<td>51195</td>
<td>43.75</td>
<td>5.72 h</td>
</tr>
<tr>
<td>pg4000</td>
<td>90643</td>
<td>49.41</td>
<td>16.15 h</td>
</tr>
<tr>
<td>pg5000</td>
<td>141283</td>
<td>44.55</td>
<td>$\approx 1.50$ d</td>
</tr>
<tr>
<td>pg10000</td>
<td>562363</td>
<td>24.17</td>
<td>$\approx 14.48$ d</td>
</tr>
<tr>
<td>ibmpg1</td>
<td>10242</td>
<td>658.50</td>
<td>6.92 m</td>
</tr>
<tr>
<td>ibmpg2</td>
<td>65228</td>
<td>270.79</td>
<td>5.86 h</td>
</tr>
<tr>
<td>ibmpg3</td>
<td>150687</td>
<td>165.63</td>
<td>$\approx 1.48$ d</td>
</tr>
<tr>
<td>ibmpg4</td>
<td>478094</td>
<td>3.43</td>
<td>$\approx 11.17$ d</td>
</tr>
<tr>
<td>ibmpg5</td>
<td>291382</td>
<td>37.93</td>
<td>$\approx 12.27$ d</td>
</tr>
<tr>
<td>ibmpg6</td>
<td>430337</td>
<td>102.44</td>
<td>$\approx 18.70$ d</td>
</tr>
</tbody>
</table>
Figure 4.6. Speedup of DualVN relative to DirectVN in solving the LP problem.

As the number of global constraints increases, the performance of the DualVN algorithm decreases. When there are 40 global constraints, the maximum speedup of DualVN is less than 50. This performance degradation is attributable to the fact that the cutting-plane method becomes less effective when the dimension of space increases. In our experiments, the cutting-plane method typically terminates within a few iterations for 4 global constraints, while it takes dozens of iterations for 10 and 20 global constraints, and even consumes more than 100 iterations for 40 global constraints.

For the VNBound algorithm, both the number of variables in the reduced-size LP problem and the LP runtime increase monotonically as the number of global constraints, and so does the LP runtime of DirectVN. The LP speedup shown in Figure 4.7 does not change monotonically because it is the ratio of two monotonically increasing runtime metrics. Although the increasing number of variables in the reduced-size LP problem results in minor speedup degradation for some power grids, e.g. the power grids with less than 100k nodes, the speedup of VNBound is always substantial even with 40 global constraints. As shown in Figure 4.8, VNBound achieves much more speedups than DualVN in solving the LP problem when there
Figure 4.7. Speedup of VNBound relative to DirectVN in solving the LP problem.

are 40 global constraints. However, DualVN is fairly efficient and more accurate when there are only a few global constraints, e.g. 4 global constraints.

The experimental results also show that the random-walk based preconditioner computation is fairly efficient, taking runtime ranging from a few seconds to a few minutes as the power grids become larger. For synthetic power grids, it takes about 11 PCG iterations on average to compute each $c_l$. For IBM power grids, the average number of PCG iterations ranges from 9 to 15. This is attributable to the fact that IBM power grids have different structures and thus their system matrixes have different condition numbers. Although the PCG method converges fast, it can be implied from Tables 4.2 and 4.3 that to compute $c_l$ by the PCG method may require more than 95% of runtime for large power grids using DualVN or VNBound. We view such observation as a chance to further speedup the DualVN and VNBound algorithm by leveraging more advanced power grid analysis techniques. In addition, according to (4.4) and Theorem 1, the PCG runtime can be reduced by using a larger $\delta_{inv}$, which makes a trade-off between solution quality and runtime.

In summary, both the DualVN algorithm and the VNBound algorithm accel-
Figure 4.8. Speedup of VNBound relative to DualVN in solving the LP problem.

erate solving the LP problem, reduce the total runtime per node significantly, and make vectorless verification of large power grids practical.

4.3 Hierarchical Matrix Inversion

In this section, we propose a hierarchical matrix inversion algorithm to speed up computing $c_l$. Section 4.3.1 introduces the motivation of our algorithm, Section 4.3.2 present the technical details, and results are shown in Section 4.3.3.

4.3.1 Motivation. Since the LP runtime has been largely reduced by the proposed dual algorithms, now most runtime is spent on the PCG method to compute $c_l$. Therefore, any further reduction in running time must address the problem of computing $c_l$'s more efficiently than the PCG method, which is clearly non-trivial given the large number of nodes in power grids. According to (4.2) and the fact that $A$ is a symmetric M-matrix, $c_l$ is the $l$’th column of the inverse matrix $A^{-1}$. As $c_l$ is computed for all nodes, eventually a whole inverse matrix $A^{-1}$ is computed. To accelerate computing $c_l$, one must be able to explore the structure of the power grid, as the voltage drops of a node and its neighboring nodes are closely related.
Let’s fist consider a node $l$ in the power grid for DC analysis model, as illustrated in Figure 4.9. Applying Kirchhoff’s Current Law (KCL), we have

$$\sum_{\forall k \in \mathcal{N}(l)} g_{l,k} (v_l - v_k) = i_l, \quad (4.19)$$

where $\mathcal{N}(l)$ is the set of node $l$’s neighboring nodes, $g_{l,k}$ is the conductance between node $l$ and $k$, $v_l$ and $v_k$ are the voltage drop at node $l$ and $k$ respectively, and $i_l$ is the current source attached to node $l$. Define $g_l \triangleq \sum_{\forall k \in \mathcal{N}(l)} g_{l,k}$, then (4.19) can be rearranged into

$$v_l = \frac{1}{g_l} (i_l + \sum_{\forall k \in \mathcal{N}(l)} g_{l,k} v_k). \quad (4.20)$$

Recall that $v_l = e_l^T i$, $v_k = e_k^T i$ and $i_l = e_l^T i$. Equation (4.20) is equivalent to

$$c_l^T i = \frac{1}{g_l} (e_l^T i + \sum_{\forall k \in \mathcal{N}(l)} g_{l,k} c_k^T i)$$

$$= \left( \frac{1}{g_l} (e_l + \sum_{\forall k \in \mathcal{N}(l)} g_{l,k} e_k) \right)^T i. \quad (4.21)$$
Equation (4.21) holds for any current vector $\mathbf{i}$, therefore,

$$
\mathbf{c}_l = \frac{1}{g_l} (\mathbf{e}_l + \sum_{k\in N(l)} g_{l,k} \mathbf{c}_k).
$$

(4.22)

According to (4.22), one can directly compute the $\mathbf{c}_l$ of a node by using the $\mathbf{c}_l$ of its neighboring nodes. Due to the structure of the power grid, a node only has a few number of neighboring nodes, so computing $\mathbf{c}_l$ using (4.22) consumes much less runtime than solving $\mathbf{c}_l$ with the PCG method. In order to apply (4.22) to speed up computing $\mathbf{c}_l$, one shall first use the PCG method to solve the $\mathbf{c}_l$ of the nodes, which can cover all the edges in the power grid, then employ (4.22) to compute the $\mathbf{c}_l$ of the other nodes. Note that finding the nodes that need to be solved by the PCG method is a vertex cover problem. Our preliminary experimental results show that only about 35%-38% of the nodes’ $\mathbf{c}_l$ can be computed by using (4.22). Intuitively, replacing node $l$ by a subset of nodes in Figure 4.9, if one can formulate the $\mathbf{c}_l$ of these nodes similarly, then the $\mathbf{c}_l$ of more nodes can be computed using this approach. Motivated by the desire to speed up computing $\mathbf{c}_l$ to the most limit, we explore this method to compute the $\mathbf{c}_l$ of a subset of nodes.

Our major contribution in this section is an efficient hierarchical approach to compute all the $\mathbf{c}_l$. We exploit the structure of the power grid to formulate the $\mathbf{c}_l$ of a subset of nodes in terms of their external neighbor nodes. We propose to partition the power grid into a set of external neighbor nodes and a number of partitions using the hypergraph partitioning technique, and compute the $\mathbf{c}_l$ of the nodes within each partition using the $\mathbf{c}_l$ of these external neighbor nodes computed by the PCG method [76]. We design the HierarchicalVN (Hierarchical Voltage Noise) algorithm to solve the maxVN-LCC problem efficiently using our hierarchical approach to compute $\mathbf{c}_l$, and the dual approach of [76] to solve (4.3). The details are presented in the next
4.3.2 Proposed Approach.

4.3.2.1 Computing $c_l$ for a Subset of Nodes. Consider a subset of nodes in the power grid as illustrated in Fig. 4.10, let’s call the nodes within this subset internal nodes, and these internal nodes’ neighbors which are not included in this subset external neighbors. In other words, for any internal node of a subset, any of its neighbors is either an internal node or an external neighbor of this subset.

Let’s first consider the DC analysis model, where $A = G$ is the conductance matrix. Let $n'$ and $m'$ be the number of internal nodes and external neighbors of a
subset respectively. Applying KCL for all the internal nodes of the subset, we have

\[
\begin{bmatrix}
G_{in} & G_{ex}
\end{bmatrix}
\begin{bmatrix}
v_{in} \\
v_{ex}
\end{bmatrix} = i_{in},
\]

(4.23)

where \(G_{in} \in \mathbb{R}^{n' \times n'}\) is the conductance matrix of the internal nodes, \(G_{ex} \in \mathbb{R}^{n' \times m'}\) is the conductance links between internal nodes and external neighbors, \(v_{in}\) and \(v_{ex}\) are the voltage drop vector of the internal nodes and external neighbors respectively, and \(i_{in}\) is the current source vector of the internal nodes. Rearrange (4.23), we get

\[
v_{in} = G_{in}^{-1}(i_{in} - G_{ex}v_{ex}).
\]

(4.24)

Introduce \(C_{in} \in \mathbb{R}^{n' \times n}, C_{ex} \in \mathbb{R}^{m' \times n}\) and \(E_{in} \in \mathbb{R}^{n' \times n}\). Let each row of \(C_{in}\) be the corresponding row of each internal node in \(A^{-1}\), each row of \(C_{ex}\) be the corresponding row of each external neighbor node in \(A^{-1}\), and each row of \(E_{in}\) be the \(e_l^T\) of the corresponding internal node. Note that the \(l\)’th row of \(A^{-1}\) is equal to the transpose of the \(l\)’th column \(c_l\), because \(A^{-1}\) is symmetric. We have \(v_{in} = C_{in}i, v_{ex} = C_{ex}i,\) and \(i_{in} = E_{in}i\), then (4.24) can be rewritten as

\[
C_{in}i = G_{in}^{-1}(E_{in}i - G_{ex}C_{ex}i)
\]

\[
= (G_{in}^{-1}(E_{in} - G_{ex}C_{ex}))i.
\]

(4.25)

Equation (4.25) holds for any current vector \(i\), so

\[
C_{in} = G_{in}^{-1}(E_{in} - G_{ex}C_{ex}).
\]

(4.26)

For the transient analysis model, \(A = G + \frac{C}{\Delta t}\). One can create an *adjusted power grid* by attaching a VDD pad to each non-VDD node and setting the conductances
of these newly added edges according to the diagonal components of \( \frac{G}{\Delta t} \). Then \( A \) is exactly the conductance matrix of the adjusted power grid. If one constructs \( G_{in} \) and \( G_{ex} \) for this adjusted power grid, then (4.26) still holds. In summary, for either DC or transient analysis, one can always treat \( A \) as a conductance matrix, create the corresponding \( G_{in} \) and \( G_{ex} \) according to \( A \), and derive (4.26). Let \( A_{in} \) and \( A_{ex} \) denote the \( G_{in} \) and \( G_{ex} \) constructed according to the power grid conductance matrix \( A \), then we have the following lemma.

**Lemma 2**  \( C_{in} = A_{in}^{-1}(E_{in} - A_{ex}C_{ex}) \).

Obviously, the \( c_i \) of a subset of nodes \( (C_{in}) \) can be computed by using the \( c_i \) of their external neighbors \( (C_{ex}) \). Consider the nodes within a sub-block of the power grid as such a subset. If all the \( c_i \) of its external neighbors have been solved, one can directly compute the \( c_i \) of its internal nodes. As we will show in the next subsection, the power grid can be divided into a set of external neighbors and a number of partitions. Each partition is a subset of nodes, and it is almost always a sub-block of the power grid. Using such a power grid partitioning technique, only a small amount of nodes are external neighbors, and most of the nodes are internal nodes of partitions. Then, one only need to compute a small amount of \( c_i \)'s corresponding to the external neighbors independently with the PCG method, and compute most \( c_i \)'s corresponding to the internal nodes using Lemma 2.

The computation in Lemma 2 can be made more efficient by exploiting the properties of these matrices. Let’s first consider their density, \( A_{in}^{-1} \in \mathbb{R}^{n' \times n'} \) and \( C_{ex} \in \mathbb{R}^{m' \times n} \) are dense, \( E_{in} \in \mathbb{R}^{n' \times n} \) is a sparse 0/1 matrix with only \( n' \) 1s, and \( A_{ex} \in \mathbb{R}^{n' \times m'} \) is also sparse since only a few internal nodes are connected with external neighbors. Because of the special characteristic of \( E_{in} \), the computation associated with \( E_{in} \) is negligible. Most time complexity is attributable to computing \( A_{in}^{-1}A_{ex}C_{ex} \).
Recall that the power grid is partitioned into a number of partitions. We observe that the number of internal nodes in a partition is usually larger than the number of its external neighbors in our preliminary experiments. In other words, we usually have \( n \gg n' > m' \). Hence, computing \( A_{in}^{-1}A_{ex} \) first consumes less runtime and memory than computing \( A_{ex}C_{ex} \) first in order to solve \( A_{in}^{-1}A_{ex}C_{ex} \), then Lemma 2 can be rearranged into

\[
C_{in} = (A_{in}^{-1}(-A_{ex}))C_{ex} + A_{in}^{-1}E_{in},
\]

(4.27)

By keeping the number of internal nodes in the subset within some bound, \( A_{in}^{-1} \) can be solved by LU factorization together with a forward solve and a backward solve.

The time complexity of the PCG method to compute a single \( c_l \) is \( O(KM) \) where \( K \) is the number of iterations toward convergence and \( M \) is the number of non-zero elements in the preconditioner. To apply (4.27) to a single partition, if \( n' \) is reasonably small, the most time-consuming computation is the dense matrix-matrix multiplication of \( A_{in}^{-1}A_{ex} \) and \( C_{ex} \), which takes \( O(m'n'n) \) time. Let \( \rho \in [0,1] \) be the ratio of internal nodes to \( n \). Assume that each partition has roughly the same size, the overall time complexity will be,

\[
O(m'n'n) \times \frac{pm}{n'} + O(KM)(1 - \rho)n
= \rho O(m'n^2) + (1 - \rho)O(KMn).
\]

In other words, for each internal node, we replace the PCG solution that takes \( O(KM) \) time by a vector-matrix multiplication that takes \( O(m'n) \) time, which will result in a significant reduction of running time if the parameters are chosen properly. Clearly, if \( n' \) is too small, then \( \rho \) is small. In Section III, we show when \( n' = 1, \rho \approx 0.35 \) to 0.38. As \( n' \) increases, \( \rho \) increases and the overall complexity may decrease depending on the changes in \( m' \). In addition, \( n' \) should be kept small enough such that the computation
of $A_{in}^{-1}$ will not dominate that of (4.27). Basically, there is a trade-off between the increasing partition size and the increasing time complexity for each internal node, and we will demonstrate it in Section 4.3.3.

4.3.2.2 Power Grid Partitioning. In order to apply Lemma 2 to compute $c_l$, one has to find a subset of nodes as external neighbors, which split the power grid into a number of non-connected partitions. Moreover, it is also desired that the size of each partition is in some user-controllable bound, so that the $c_l$ of the nodes in these partitions can be computed efficiently according to (4.27). Fig. 4.11 shows an example of such a desired partitioned power grid. For simplicity, there are only a total of 36 external neighbors and four partitions, each of which has 9 or 10 internal nodes. Our power grid partitioning problem can be formulated mathematically as follows.

**Problem 2 (PG-Partition)** Let $V$ be the set of all the non-VDD nodes in the power
grid, $\mathcal{E}$ be the set of all the edges connecting these nodes, rps be a user-specified rough partition size, npart \((npart \gg 1)\) be the desired number of partitions corresponding to rps. Let $\mathcal{P}_k$ and $\mathcal{N}(\mathcal{P}_k)$ be the set of internal nodes and external neighbors of the $k$'th partition respectively, $\epsilon$ be a user-specified imbalance bound of the partition sizes. Find a minimum set $\mathcal{S}$ of external neighbor nodes, which partition the power grid $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into npart mutually disjoint partitions $\mathcal{P}_k \ (1 \leq k \leq npart)$, such that

$$
\mathcal{V} = \mathcal{S} \bigcup_{k=1}^{npart} \bigcup_{k=1}^{npart} \mathcal{P}_k, \quad \mathcal{S} = \bigcup_{k=1}^{npart} \mathcal{N}(\mathcal{P}_k),
$$

$$
\mathcal{S} \bigcap \mathcal{P}_k = \emptyset, \forall 1 \leq k \leq npart,
$$

$$
\mathcal{P}_j \bigcap \mathcal{P}_k = \emptyset, \forall 1 \leq j \neq k \leq npart,
$$

$$
|\mathcal{P}_k| \leq \bar{P}, \forall 1 \leq k \leq npart,
$$

where $\bar{P}$ is a partition size upper bound determined by rps and $\epsilon$.

We propose to solve this problem by introducing a hypergraph model of the power grid, and performing hypergraph partitioning on it. Let $\mathcal{V}_h$ and $\mathcal{E}_h$ be the set of vertices and hyperedges in the hypergraph model respectively. Define

$$
\mathcal{V}_h \triangleq \mathcal{E}, \mathcal{E}_h \triangleq \mathcal{V}.
$$

Then $(\mathcal{V}_h, \mathcal{E}_h)$ is the hypergraph model. The vertices in $\mathcal{V}_h$ represent the edges in the power grid, and the hyperedges in $\mathcal{E}_h$ represent the nodes. In conventional hypergraphs, each vertex has a specific area, and each hyperedge has a specific weight. Hypergraph partitioning algorithms typically partition the hypergraph by removing some hyperedges (also called hyperedge-cuts) to get isolated partitions of the hypergraph. The objective is to minimize the total weight of hyperedge-cuts while keeping the area of different partitions being balanced. For this hypergraph model,
the hyperedge-cuts represent the external neighbors of the power grid, the hyperedges in each partition represent the internal nodes, and the hyperedge-cuts associated with each partition represent the external neighbors of that partition.

In order to apply the hypergraph partitioning algorithms to solve Problem 2, we define the area of each vertex and the weight of each hyperedge in the hypergraph model as follows. Let $a(j)$ be the area of the $j$'th vertex in $V_h$, $H_{j,1}$ and $H_{j,2}$ be the two hyperedges that include vertex $j$, $w(k)$ be weight of the $k$'th hyperedge in $E_h$. Define

$$a(j) \triangleq \frac{1}{|H_{j,1}|} + \frac{1}{|H_{j,2}|}, \forall 1 \leq j \leq |V_h|,$$  \hspace{1cm} (4.29)

$$w(k) \triangleq 1, \forall 1 \leq k \leq |E_h|.$$

Then it can be proved that there exists some partition size upper bound $P$, which is determined by $rps$ and $\epsilon$.

Let $P^h_k (1 \leq k \leq npart)$ be the set of vertices in the $k$'th partition of the hypergraph model, $a(P^h_k)$ be the total area of the $k$'th partition. Define

$$a(P^h_k) \triangleq \sum_{\forall \text{ vertex } j \in P^h_k} a(j).$$

With this kind of area definition, the area of each partition is larger than the number of hyperedges within it, because the associated hyperedge-cuts of a partition also contribute to its area. For example, let’s look at any partition with 9 internal nodes as shown in Fig. 4.11. It has 24 edges, 9 nodes, and 12 external neighbors. In its corresponding hypergraph model, it has 24 vertices, 9 hyperedges and 12 associated hyperedge-cuts. The area of each vertex in this partition is $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. Then the total area of this partition is $24 \times \frac{1}{2} = 12 > 9$. As each hyperedge in a partition represents an internal node within that partition, the number of hyperedges in the $k$'th partition
is equal to $|\mathcal{P}_k|$, so we have the following lemma.

**Lemma 3** For every $1 \leq k \leq \text{npart}$, $|\mathcal{P}_k| < a(\mathcal{P}_k^h)$.

Define the average area of all the partitions

$$\overline{a_P} \triangleq \frac{\sum_{k=1}^{\text{npart}} a(\mathcal{P}_k^h)}{\text{npart}} = \frac{\sum_{\forall \text{ vertex } j \in \mathcal{V}_h} a(j)}{\text{npart}} = \frac{\left| \mathcal{E}_k \right|}{\text{npart}} = \frac{\left| \mathcal{V} \right|}{\text{npart}} = \frac{n}{\text{npart}}.$$  

In order to balance the area of different partitions, hypergraph partitioning algorithms typically keep the imbalance of $a(\mathcal{P}_k^h)$ among all partitions within the user-specified imbalance bound $\epsilon$. We have

$$\frac{|a(\mathcal{P}_k^h) - \overline{a_P}|}{\overline{a_P}} \leq \epsilon, \; \forall 1 \leq k \leq \text{npart}.$$  

Therefore,

$$a(\mathcal{P}_k^h) \leq (1 + \epsilon) \cdot \overline{a_P} = (1 + \epsilon) \cdot \frac{n}{\text{npart}}. \quad (4.31)$$

Combine Lemma 3 with (4.31), we get

$$|\mathcal{P}_k| < (1 + \epsilon) \cdot \frac{n}{\text{npart}}.$$  

Define

$$\text{npart} \triangleq \left\lceil \frac{n}{rps} \right\rceil, \quad (4.32)$$

then

$$|\mathcal{P}_k| < (1 + \epsilon) \cdot \frac{n}{\text{npart}} \leq (1 + \epsilon) \cdot rps,$$  

and we have the following lemma.
Lemma 4 Define $\mathcal{P} \triangleq (1 + \epsilon) \cdot rps$, then for every $1 \leq k \leq \text{npart}$, $|\mathcal{P}_k| < \mathcal{P}$.

Clearly, the vertex area definition guarantees that the number of internal nodes in each partition ($|\mathcal{P}_k|$) is bounded by $\mathcal{P}$. One can adjust the size of partitions by using different $rps$. As the weight of each hyperedge in the hypergraph model is set to 1, the hypergraph partitioning algorithms will minimize the number of hyperedge-cuts, thus deriving a minimum set of external neighbors. After the hypergraph model is partitioned, the partitioning result can be mapped back to the original power grid to derive $\mathcal{S}$, $\mathcal{P}_k$, and $\mathcal{N}(\mathcal{P}_k)$, since $\mathcal{S}$ is the set of nodes corresponding to the hyperedge-cuts, $\mathcal{P}_k$ is the set of nodes corresponding to the hyperedges with in the $k$'th partition, and $\mathcal{N}(\mathcal{P}_k)$ is the set of nodes corresponding to the associated hyperedge-cuts of the $k$'th partition.

4.3.2.3 The HierarchicalVN Algorithm. Combining our hierarchical approach to compute $c_l$, the PCG method and the dual approach in [76], we design the HierarchicalVN algorithm to solve the $\text{maxVN-LCC}$ problem as illustrated in Fig. 4.12. Two user-specified error-tolerances $\delta_{\text{inv}}$ and $\delta_{\text{lp}}$ are employed to control the accuracy of the solution of (4.2) and (4.3) respectively. In this algorithm, the power grid is partitioned at first, then the partitions are verified sequentially. The verification of a partition has two steps. First, verify its un-solved external neighbors and store the computed $c_l$ of these external neighbors in memory. Second, solve its internal nodes, where the $c_l$ of these internal nodes are computed by (4.27). As all the $c_l$ of external neighbors are computed by the PCG method with error-tolerance $\delta_{\text{inv}}$, thus not being exact, one can only get an approximated $c_l$ of each internal node using (4.27). In order to guarantee the accuracy of the solution, we check whether the computed $c_l$ is acceptable, and use the PCG method to refine it if necessary.

4.3.2.4 Practical Memory Management. In order to apply the HierarchicalVN algorithm, there must be at least enough memory to store the external neighbors’
**Algorithm** HierarchicalVN

**Inputs**


\( rps, \epsilon \): as specified in Problem 2.

\( \delta_{inv}, \delta_{lp} \): user-specified error-tolerances.

**Outputs**

Maximum voltage noises at each node.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create the hypergraph model according to (4.28), (4.29) and (4.30)</td>
</tr>
<tr>
<td>2</td>
<td>Compute ( npart ) according to (4.32)</td>
</tr>
<tr>
<td>3</td>
<td>Partition the hypergraph model into ( npart ) partitions to derive ( S, P_k, N(P_k) ) (( 1 \leq k \leq npart )) as defined in Problem 2</td>
</tr>
<tr>
<td>4</td>
<td>for ( k = 1 ) to ( npart )</td>
</tr>
<tr>
<td>5</td>
<td>for all un-solved external neighbors in ( N(P_k) )</td>
</tr>
<tr>
<td>6</td>
<td>Apply the PCG method in [76] to compute ( c_I )</td>
</tr>
<tr>
<td>7</td>
<td>Apply the dual approach in [76] to solve (4.3)</td>
</tr>
<tr>
<td>8</td>
<td>Report the maximum voltage noise</td>
</tr>
<tr>
<td>9</td>
<td>for all internal nodes in ( P_k )</td>
</tr>
<tr>
<td>10</td>
<td>Apply (4.27) to compute an approximated ( c_I )</td>
</tr>
<tr>
<td>11</td>
<td>Check whether the approximated ( c_I ) satisfies (4.4), if not, use the PCG Method to refine the approximated ( c_I ) until it satisfies (4.4)</td>
</tr>
<tr>
<td>12</td>
<td>Apply the dual approach in [76] to solve (4.3)</td>
</tr>
<tr>
<td>13</td>
<td>Report the maximum voltage noise</td>
</tr>
</tbody>
</table>

Figure 4.12. The HierarchicalVN Algorithm.
rows $C_{ex}$ and the partial inversion $A_{in}^{-1}$ of any partition. With limited amount of memory available in the computer, one must keep the number of external neighbors and internal nodes for any partition within some bound, so that $C_{ex}$ and $A_{in}^{-1}$ can be stored. Recall that $n'$ and $m'$ denote the number of internal nodes and the number of external neighbors for a partition respectively. In our experiments, we observe that $rps \geq n' > m'$ almost always. Hence, we adopt rough partition size $rps$ from a few dozens to one thousand, such that $C_{ex}$ and $A_{in}^{-1}$ can easily be buffered. Since we verify the internal nodes one by one, there is no need to store the internal nodes’ rows $C_{in}$.

As illustrated in Fig. 4.11, a node can be the external neighbor of two or more partitions, then all these associated partitions will use its $c_l$ to compute the approximate $c_l$ of internal nodes. For such an external neighbor node, once the $c_l$ is computed, one need to keep it in memory until all of its associated partitions have been verified. However, this approach can be problematic. Consider the worst-case scenario, if the first partition and the last partition share a lot of external neighbors, then the $c_l$ of their common external neighbors need to be stored in memory for the whole duration of the power grid verification. Although such scenario seldom happens, one usually need to keep a number of useful $c_l$ for the un-verified partitions, but the available memory in the computer may not be sufficient to store all of the useful $c_l$. Thus it is desired that the partitions of the power grid have good locality, such that the neighboring partitions, which often share some external neighbors, can be verified by the HierarchicalVN algorithm in a small time frame, thus minimizing the number of useful $c_l$ that one need to buffer at runtime.

In our implementation, we use a fixed-size buffer to store the computed $c_l$ of external neighbors, and assign a weight for each buffered $c_l$ to be the number of un-verified partitions that will use it for computation. If we have to store a new $c_l$ when the buffer is full, then the $c_l$ with the minimum weight is freed. After finishing the
verification of a partition, we decrease the weight of the buffered \( c_l \) for its external neighbors by 1. A zero weight \( c_l \) implies that no partition will use it for computation, so any buffered \( c_l \) is freed as soon as its weight is decreased to 0. Using this scheme, if any useful \( c_l \) is freed when the buffer is full, we need to re-compute it when we need it for computation.

In our experiments, we set the buffer size to be 1000 \( c_l \), and employ the hypergraph partitioning tool hMETIS [69] to partition the hypergraph model. Results with a rough partition size \( rps = 100 \) for different power grids show that the maximum number of buffered useful \( c_l \) is usually less than 800, and no \( c_l \) is re-computed, except for the largest power grid of 562K non-VDD nodes. When verifying the largest grid, although the buffer is fully utilized, only about 5% of the external neighbors’ \( c_l \) are re-computed. Therefore, it is shown that the power grid partitions generated by hMETIS have the desired locality, and there is no need to find a order to verify these partitions with small memory requirement.

4.3.3 Experimental Results. We implement our HierarchicalVN algorithm and the DualVD algorithm [76] for comparison in C++. Both algorithms share the same routine to compute the stochastic preconditioner, to perform PCG iterations to compute \( c_l \) when necessary, and to solve (4.3) (using MOSEK [70] as the LP solver for the cutting-plane method). (Note that the DualVD algorithm [76] is actually the initial implementation of the DualVN algorithm in [78], its performance is slightly worse than that of DualVN. We design the HierarchicalVN algorithm based on DualVD instead of DualVN, because this work is done before the work in [78].) In addition, our HierarchicalVN algorithm employs the proposed hierarchical matrix inversion algorithm to compute \( c_l \). The hypergraph model of the power grid is partitioned by hMETIS [69], a hypergraph partitioning tool. The LU factorization tool SuperLU [72] is used to compute \( A_m^{-1} \) in (4.27). Both algorithms are compiled with the same
GCC compiler and executed on one core of the same 64-bit Linux workstation with 2.4GHz Intel Q6600 processor and 8GB memory.

We adopt the 7 power grids used in [76] and follow [76] to generate 4 global constraints and to set both error tolerances $\delta_{\text{inv}}$ and $\delta_{\text{lp}}$ to 0.1mV. Additional settings of global constraints are not experimented since they introduce marginal running time overhead as shown in [76] for the second sub-problem and do not affect the computation of $c_i$.

Since hMETIS allows to choose between the recursive bisection partitioning and the k-way partitioning, we experiment with both options. The “unbalance factor” for these two options are set to 2 and 10 respectively. We observe that the k-way partitioning consumes much more memory than the recursive bisection partitioning, and it fails to partition the largest power grid of 562K non-VDD nodes into more than 1000 partitions due to lack of enough memory. For all the test cases that we have evaluated, recursive bisection partitioning consumes less runtime and generates slightly smaller number of hyperedge-cuts. Therefore, we only report the results using recursive bisection partitioning. For both of these two options, the runtime spent on partitioning the hypergraph model of the power grid ranges from a few seconds to about 10 minutes, and it is negligible in comparison with the overall runtime of the power grid verification.

The performance of the proposed hierarchical matrix inversion algorithm depends on the choice of the rough partition size $rps$. For all $rps$ settings from a few dozens to one thousand, $A_m^{-1}$ is computed by SuperLU which usually consumes less than 1 seconds and thus is negligible. On the other hand, as we increase $rps$, the number of partitions decreases and the number of internal nodes increases. Therefore, more $c_i$’s can be obtained using (4.27) instead of the PCG method. However, because the number of external neighbors for a single partition also increases as $rps$ increases,
it takes more runtime to compute the $c_i$ of each internal node. So, there is a trade-off between the increasing number of internal nodes and the increasing time required to compute the $c_i$‘s for them. To evaluate this trade-off, we perform a set of experiments using the largest power grid of 562K non-VDD nodes. We verify about 10K nodes of this power grid with rough partition size $rps \in \{20, 40, 60, 100, 250, 500, 750, 1000\}$, and estimate the average runtime per internal node and the overall average runtime per node. The results are illustrated in Fig. 4.13 where the x-axis is the percentage of the internal nodes. When $rps$ increases from 20 to 1000, the percentage of the internal nodes increases from 72% to 95%, and it takes more runtime to verify an internal node. The overall average first decreases as the number of internal nodes increases but then increases as it takes more runtime to compute $c_i$. We observe the minimum is reached at $rps = 100$ while the other choices would result in a maximum performance degradation close to 30%. We perform additional experiments with the aforementioned $rps$ setting for the other power grids, and notice that the minimum runtime per node is achieved at different $rps$ for different power grids, but in general a non-optimal but reasonable choice of $rps$ would not result in significant performance degradations. Therefore, we always choose $rps = 100$ for the HierarchicalVN algorithm hereafter.

The runtime comparison of the HierarchicalVN and DualVD is presented in Table 4.4. The runtime can be decomposed into two parts: the runtime to compute $c_i$ and the runtime to solve the LP problem (4.3). For the HierarchicalVN algorithm, the former consists of power grid partitioning runtime, PCG runtime, and the runtime to compute the $c_i$ of the internal nodes using (4.27). For the DualVD algorithm, the runtime to compute $c_i$ is just the PCG runtime. Both algorithms have approximately the same runtime for solving LP problems as they share the same code.

Since we focus on the efficiency of the hierarchical matrix inversion algorithm,
Table 4.4. Runtime Comparison of HierarchicalVN and DualVD with 4 Global Constraints

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>DualVD [76]</th>
<th>HierarchicalVN</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5875</td>
<td>28.40 s</td>
<td>38.45 s</td>
<td>59</td>
</tr>
<tr>
<td>22939</td>
<td>8.11 m</td>
<td>9.95 m</td>
<td>230</td>
</tr>
<tr>
<td>35668</td>
<td>21.83 m</td>
<td>25.91 m</td>
<td>357</td>
</tr>
<tr>
<td>51195</td>
<td>45.95 m</td>
<td>54.55 m</td>
<td>512</td>
</tr>
<tr>
<td>90643</td>
<td>2.57 h</td>
<td>3.01 h</td>
<td>907</td>
</tr>
<tr>
<td>141283</td>
<td>6.37 h</td>
<td>7.47 h</td>
<td>1413</td>
</tr>
<tr>
<td>562363</td>
<td>4.58 d</td>
<td>5.37 d</td>
<td>5624</td>
</tr>
</tbody>
</table>
we report the runtime for computing $c_l$ under columns “Inverse”. Moreover, we also report the total runtime under the columns “Total”. The time units are abbreviated such that “s”, “m”, “h” and “d” denote “seconds”, “minutes”, “hours”, and “days” respectively. As the DualVD algorithm takes too much time to verify the power grid of 562K non-VDD nodes, the reported runtime of the DualVD algorithm for that power grid is an estimation from the runtime of 1000 nodes chosen randomly. It can be seen that the HierarchicalVN algorithm achieves close to 3X speed-ups for all large power grids in comparison to the DualVD algorithm, which is highly non-trivial since DualVD is already very efficient. Moreover, for the HierarchicalVN algorithm, we report the number of partitions under the column “Parts” and the number of external neighbor nodes under the column “E.N.”. It confirms the effectiveness of the proposed power grid partitioning approach since only about 15% of all the nodes are identified as external neighbors with a small $rps$ setting of 100.

4.4 Constraint Abstraction

Although the aforementioned algorithms have largely improved the solution
efficiently of steady-state verification, the computation cost of these approaches is still too high. For example, as shown in Table 4.4, the verification of the largest synthetic benchmark with 562K nodes still requires two days to finish. This is mainly due to the fact that the problem sizes of (4.2) and (4.3) are proportional to the number of nodes in the power grid. Hence, it is of great interest to further reduce the computation cost by exploring more efficient techniques. In this section, we propose a constraint abstraction technique to reduce the computation cost of vectorless verification. The boundary condition of a subgrid is modeled by boundary constraints, which enable efficient calculation of conservative bounds of power supply noises in a divide-and-conquer manner.

4.4.1 Methodology. Consider a localized region of the power grid, referred to as a subgrid. For each subgrid, we refer to the nodes inside the subgrid as internal nodes, the nodes which are outside the subgrid but connected with some internal nodes as neighboring nodes, and other nodes in the power grid as external nodes.

As illustrated in Figure 4.14, conventional approaches verify an internal node of a subgrid by considering the whole grid structure, including all neighboring nodes,
external nodes, and the corresponding voltage supplies and current sources attached. The corresponding problem sizes of (4.2) and (4.3) increase as the size of the power grid increases. In order to verify the subgrid efficiently, we propose to treat neighboring nodes as \textit{uncertain voltage sources} (see Figure 4.14), which can be modeled by \textit{boundary constraints} (detailed in the next section) for realistic scenarios. As a result, the subgrid can be verified based on proper boundary constraints, without involving the external grid structure explicitly. The resultant problem sizes are roughly equal to the size of the subgrid, enabling significant reduction in computation cost. This approach is called \textit{constraint abstraction}, since boundary constraints provide a high-level abstraction of the boundary condition of the subgrid.

\textbf{4.4.2 Grid Partitioning \& Boundary Constraints.} We apply constraint abstraction for full-chip power grid verification in a \textit{divide-and-conquer} manner. As
illustrated in Figure 4.15, the grid is partitioned into several disjoint *subgrids*, which are split by a small set of *global nodes* (i.e., the neighboring nodes of subgrids). Such a partition can be obtained by using the power grid partitioning technique proposed in [77]. Typically, a proper partition results in relatively small number of global nodes, and most nodes are internal nodes of subgrids.

For each subgrid, we use *boundary constraints* to model its boundary condition, i.e., the voltage noises at its neighboring global nodes, which are connected with the internal nodes inside the subgrid. Let \( \hat{m} \) be the number of neighboring global nodes of a subgrid, and \( \mathbf{v}_{ex} \) be the \( \hat{m} \times 1 \) vector of voltage noises at these nodes. Then the boundary constraints are represented as

\[
0 \leq \mathbf{v}_{ex} \leq \mathbf{v}_\ell, \text{ and } \sum_{1 \leq j \leq \hat{m}} v_{ex,j} \leq v_g,
\]

where \( v_{ex,j} \) is the \( j \)th element of \( \mathbf{v}_{ex} \), \( \mathbf{v}_\ell \) is an upper bound vector on the voltage noises \( \mathbf{v}_{ex} \), and \( v_g \) is an upper bound on the sum of these voltage noises.

In practice, \( \mathbf{v}_\ell \) can be either the exact worst-case voltage noises at neighboring global nodes, or some upper bounds, which are computed by verifying global nodes. \( v_g \) can be computed by solving the following linear program.

\[
\text{Maximize } \sum_{1 \leq j \leq \hat{m}} v_{ex,j} \quad \text{s.t.} \quad A\mathbf{v} = \mathbf{i}, 0 \leq \mathbf{i} \leq \mathbf{I}_L, U\mathbf{i} \leq \mathbf{I}_G.
\]

Similar to solving the sub-problems (4.2) and (4.3) to compute the worst-case voltage noise, we calculate a coefficient vector to represent the objective function as an affine function of current sources. This can be achieved by solving a linear system \( A\mathbf{x} = \mathbf{e}_g \), where \( \mathbf{e}_g \) is a vector of 0s except that its elements corresponding to the neighboring global nodes are set to 1. With such a coefficient vector in hand, the optimal value of
(4.34) can be obtained by solving a linear program like (4.3). Here, the computation cost is equivalent to verifying a node by solving (4.2) and (4.3).

Based on the above discussion, a partitioned power grid can be verified in the following steps.

1. For each global node, compute its worst-case voltage noise (or an upper bound of voltage noise).

2. For each subgrid, build boundary constraints, and then evaluate the worst-case voltage noises at internal nodes subject to boundary constraints.

In the first step, as there are only relatively small amount of global nodes, previous methods [2, 76, 78] can be applied. The verification of subgrids in the second step is detailed in the next section.

4.4.3 Verification of Subgrids. As summarized in Problem 1, we need to evaluate the worst-case voltage noises based on the generalized system equation $A\mathbf{v} = \mathbf{i}$ for both DC and transient models. Consider a subgrid with $\hat{n}$ internal nodes and $\hat{m}$ neighboring global nodes. The equation $A\mathbf{v} = \mathbf{i}$ is reduced to

$$
\begin{bmatrix}
A_{in} & A_{ex}
\end{bmatrix}
\begin{bmatrix}
\mathbf{v}_{in} \\
\mathbf{v}_{ex}
\end{bmatrix} = \mathbf{i}_{in},
$$

(4.35)

where $A_{in}$ is the $\hat{n} \times \hat{n}$ conductance matrix of the subgrid, $A_{ex}$ is a non-positive $\hat{n} \times \hat{m}$ matrix representing the conductance links between internal nodes and neighboring global nodes, $\mathbf{v}_{in}$ and $\mathbf{i}_{in}$ are the vectors of voltage noises and current sources at internal nodes, respectively.

Since $A_{in}$ is also a symmetric positive definite $M$-matrix, it is invertible, and
\( A_{in}^{-1} \) is symmetric and non-negative. Then equation (4.35) can be rearranged as

\[
v_{in} = A_{in}^{-1}i_{in} - A_{in}^{-1}A_{ex}v_{ex}, \tag{4.36}
\]

Let \( v_{in,j} \) be the \( j \)th element of \( v_{in} \), \( c_{in,j} \) be the \( j \)th column of \( A_{in}^{-1} \), and \( c_{ex,j} \) be the transpose of the \( j \)th row of \(-A_{in}^{-1}A_{ex} \). (Note that both \( c_{in,j} \) and \( c_{ex,j} \) are non-negative, and \( c_{ex,j}^T = -c_{in,j}^T A_{ex} \).) We have

\[
v_{in,j} = c_{in,j}^T i_{in} + c_{ex,j}^T v_{ex}, \tag{4.37}
\]

where the internal current vector \( i_{in} \) is defined by local and global current constraints, and the boundary condition \( v_{ex} \) is restricted by boundary constraints.

As a result, the subgrid can be verified by solving the following linear program for each internal node \( 1 \leq j \leq \hat{m} \).

Maximize \( v_{in,j} = c_{in,j}^T i_{in} + c_{ex,j}^T v_{ex} \) s.t.

\[
0 \leq i_{in} \leq \hat{I}_L, \hat{U} i_{in} \leq \hat{I}_G,
\]

\[
0 \leq v_{ex} \leq v_{\ell}, \sum_{1 \leq j \leq \hat{m}} v_{ex,j} \leq v_g,
\]

where \( \hat{I}_L, \hat{I}_G \) and \( \hat{U} \) are matrices for local and global current constraints related to the subgrid. It is to be noted that the decision variables \( i_{in} \) and \( v_{ex} \) are independent, thus (4.38) can be further decomposed into two linear programs and solved separately.

Maximize \( c_{in,j}^T i_{in} \) s.t. \( 0 \leq i_{in} \leq \hat{I}_L, \hat{U} i_{in} \leq \hat{I}_G \),

Maximize \( c_{ex,j}^T v_{ex} \) s.t. \( 0 \leq v_{ex} \leq v_{\ell}, \sum_{1 \leq j \leq \hat{m}} v_{ex,j} \leq v_g \).

\( \text{Lemma 5} \) Let \( \hat{v}_{in,j} \) be the exact worst-case voltage noise at internal node \( j \) (i.e., the
optimal value of the linear program (4.1)), and $v_{in,j}^+$ be the worst-case voltage noise computed through constraint abstraction (i.e., the optimal value of the linear program (4.38)), then $v_{in,j}^* \leq v_{in,j}^+$.

**Proof 5** Let $i_{in}^*$ and $v_{ex}^*$ be the local current vector inside the subgrid and the voltage noises at neighboring global nodes corresponding to $v_{in,j}^*$, respectively, so that

$$v_{in,j}^* = c_{in,j}^T i_{in}^* + c_{ex,j}^T v_{ex}^*.$$ 

Let $i_{in}^+$ and $v_{ex}^+$ be the optimal solution of (4.38), so that

$$v_{in,j}^+ = c_{in,j}^T i_{in}^+ + c_{ex,j}^T v_{ex}^+.$$ 

Clearly, we have $c_{in,j}^T i_{in}^* \leq c_{in,j}^T i_{in}^+$, and $c_{ex,j}^T v_{ex}^* \leq c_{ex,j}^T v_{ex}^+$. It follows that $v_{in,j}^* \leq v_{in,j}^+$.

This verification approach derives an upper bound of voltage noise at each internal node inside the subgrid. The difference between the computed upper bound $v_{in,j}^+$ and the exact worst-case voltage noise $v_{in,j}^*$ is called *overestimation*. As we will show in the experimental results, the amount of overestimation is very small, i.e., the computed voltage noise bound is fairly tight and realistic.

Except for solving (4.34) to obtain $v_g$, verifying the subgrid mainly involves computing $A_{in}^{-1}$ and $-A_{in}^{-1} A_{ex}$, and solving the linear programs (4.39) and (4.40) for each internal node. By keeping the size of the subgrid within some bound, the Cholesky decomposition of $A_{in}$ can be computed, and then $A_{in}^{-1}$ can be calculated row by row (or column by column since $A_{in}^{-1}$ is symmetric) by solving a linear system like (4.2) through forward and back substitutions. As $A_{ex}$ is a sparse matrix representing the links between internal nodes and neighboring global nodes, the matrix multiplication $-A_{in}^{-1} A_{ex}$ can be performed efficiently. In practice, both $A_{in}^{-1}$ and $-A_{in}^{-1} A_{ex}$
are not stored explicitly, their rows (i.e., \( c_{m,j}^T \) and \( c_{ex,j}^T \)) are computed, used and discarded at runtime. The linear program (4.39) can be solved by standard LP solvers if the subgrid size is within some reasonable bound. As the linear program (4.40) only has one constraint defining an upper bound of the sum of all variables, it can be efficiently solved by sorting the coefficients in non-increasing order, and setting the corresponding variable to be the maximum feasible value sequentially.

4.4.4 Analysis of Computation Cost. In this subsection, we present the computation advantage of the proposed constraint abstraction approach over a direct one. Since the computation for \(-A_{in}^{-1}A_{ex}\) and (4.40) can be performed fairly efficiently, the major computation cost for verifying a subgrid is to solve (4.34), to calculate \(A_{in}^{-1}\), and to solve linear program (4.39) for each internal node. In contrast, a direct approach computes \(A^{-1}\) (by solving (4.2)) and solves linear program (4.3) for all the nodes.

Suppose the cost of computing the Cholesky decomposition is \(f_1(N)\), the cost of one forward and one back substitution is \(f_2(N)\), and the cost of solving a linear program (like (4.3)) is \(f_3(N)\), where \(N\) is the size of the matrix, and the number of decision variables in the linear program. Typically, \(f_3(N) \gg f_1(N) \gg f_2(N)\).

Consider a direct approach with Cholesky decomposition for computing \(A^{-1}\). The total computation cost is given by

\[
f_1(n) + nf_2(n) + nf_3(n).
\] (4.41)

Let \(n_0\) be the number of global nodes, \(n_j, 1 \leq j \leq k\) be the number of internal nodes inside each subgrid, where \(\sum_{0 \leq j \leq k} n_j = n\). The cost of verifying global nodes with the direct approach is \(f_1(n) + n_0f_2(n) + n_0f_3(n)\). For each subgrid, the cost of solving (4.34) is \(f_2(n) + f_3(n)\), the cost of computing \(A_{in}^{-1}\) and solving linear program (4.39) is
\[ f_1(n_j) + n_jf_2(n_j) + n_jf_3(n_j). \]
Hence, the computation cost of the proposed approach can be approximated as

\[ f_1(n) + (n_0 + k)f_2(n) + (n_0 + k)f_3(n) \]
\[ + \sum_{1 \leq j \leq k} (f_1(n_j) + n_jf_2(n_j) + n_jf_3(n_j)). \quad (4.42) \]

Expressions (4.41) and (4.42) provide a rough estimation of computation costs based on the size of the power grid and its subgrids. Generally, the time complexity of Cholesky decomposition is \(O(N^3)\), the time complexity of forward and back substitutions is \(O(N^2)\), and the complexity of linear programming by standard solvers can be in even higher order (e.g., \(O(N^4)\)). In practice, the sparsity of the conductance matrix \(A\) (as well as the constraint matrix \(U\)), combined with efficient reordering, enables actual computation costs to be less than such theoretical bounds, but the cost of the direct approach still remains greater than quadratic. Hence, the computation cost of the proposed approach will be much smaller than that of the direct approach if the power grid is partitioned properly.

4.4.5 Experiments.

4.4.5.1 Experimental Setup. The proposed constraint abstraction approach has been implemented in C++. hMETIS [69] is utilized to partition the power grid into subgrids with relatively small number of global nodes. We use a user-specified rough subgrid size called \(rss\) to control the size of subgrids, and the number of subgrids is calculated by \(\text{round}(\frac{n}{rss})\), where \(n\) is the number of nodes (after merging nodes connected by shorts) in the grid. The DualVN algorithm proposed in [78] is employed to verify global nodes, and to solve the linear program (4.34) for building boundary constraints, where its error tolerance for solving linear programs is set to be 0.1mV. CHOLMOD [67] and GotoBLAS2 [28] are employed to solve the involved
linear systems through Cholesky factorization, and MOSEK [70] is used to solve the linear program (4.39) for verifying each subgrid. We choose the simplex method of MOSEK, since it is slightly faster than the interior point method for our experiments.

For performance comparison, we implemented the DirectVN algorithm of [78] by solving linear system (4.2) and linear program (4.3) with CHOLMOD and MOSEK, respectively. The DualVN algorithm [78] has also been implemented for verifying the whole power grid, while linear system (4.2) is solved by CHOLMOD. Experiments are carried out on a 64-bit Linux server with 2.67GHz Intel Xeon X5650 processor and 64GB memory. Although the server has 12 cores, only one core is used for experiments.

We employ the synthetic power grids used in [78] for performance tests, and also experiment with IBM power grid benchmarks [35]. As IBM power grids have multiple networks, we choose to verify the ground network in our experiments, while their VDD networks can be verified separately. Local current constraints are extracted from the grid description, and global current constraints are generated by scaling down the total amount of current drawn by groups of current sources. For each power grid, we specify 4 global constraints in our tests.

4.4.5.2 Exploring Rough Subgrid Size. Since the computation cost of the proposed approach is highly dependent on the sizes of subgrids as discussed in Section 4.4.4, we experiment with different rough subgrid sizes $rss$ (ranging from 100 to 10K) for performance analysis. Note that the actual sizes of most subgrids would be slightly smaller than $rss$, because global nodes do not belong to any subgrid. The total runtime of the proposed approach can be roughly divided into three parts: the runtime to partition the power grid, the runtime to solve global nodes, and the runtime to verify subgrids. Figure 4.16 plots these runtime components for verifying a synthetic power grid with different $rss$. As $rss$ increases, both the number of
subgrids and the number of resultant global nodes decrease, so it takes less runtime
to partition the power grid, and to verify global nodes. On the contrary, both the
subgrid sizes and the total number of internal nodes increase as $rss$, thus solving
the subgrids consumes more runtime when $rss$ becomes larger. Obviously, there is a
tradeoff between the computation costs of global nodes and subgrids. If the runtime
for grid partitioning is sufficiently small, then the minimum runtime is achieved at
the best tradeoff point, where the runtime of global nodes and the runtime of subgrids
are roughly equal, e.g., the optimal $rss$ is 1K for the test case shown in Figure 4.16.

Recall that constraint abstraction results in overestimation of the worst-case
voltage noises as stated in Lemma 5. We also evaluate the effects of different $rss$ on
the amount of overestimation. As shown in Figure 4.17, both the maximum and the
average overestimation increase when a larger $rss$ is specified. This phenomenon is
due to the fact that the boundary constraints become less effective when the subgrid
sizes become larger.

To determine a proper $rss$, both runtime and solution accuracy need to be
Figure 4.17. Solution accuracy of verifying a synthetic power grid “pg4000” (90,643 nodes) with different rough subgrid sizes \((rss)\). \(E_{\text{max}}/E_{\text{avg}}\): the maximum/average overestimation of voltage noises in mV.

considered. Fortunately, the overestimation for most \(rss\) settings are sufficiently small, so that the \(rss\) which provides the best runtime efficiency can be employed for production runs.

4.4.5.3 Performance Results. The performance data are presented in Table 4.5. The runtime of DirectVN and DualVN are reported under columns 4 and 5 for performance comparison. The golden solution is produced by DirectVN for analysis of overestimation. Since DiretVN takes too much runtime to verify large power grids, the reported data with \(\approx\)s are estimations from 1000 random nodes. The DualVN algorithm is fairly efficient, and can solve the largest grid “pg10000” with 562K nodes in 19 hours. The proposed constraint abstraction approach further improves the runtime efficiency of vectorless verification, achieving up to about 17X speedup over DualVN. As a result, “pg10000” can be verified in about 1 hour. Generally, grid partitioning takes a few seconds to a few minutes depending on the grid size and the number of desired subgrids. Most runtime of the proposed approach is spent on solving global nodes and subgrids. Moreover, for most test cases, the maximum overestimation is
Table 4.5. Performance Results of the Proposed Constraint Abstraction Approach. $n$: the number of nodes after merging the nodes connected by shorts; $v_{max}$: the maximum voltage noise across the grid in mV; $rss$: rough subgrid size; $n_0$: the number of global nodes; $E_{max}/E_{avg}$: the maximum/average overestimation of voltage noises in mV; the runtime units “s”, “m”, “h” and “d” represent seconds, minutes, hours, and days, respectively.

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>DirectVN</th>
<th>DualVN [78]</th>
<th>Constraint Abstraction</th>
<th>Speedup Relative to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>$n$</td>
<td>$v_{max}$</td>
<td>Runtime</td>
<td>$rss$</td>
</tr>
<tr>
<td>pg1000</td>
<td>5875</td>
<td>46.31</td>
<td>2.66 m</td>
<td>15.40 s</td>
</tr>
<tr>
<td>pg2000</td>
<td>22939</td>
<td>39.91</td>
<td>53.47 m</td>
<td>2.52 m</td>
</tr>
<tr>
<td>pg2500</td>
<td>35668</td>
<td>28.80</td>
<td>2.06 h</td>
<td>5.85 m</td>
</tr>
<tr>
<td>pg3000</td>
<td>51195</td>
<td>43.63</td>
<td>4.83 h</td>
<td>12.27 m</td>
</tr>
<tr>
<td>pg4000</td>
<td>90643</td>
<td>54.38</td>
<td>15.93 h</td>
<td>38.73 m</td>
</tr>
<tr>
<td>pg5000</td>
<td>141283</td>
<td>$\approx$45.91</td>
<td>$\approx$2.15 d</td>
<td>1.25 h</td>
</tr>
<tr>
<td>pg10000</td>
<td>562363</td>
<td>$\approx$23.11</td>
<td>$\approx$44.07 d</td>
<td>18.58 h</td>
</tr>
<tr>
<td>ibmpg1</td>
<td>10242</td>
<td>677.67</td>
<td>4.26 m</td>
<td>26.09 s</td>
</tr>
<tr>
<td>ibmpg2</td>
<td>65228</td>
<td>357.92</td>
<td>3.95 h</td>
<td>16.65 m</td>
</tr>
<tr>
<td>ibmpg3</td>
<td>150687</td>
<td>$\approx$179.91</td>
<td>$\approx$3.47 d</td>
<td>1.29 h</td>
</tr>
<tr>
<td>ibmpg4</td>
<td>478094</td>
<td>$\approx$3.42</td>
<td>$\approx$14.12 d</td>
<td>15.37 h</td>
</tr>
<tr>
<td>ibmpg5</td>
<td>291382</td>
<td>$\approx$42.44</td>
<td>$\approx$19.51 d</td>
<td>6.13 h</td>
</tr>
<tr>
<td>ibmpg6</td>
<td>430337</td>
<td>$\approx$109.96</td>
<td>$\approx$55.97 d</td>
<td>12.47 h</td>
</tr>
</tbody>
</table>
within 3mV, and the average overestimation is much smaller than 1mV. In addition, it is worth noting that the proposed approach is more effective when the grid size becomes larger, making it suitable for large-scale power grid verification.

The major merit of constraint abstraction is that it enables significant reduction in computation cost with small amount of overestimation. In practice, it can be used for fast estimation of power supply noises across the chip. The risky regions can be identified in small amount of runtime, and then more accurate methods can applied if it is necessary.
VECTORLESS TRANSIENT VERIFICATION

Vectorless transient verification is even more challenging than steady-state verification, since multiple time steps should be evaluated after the discretization of the system equation. In this chapter, we consider integrated RLC power grids with both VDD and GND networks, and introduce transient constraints to restrict the waveform of each current source for sign-off verification. Moreover, we propose a general methodology to perform transient verification by transient simulation and noise optimization, and design a variable reduction algorithm to simplify the vectorless verification problem with a user-specified error tolerance, so that the conservative bounds of voltage noises can be computed efficiently.

5.1 Background

5.1.1 RLC Power Grid Model. We consider an integrated RLC power grid model as illustrated in Fig. 5.1. It consists of resistors, inductors, capacitors, current sources, VDD and GND pads. Each branch is represented by a resistor, or an inductor, or a capacitor. As this grid contains both VDD and GND networks, we refer to the nodes in the VDD network as supply nodes, and the nodes in the GND network as ground nodes. Resistors and inductors are only located between two supply nodes or two ground nodes, while capacitors are only located between a supply node and a ground node. For simplicity of illustration, Fig. 5.1 only shows at most a single capacitor connected with a node. However, it is assumed that a node can be connected with multiple capacitors. The current sources attached to the grid model the behavior of the underlying circuitry, which draws current from the VDD network and injects current to the GND network. In this paper, we assume a single supply voltage, while this RLC power grid model and the proposed vectorless verification approach are also applicable for power grids with multiple supply voltages.
Let $n$ be the total number of nodes that are not VDD/GND pads in the power grid, $u_j(t)$ and $I_j(t)$ be the nodal voltage and the current source at node $j = 1, 2, 3, \ldots, n$, respectively. It is assumed that $I_j(t) = 0$ if node $j$ does not have a current source attached, and the positive direction of current is from VDD to GND. Let $\mathbf{I}(t)$ be the $n \times 1$ vector of current sources, and $\mathbf{\hat{I}}(t)$ be the $n \times 1$ vector representing the incoming current source of each node, i.e., its $j$th element $\hat{I}_j(t)$ is defined as

$$\hat{I}_j(t) \triangleq \begin{cases} -I_j(t), & \text{if node } j \text{ is a supply node,} \\ I_j(t), & \text{if node } j \text{ is a ground node.} \end{cases}$$ \hspace{1cm} (5.1)$$

Let $\mathbf{v}(t)$ be the $n \times 1$ vector of voltage noises with its $j$th element

$$v_j(t) \triangleq \begin{cases} u_j(t) - vdd, & \text{if node } j \text{ is a supply node,} \\ u_j(t), & \text{if node } j \text{ is a ground node,} \end{cases}$$

where $vdd$ is the supply voltage. As derived in [79], the system equation of the power
grid can be formulated as
\[ Lv(t) + Gv'(t) + Cv''(t) = \left( \dot{I}(t) \right)', \]
(5.2)

where \( G \) is the \( n \times n \) conductance matrix, \( L \) is the \( n \times n \) matrix similar to the conductance matrix but representing inductance links with its elements being the reciprocals of inductance values, and \( C \) is the \( n \times n \) matrix similar to the conductance matrix but representing capacitance links.

Using the backward Euler method or the trapezoidal rule [52], (5.2) can be discretized in time and rearranged as

\[
\begin{align*}
\left( G + \Delta t L + \frac{1}{\Delta t} C \right) v(t) &= \dot{I}(t) - \dot{I}(t - \Delta t) + \\
\left( G + \frac{2}{\Delta t} C \right) v(t - \Delta t) - \frac{C}{\Delta t} v(t - 2\Delta t), \text{ or}
\end{align*}
\]

\[
\begin{align*}
\left( G + \frac{\Delta t}{2} L + \frac{2}{\Delta t} C \right) v(t) &= \dot{I}(t) - \dot{I}(t - 2\Delta t) + \left( \frac{4}{\Delta t} C - \Delta t L \right) v(t - \Delta t) + \\
\Delta t L v(t - \Delta t) + \left( G - \frac{\Delta t}{2} L - \frac{2}{\Delta t} C \right) v(t - 2\Delta t),
\end{align*}
(5.3)
\]

respectively, where \( \Delta t \) is the time step. Clearly, these system equations are similar to the DC analysis equation \( Gv = \dot{I} \), where the left-hand-side power grid matrix is a combination of R/L/C components, and the right-hand-side vector is computed from current and previous \( \dot{I}/v \) states. In this paper, we employ the trapezoidal rule for grid verification since it has better accuracy. For simplicity of notations, we define \( n \times n \) matrix \( A, B \) and \( D \) as

\[
A \triangleq G + \frac{\Delta t}{2} L + \frac{2}{\Delta t} C, \quad B \triangleq \frac{4}{\Delta t} C - \Delta t L, \quad D \triangleq G - \frac{\Delta t}{2} L - \frac{2}{\Delta t} C.
\]

Note that \( A \) is a symmetric \( \mathcal{M} \)-matrix, so \( A \) is invertible and \( A^{-1} \) is symmet-
ric. $\mathbf{B}$ and $\mathbf{D}$ are also symmetric due to the fact that $\mathbf{G}$, $\mathbf{L}$ and $\mathbf{C}$ are symmetric. Therefore, equation (5.3) can be simplified and rearranged as

$$
\mathbf{v}(t) = \mathbf{A}^{-1}\left(\hat{\mathbf{I}}(t) - \hat{\mathbf{I}}(t-2\Delta t) + \mathbf{B}\mathbf{v}(t-\Delta t) + \mathbf{D}\mathbf{v}(t-2\Delta t)\right),
$$

which represents the voltage noises at time $t$ as a function of current excitations and the voltage noises at previous time steps, and it will be used to verify the voltage noise across the power grid.

5.1.2 Current Constraints. As studied in [6], the current waveforms of the power grid including both VDD and GND networks can be modeled by three types of constraints: local constraints, global constraints and equality constraints. Recall that local constraints and global constraints are introduced in 2.2. To verify both VDD and GND networks, we need ensure that the current flowing out of the VDD network is equal to the current flowing into the GND network. For a circuit block, this is also true if the input and output currents are negligible. If we assume that there are $b$ circuit blocks satisfying this equality relationship, then the equality constraints can be formulated as

$$
\mathbf{E}\mathbf{I}(t) = 0, \forall t, \text{ or } \mathbf{E}\mathbf{I}(k\Delta t) = 0, \forall k,
$$

where $\mathbf{E}$ is a $b \times n$ matrix consisting of $\pm 1$s and $0$s. For each circuit block, $+1$s and $-1$s correspond to the current sources that are attached to the VDD and GND networks, respectively, while $0$s correspond to other current sources.

5.2 Vectorless Verification with Transient Current Constraints

In this section, we propose to perform vectorless verification with transient current constraints. Section 5.2.1 introduces transient current constraints, Section 5.2.2 presents the problem formulation, and a case study is presented in Section 5.2.3.
5.2.1 Transient Current Constraints. The current excitations at a particular
time $t$ is well defined by local, global and equality constraints. However, these con-
straints can not model the transient behavior of current sources. When verifying the
power grid, ignoring the transient behavior will lead to pessimistic estimation of the
voltage noise, which is caused by un-realistic transient waveforms. For example, con-
sider a node which supplies a gate in a power grid. If we evaluate the voltage noise
without considering the transient characteristic of the load current, the worst-case
voltage noise may be achieved when the gate draws the maximum current over a long
time period, which is never the case.

In order to capture the transient behavior of current sources, we propose novel
transient constraints to restrict the total amount of current (or more exactly “charge”)
that each current source can draw within a time interval, i.e., a number of continuous
time steps. Let $N_{ts}$ be the number of time steps under consideration, then transient
constraints can be formulated as

$$
\int_0^{N_{ts} \times \Delta t} I(t) dt \leq I_T \times \Delta t, \text{ or } \sum_{k=1}^{N_{ts}} I(k \Delta t) \leq I_T,
$$

where $I_T \geq 0$ is an $n \times 1$ upper bound vector, and the integration operation is
element-wise. For each current source, the transient constraint can be viewed as its
power constraint over the time interval. Different from hierarchical power constraints
[18], which restrict the power consumption of circuit blocks in a hierarchical manner,
 transient constraints bound the power consumption of individual current source.

To extract these transient constraints from the underlying circuitry, we must
analyze the circuit to derive the maximum amount of switching instants for each
gate/cell within $N_{ts}$ time steps, then translate these switching instants into current
waveforms, and finally discretize the waveform to get transient upper bounds. As
switching activity analysis has already been studied in [49] and [50] to generate real-
istic stimuli for power grid analysis, we can follow these works to compute $I_T$.

In this work, we consider the sign-off verification of power grids with local constraints, global constraints, equality constraints, and transient constraints, while the proposed vectorless verification approach can also be extended to handle other constraints.

5.2.2 Problem Formulation. As studied in [1] and [6], the nodal voltage of an RC/RLC power grid can fluctuate in both directions, i.e., overshoot and voltage drop in the VDD network, ground bounce and undershoot in the GND network. In many cases, overshoot and undershoot cannot be neglected. They can be even comparable to voltage drop and ground bounce as shown by the case study in Section 5.2.3 and the experimental results in Table 5.2. To verify the power grid conservatively, we need to evaluate the worst-case voltage noises in both directions.

Assume that there is no current excitation for all $t \leq 0$, so that $v(t) = 0$, $\forall t \leq 0$. Consider $N_{ts}$ time steps, then the vectorless verification is to solve the following optimization problem for each node $1 \leq j \leq n$,

$$\text{Maximize/Minimize } v_j(t), \forall t = k'\Delta t, 1 \leq k' \leq N_{ts}, \quad (5.5)$$

subject to: $\forall 1 \leq k \leq N_{ts}, v(k\Delta t) = A^{-1}\left(\widehat{I}(k\Delta t) - \widehat{I}((k-2)\Delta t) + Bv((k-1)\Delta t) + Dv((k-2)\Delta t)\right)$,

$0 \leq I(k\Delta t) \leq I_L, UI(k\Delta t) \leq I_G, EI(k\Delta t) = 0$,

and $\sum_{k=1}^{N_{ts}} I(k\Delta t) \leq I_T$,

where $\widehat{I}(t)$ is defined in (5.1), and it represents the incoming current source of each node. By maximizing the voltage noise, we get the worst-case overshoot or ground bounce. By minimizing the voltage noise, we obtain the worst-case voltage drop or
undershoot. Clearly, two linear programming (LP) problems need to be solved for each node at each time step. There is a group of constraints at each time step except for the transient constraints, resulting in complicate LP problems.

According to [1], the optimization problems at time $t - \Delta t$ are sub-problems of the optimization problems at time $t$. For each node, the magnitude of the worst-case voltage noise is a non-decreasing function for all $t \geq 0$, and this is also proved in [18]. Therefore, we only need to solve two LP problems for each node at time $t = N_{ts} \Delta t$ to verify the grid. For each node $1 \leq j \leq n$,

$$\text{Maximize/Minimize } v_j(N_{ts} \Delta t), \quad (5.6)$$

subject to the same set of constraints as stated in (5.5).

Although most works focus on solving the worst-case voltage noise at each node, it is proposed in [24] that verifying the integral of voltage noise (or the mean voltage noise) is more important, because a sharp voltage noise may not affect timing, but a large cumulative voltage noise will. Let’s still consider $N_{ts}$ time steps, then the problem for verifying the integral of voltage noise can be formulated as follows. For each node $1 \leq j \leq n$,

$$\text{Maximize/Minimize } \sum_{k=1}^{N_{ts}} v_j(k \Delta t), \quad (5.7)$$

subject to the same set of constraints as stated in (5.5).

However, it is very challenging to solve either (5.6) or (5.7) directly, because the constraints are too complicated, especially the relationships between voltage noises and current excitations. As there are $2n$ LP problems and $n$ is usually large for practical power grids, such LP problems have to be solved very efficiently.
5.2.3 A Case Study. Before presenting the proposed approach for vectorless verification, we use a case study to demonstrate the importance of vectorless verification with transient constraints.

Consider the simple RLC power grid shown in Fig. 5.2, we are in particular interested in the voltage noise at node $j$. Since power grid can be viewed as a linear time-invariant (LTI) system, the voltage noise at each node is the sum of voltage noises caused by individual current source (assuming other current sources being 0). For this simple grid, the voltage noise at node $j$ is attributable to the response of three current sources independently. As shown in Fig. 5.3, the voltage noise has different frequency response corresponds to each current source, the worst-case current pattern would be a combination of current waveforms at different frequencies with specific phases. However, such worst-case current excitation is not obvious from the standpoint of designers, and it can only be solved under the optimization framework of vectorless verification. Although some realistic current pattern can be extracted for power grid analysis if the circuit design is completed, the pattern extraction is mainly based on the circuit and often overlooks the characteristics of the grid, so the resulting voltage noise estimation may be optimistic. Therefore, vectorless verification is a critical technique to ensure robust power grid design. Except for early power grid verification, it can also serve as an alternative approach to estimate the conservative voltage noise for sign-off verification.

Using time step $\Delta t = 10$ps and the number of time steps $N_{ts} = 10,000$, we evaluate the worst-case voltage noise of node $j$ at $t = 100$ns by solving the vectorless verification problem (5.6). Except for the local constraints which define an upper bound of 10mA for the three current sources, a transient constraint $\sum_{k=1}^{10,000} I(k \times 10$ps) $\leq 10,000$mA is specified for each current source to restrict the transient current waveform. As illustrated in Fig. 5.4 and 5.5, the computed worst-
Figure 5.2. A simple RLC power grid.

Figure 5.3. The frequency response of the voltage noise at node $j$ for each current source.
Figure 5.4. Worst-case current waveforms for the maximum voltage drop with and without transient constraints (TC). The maximum voltage drops with and without TC are 61.8 and 83.8mV, respectively. Ignoring TC leads to a 35.6% overestimation on the voltage drop.

Figure 5.5. Worst-case current waveforms for the maximum overshoot with and without transient constraints (TC). The maximum overshoots with and without TC are 60.8 and 84.4mV, respectively. Ignoring TC leads to a 38.8% overestimation on the overshoot.
case current waveforms without transient constraints are regular periodic square waves except I3. Theoretically, the exact worst-case waveforms of I3 without transient constraints should also be regular and periodic like the current pattern from $t = 70$ to $100\text{ns}$, the computed worst-case current patterns are not regular because of the round-off error. With transient constraints, the current waveforms are restricted to more realistic scenarios for noise estimation as shown in Fig. 5.4 and 5.5. It is worth noting that the worst-case current waveforms with transient constraints have less pulses with possibly smaller pulse width compared to those without transient constraints. The kept pulses under transient constraints are close to the end of the time interval, because the worst-case voltage noise (at $t = 100\text{ns}$) is mainly due to the most recent current excitations. The verification without transient constraints leads to a 35.6% overestimation on the voltage drop and a 38.8% overestimation on the overshoot. In practice, such overestimation depends on grid structure and constraints, and may vary case by case. To make more realistic voltage noise predictions, it is important to employ transient constraints for vectorless verification.

5.3 Proposed Methodology

In this section, we first introduce two important properties of the system equation in Section 5.3.1, formulate the expression of voltage noises in Section 5.3.2, then present the problem decomposition in Section 5.3.3, and finally propose the methodology for vectorless verification in Section 5.3.4.

5.3.1 Properties of System Equation. Based on the initial condition that the power grid has no stimulus when $t \leq 0$, we can write the system equation of the power grid at different time steps according to (5.4). For example, at time $t =$
\[ v(\Delta t) = A^{-1}I(\Delta t), \]
\[ v(2\Delta t) = A^{-1}(I(2\Delta t) + Bv(\Delta t)) \]
\[ = A^{-1}I(2\Delta t) + A^{-1}BA^{-1}I(\Delta t), \]
\[ v(3\Delta t) = A^{-1}(I(3\Delta t) - I(\Delta t) + Bv(2\Delta t) + Dv(\Delta t)) \]
\[ = A^{-1}I(3\Delta t) + A^{-1}BA^{-1}I(2\Delta t) + \]
\[ (A^{-1}BA^{-1}BA^{-1} + A^{-1}DA^{-1} - A^{-1})I(\Delta t). \]

Generally, we have the following lemma.

**Lemma 6** There exist a unique series of \( n \times n \) matrices \( H_1, H_2, \cdots, H_k, H_{k+1}, \cdots \), such that \( \forall k \geq 1 \), we have

\[ v(k\Delta t) = \sum_{p=1}^{k} H_p I((k+1-p)\Delta t) \] (5.8)

\[ = H_1 I(k\Delta t) + H_2 I((k-1)\Delta t) + \cdots + H_k I(\Delta t). \]

**Proof 6** According to the expression of \( v(\Delta t), v(2\Delta t), \) and \( v(3\Delta t) \), we have

\[ H_1 = A^{-1}, \quad H_2 = A^{-1}BA^{-1}, \] (5.9)
\[ H_3 = A^{-1}BA^{-1}BA^{-1} + A^{-1}DA^{-1} - A^{-1}. \] (5.10)

Based on (5.4), we can infer that

\[ H_k = A^{-1}(BH_{k-1} + DH_{k-2}), \forall k \geq 4. \] (5.11)

**Lemma 6** and the formulations of \( H_k \) (5.9-5.11) can be proved by induction as follows.
Obviously, Lemma 6, (5.9-5.11) are true when \( k = 1, 2, 3, 4 \). For \( k \geq 5 \), assume that

\[
\begin{align*}
v_{k-2} &= H_1 \hat{I}_{k-2} + H_2 \hat{I}_{k-3} + \cdots + H_{k-2} \hat{I}_1, \\
v_{k-1} &= H_1 \hat{I}_{k-1} + H_2 \hat{I}_{k-2} + \cdots + H_{k-1} \hat{I}_1,
\end{align*}
\]

where \( v_k \) and \( \hat{I}_k \) represent \( v(k\Delta t) \) and \( \hat{I}(k\Delta t) \), respectively. Also assume that (5.9-5.11) hold for \( H_p, \forall 1 \leq p \leq k-1 \). Then, according to (5.4), we have

\[
\begin{align*}
v_k &= A^{-1}(\hat{I}_k - \hat{I}_{k-2} + Bv_{k-1} + Dv_{k-2}) \\
    &= A^{-1}\hat{I}_k + A^{-1}BH_1\hat{I}_{k-1} + \\
    &\quad (A^{-1}BH_2 + A^{-1}DH_1 - A^{-1})\hat{I}_{k-2} + \cdots + \\
    &\quad A^{-1}(BH_{k-2} + DH_{k-3})\hat{I}_2 + \\
    &\quad A^{-1}(BH_{k-1} + DH_{k-2})\hat{I}_1 \\
    &= H_1 \hat{I}_k + H_2 \hat{I}_{k-1} + \cdots + H_{k-1} \hat{I}_2 + H_k \hat{I}_1,
\end{align*}
\]

where (5.9-5.11) still hold for \( H_p, \forall 1 \leq p \leq k \). Therefore, by induction, Lemma 6 is true, and \( H_k \) satisfies (5.9-5.11).

Lemma 7 \( \forall k \geq 1, H_k \text{ is symmetric.} \)

Proof 7 Recall that \( A^{-1}, B \) and \( D \) are symmetric, so \( H_1, H_2 \) and \( H_3 \) are symmetric. Moreover, it can be verified that \( H_4 \) and \( H_5 \) are also symmetric. For \( k \geq 6 \), assume
that $H_p, \forall 1 \leq p \leq k - 1$ are symmetric, we have

$$H_{k-2} = A^{-1}(BH_{k-3} + DH_{k-4}) = H_{k-2}^T = H_{k-3}BA^{-1} + H_{k-4}DA^{-1},$$

$$H_{k-1} = A^{-1}(BH_{k-2} + DH_{k-3}) = H_{k-1}^T = H_{k-2}BA^{-1} + H_{k-3}DA^{-1}.$$

Then,

$$H_k = A^{-1}(BH_{k-1} + DH_{k-2})$$

$$= A^{-1}B(H_{k-2}BA^{-1} + H_{k-3}DA^{-1})$$

$$+ A^{-1}D(H_{k-3}BA^{-1} + H_{k-4}DA^{-1})$$

$$= A^{-1}BH_{k-2}BA^{-1} + A^{-1}(BH_{k-3}D +$$

$$DH_{k-3}B)A^{-1} + A^{-1}DH_{k-4}DA^{-1}.$$ 

Since each right-hand-side term is symmetric, $H_k$ must be symmetric. By induction, Lemma 7 is true.

### 5.3.2 Voltage Noise at Each Node.

Let $e_j$ be an $n \times 1$ vector of all 0s except the $j$th element being 1. Assume that $\mathbf{\tilde{I}}(\Delta t) = e_j$, and $\mathbf{\tilde{I}}(p\Delta t) = 0, \forall 2 \leq p \leq k$. Define $c_{j,k}$ as the vector of corresponding voltage noises at time $t = k\Delta t, \forall k \geq 1.$
According to (5.8), we get

\[ c_{j,k} \triangleq v(k\Delta t)|\hat{\mathbf{I}}(\Delta t) = e_j, \hat{\mathbf{I}}(p\Delta t) = 0, \forall 2 \leq p \leq k = H_k e_j. \]

Note that \( c_{j,k} \) is the \( j \)th column of \( H_k \). Since \( H_k \) is symmetric, its \( j \)th row is equal to \( c_{j,k}^T \). Applying this fact to (5.8), we can write the voltage noise of each node \( 1 \leq j \leq n \) at time \( t = k\Delta t, \forall k \geq 1 \) as

\[
v_j(k\Delta t) = \sum_{p=1}^{k} c_{j,p}^T \hat{\mathbf{I}}((k + 1 - p)\Delta t) = c_{j,1}^T \hat{\mathbf{I}}(k\Delta t) + c_{j,2}^T \hat{\mathbf{I}}((k - 1)\Delta t) + \cdots + c_{j,k}^T \hat{\mathbf{I}}(\Delta t),
\]

where the voltage noise is represented as a linear function of current excitations at different time steps.

Consider the power grid as an \( n \)-input-\( n \)-output LTI system with input current vector \( \hat{\mathbf{I}}(t) \) and output voltage noise vector \( \mathbf{v}(t) \). Conventionally, to represent a particular output as an affine function of inputs for such a linear system, we have to compute the impulse response of each input. However, because of the symmetry of \( H_k \) as stated in Lemma 7, the power grid has symmetric impulse responses. For example, let’s consider two nodes \( j_1 \) and \( j_2 \) shown in Fig. 5.6. We apply an impulse current excitation at one node, and evaluate the voltage noise response at the other node. These two nodes would have the same response due to symmetry. For each node, its voltage noise responses corresponding to the impulse current excitations at all the nodes are equal to the voltage noise responses of all the nodes corresponding to the impulse current excitation at the node itself. From the circuit perspective, this symmetry is due to the fact that all the R/L/C components are bidirectional and linear. It has been employed to formulate the voltage noise of each node as an affine function of current sources in (5.12), where \( c_{j,k} \) is the vector of voltage noise
responses of all the nodes at time \( t = k\Delta t \) corresponding to the impulse current excitation at node \( j \) when \( t = \Delta t \). Note that (5.12) can also be viewed as the convolution of impulse responses and inputs.

Summing up (5.12), we get

\[
\sum_{k=1}^{q} v_j(k\Delta t) = \sum_{k=1}^{q} \left( \sum_{p=1}^{k} c_{j,p} \right) \mathbf{b}^{T}((q + 1 - k)\Delta t). \]

Define

\[
\hat{c}_{j,k} \triangleq \sum_{p=1}^{k} c_{j,p} = \sum_{p=1}^{k} H_p e_j = v(k\Delta t)|_{\mathbf{I}(p\Delta t) = e_j, \forall 1 \leq p \leq k},
\]

then we have

\[
\sum_{k=1}^{q} v_j(k\Delta t) = \sum_{k=1}^{q} \hat{c}_{j,k}^{T} \mathbf{b}((q + 1 - k)\Delta t) \tag{5.13}
\]

\[
= \hat{c}_{j,1}^{T} \mathbf{b}(q\Delta t) + \hat{c}_{j,2}^{T} \mathbf{b}((q - 1)\Delta t) + \cdots + \hat{c}_{j,q}^{T} \mathbf{b}(\Delta t).
\]

It is to be noted that \( \hat{c}_{j,k} \) is the vector of voltage noises at \( t = k\Delta t \) corresponding to a constant current excitation \( \mathbf{I}(p\Delta t) = e_j, \forall 1 \leq p \leq k \).

5.3.3 Problem Decomposition. Applying (5.12) to represent the voltage noise, we can decompose the optimization problem (5.6) into the following two sub-problems.
For each node $1 \leq j \leq n$,

I: Compute $c_{j,k} = \mathbf{v}(k\Delta t)|_{\mathbf{i}(\Delta t) = e_j, \mathbf{I}(p\Delta t) = 0, \forall 2 \leq p \leq k}$, \hfill (5.14)

\[
\forall 1 \leq k \leq N_{ts};
\]

II: Maximize/Minimize

\[
v_j(N_{ts}\Delta t) = \sum_{k=1}^{N_{ts}} c_{j,k}^T \mathbf{I}((N_{ts} + 1 - k)\Delta t)
\]

subject to: $0 \leq \mathbf{I}(k\Delta t) \leq \mathbf{I}_L$, $\mathbf{U}(k\Delta t) \leq \mathbf{I}_G$,

\[
\mathbf{E}\mathbf{I}(k\Delta t) = 0, \sum_{k=1}^{N_{ts}} \mathbf{I}(k\Delta t) \leq \mathbf{I}_T.
\]

Moreover, the optimization problem (5.7) can also be decomposed similarly by using (5.13) as the objective function. For each node $1 \leq j \leq n$,

I: Compute $\mathbf{c}_{j,k} = \mathbf{v}(k\Delta t)|_{\mathbf{i}(\Delta t) = e_j, \mathbf{I}(p\Delta t) = 0, \forall 1 \leq p \leq k, \forall 1 \leq k \leq N_{ts}}$; \hfill (5.16)

II: Maximize/Minimize

\[
\sum_{k=1}^{N_{ts}} v_j(k\Delta t) = \sum_{k=1}^{N_{ts}} \mathbf{c}_{j,k}^T \mathbf{I}((N_{ts} + 1 - k)\Delta t)
\]

subject to the same set of constraints as stated in (5.15).

Clearly, the sub-problems (5.14) and (5.16) are power grid transient analysis problems with an impulse current excitation and a constant current excitation, respectively. Different from conventional power grid transient analysis with realistic current waveforms to evaluate voltage noises, we use impulse or constant current excitation for power grid simulation to characterize voltage noise responses. The sub-problems (5.15) and (5.17) are still linear programming (LP) but they are much easier to solve compared to (5.6) and (5.7), because the objective functions are formulated as linear functions of current excitations. Note that without transient constraints, the
Algorithm Vectorless Transient Verification

**Input:** power grid matrices $G, L, C$, current constraints in (5.15), $\Delta t$, $N_{ts}$.

**Output:** The worst-case voltage noises at each node.

1. **for** each node $1 \leq j \leq n$ **do**
2. Perform transient simulation with impulse current vector $\hat{I}(\Delta t) = e_j$ to compute $c_{j,k}$ in (5.14)
3. Solve the LP problems (5.15) to obtain the worst-case voltage noises in both directions
4. **end for**

Figure 5.7. The vectorless transient verification algorithm.

LP problems (5.15) and (5.17) can be further divided into many smaller LP problems at each time step and solved independently. In comparison with the exact approach of [1], this problem decomposition largely simplifies the vectorless verification of RLC power grids.

In addition, it is to be noted that the proposed problem decomposition is a generic approach for vectorless verification. Although the proofs of fundamental Lemma 6 and 7 are based on our nodal analysis equation (5.3) with trapezoidal rule, these lemmas are essential properties of the power grid as an LTI system, and are independent of specific discretization rules and system equations used. Therefore, $c_{j,k}$ and $\hat{c}_{j,k}$ can be computed by any accurate power grid transient analysis algorithms.

5.3.4 Methodology. Based on the problem decomposition, we propose to verify each node of the power grid by two orthogonal phases:

1. First, transient simulation.

2. Second, noise optimization.

We first perform transient simulation with impulse or constant current excitation to
compute $c_{j,k}$ or $\tilde{c}_{j,k}$ depending on the objective, and then solve two LP problems to evaluate the worst-case voltage noises in both directions. The resultant vectorless verification algorithm is summarized in Fig. 5.7. As a byproduct of solving the LP problems, the corresponding current waveforms leading to the worst-case voltage noises can also be obtained. Such current waveforms are important for designers as they serve as guidelines for grid modification. Therefore, the proposed methodology is capable of evaluating the worst-case voltage noises and identifying corresponding current waveforms.

The full-chip verification involves transient simulation and noise optimization for each node, thus being computationally expensive. However, compared to simulation-based approaches, which may need to enumerate infinite current excitations for theoretical guarantee, the proposed methodology verifies each node by performing transient simulation with a single current vector and solving LP problems. In practice, the proposed methodology can be applied to verify the risky regions of the power grid if full-chip verification is prohibitive.

More importantly, this methodology largely simplifies the vectorless verification problem, and relates power grid transient analysis to vectorless verification, so that existing power grid analysis algorithms can be leveraged. As the left-hand-side matrix of the system equation (5.3) is a symmetric $M$-matrix, it represents a resistor network, which can be obtained by converting the inductance and capacitance links into resistance branches accordingly. Then, the RLC power grid is reduced to a pure resistor network, which can be simulated very efficiently by using existing power grid analysis algorithms. In our implementation, we employ the preconditioned conjugate gradient (PCG) method [14, 27] with a random-walk based stochastic preconditioner [62] for fast power grid simulation. However, it is still very challenging to solve the LP problems for noise optimization, because practical power grids usually have a large
The Distribution of the Absolute Values of Coefficients

Figure 5.8. The distribution of the absolute values of coefficients computed for verifying a random node in an RLC power grid with about 1.1 million nodes ($\Delta t = 10$ ps, $N_{ts} = 100$).

number of current sources, and many time steps need to be evaluated for verification, resulting in prohibitively complicated LP problems.

5.4 Variable Reduction

To solve the LP problems efficiently, we propose to generate reduced-size LP problems with a user-specified error tolerance. The motivation and principles are introduced in Section 5.4.1 and 5.4.2, respectively. The algorithm details are presented in Section 5.4.3.

5.4.1 Motivation. Since computing the magnitude of voltage noise is the mainstream technique for vectorless verification, we consider solving the LP problems (5.15) to evaluate the worst-case voltage noises in both directions through noise maximization and minimization. The discussion is limited to the noise maximization problem, because the minimization problem can be converted to a maximization problem (by multiplying the objective by $-1$) and solved accordingly.

Using the definition of $\hat{I}(t)$ in (5.1), we can re-write (5.15) as a function of $I(t)$ with updated coefficient vectors. The zero-valued current sources should be
dropped\(^3\), so that the number of variables in (5.15) is equal to \(N_{ts} \times N_{cs}\), where \(N_{cs}\) is the number of actual current sources. To simplify the notation, let \(\mathbf{c}\) be the vector of all coefficients, \(\mathbf{I}\) be the vector of current variables at all time steps, \(\mathcal{I}_F\) be the feasible set of current excitations defined by the constraints in (5.15). Then the noise maximization problem in (5.15) can be represented as

\[
\text{Maximize } \mathbf{c}^T \mathbf{I}, \text{ subject to } \mathbf{I} \in \mathcal{I}_F.
\] (5.18)

As this LP problem usually has a large amount of variables for practical power grids, solving it directly with standard LP algorithms often takes huge amount of runtime. It is critical to simplify the LP problem by reducing the number of variables for efficient computation.

Fortunately, the grid locality can be exploited for variable reduction. There are two kinds of locality: (1) spatial locality, i.e. the voltage noise of a node is mainly dependent on the current sources in its neighborhood; (2) temporal locality, i.e. the voltage noise at a time point is mainly attributable to its recent current excitations. In practice, many coefficients have very small absolute values due to grid locality, so the corresponding current variables do not contribute much to the voltage noise. As shown by the example in Fig. 5.8, the absolute values of most coefficients are much smaller than \(10^{-6}\), removing the corresponding current variables from the objective function of the LP problem (5.18) would not introduce much error to the computed worst-case voltage noise, i.e. these current variables are insignificant for verification. However, simply ignoring such current variables during verification is not reliable, we need a conservative approach with accuracy guarantee for variable reduction.

5.4.2 Principles. In (5.18), both \(\mathbf{c}^T\) and \(\mathbf{I}\) can be partitioned (and reordered if

\(^3\)It is important to do so, because the runtime of standard LP solvers is typically dependent on the number of variables and constraints.
necessary) into two parts, such that

\[
c^T I = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}^T \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = c_1^T I_1 + c_2^T I_2,
\]

where \( I_1 \) is the vector of \textit{significant} current variables, \( I_2 \) is the vector of \textit{insignificant} current variables to be removed, \( c_1 \) and \( c_2 \) are the corresponding coefficient vectors. Assume that there is no equality constraints relating the current variables of \( I_1 \) and \( I_2 \), and consider the following two LP problems:

Maximize \( c_1^T I_1 \), subject to \( \begin{bmatrix} I_1 \\ 0 \end{bmatrix} \in I_F \), \hspace{1cm} (5.19)

Maximize \( c_2^T I_2 \), subject to \( \begin{bmatrix} 0 \\ I_2 \end{bmatrix} \in I_F \). \hspace{1cm} (5.20)

We call (5.19) the reduced-size LP problem as it can be viewed as a reduced-size version of (5.18). Let \((I^*, v^*)\) be the optimal solution and optimal value of (5.18), \((I_1^*, v_1^*)\) and \((I_2^*, v_2^*)\) be that of (5.19) and (5.20), respectively. Then the following lemma must hold.

\textbf{Lemma 8} \( v_1^* \leq v^* \leq v_1^* + v_2^* \).

\textbf{Proof 8} Obviously, \( v_1^* = c_1^T I_1^* = c^T \begin{bmatrix} I_1^* \\ 0 \end{bmatrix} \leq c^T I^* = v^* \). Moreover, since no equality constraints relates the current variables of \( I_1 \) and \( I_2 \), the optimal value of the following
**LP problem**

Maximize \[
\begin{bmatrix}
c_1 \\
0
\end{bmatrix}^T I, \quad \text{subject to } I = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \in \mathcal{I}_F,
\]

is equal to that of (5.19), so \[
\begin{bmatrix}
c_1 \\
0
\end{bmatrix}^T I^* \leq v_1^*.
\]

Similarly, we have \[
\begin{bmatrix}
0 \\
c_2
\end{bmatrix}^T I^* \leq v_2^*.
\]

Hence,

\[
v^* = \begin{bmatrix}
c_1 \\
0
\end{bmatrix}^T I^* + \begin{bmatrix}
0 \\
c_2
\end{bmatrix}^T I^* \leq v_1^* + v_2^*.
\]

Lemma 8 defines theoretical bounds of the maximum voltage noise. In practice, it is desired to identify the insignificant current variables \(I_2\), whose corresponding optimal value \(v_2^*\) is acceptably small, so that we can evaluate the worst-case voltage noise by solving the reduced-size LP problem (5.19) instead of the original LP problem (5.18).

We employ a user-specified error tolerance \(\delta_{lp}\) to control the accuracy of the computed worst-case voltage noise, and try to solve the following *variable reduction problem*:

Consider the LP problem (5.18), find the maximum set of insignificant current variables \(I_2\), such that \(v_2^* \leq \delta_{lp}\).

Then, by solving the reduced-size LP problem (5.19) to compute \(v_1^*\), we have

\[
v_1^* \leq v^* \leq v_1^* + \delta_{lp},
\]

where \(v_1^* + \delta_{lp}\) is reported as the conservative bound over the maximum voltage noise.

### 5.4.3 Details.

To solve the variable reduction problem, we need to check the
condition \( v_2^* \leq \delta \) for some \( I_2 \). However, solving (5.20) to compute the exact \( v_2^* \) would be very challenging because of the large number of variables involved. In order to find a proper set of insignificant current variables efficiently, we choose to compute an upper bound of \( v_2^* \) heuristically, and make sure that the computed upper bound is no larger than \( \delta \), so (5.21) still holds. Details follow.

We adopt non-overlapped equality constraints and hierarchical global constraints shown in Fig. 5.9, as well as local constraints and transient constraints. It can be seen that equality constraints and global constraints naturally divide the current sources (i.e., the current variables at each time step) into groups. Let \( b \) and \( m \) be the number of equality constraints and global constraints at each time step, respectively. There are a total of \( N_{ts} \times b \) equality constraints and \( N_{ts} \times m \) global constraints. Recall that there are \( N_{ts} \times N_{cs} \) current variables. Let \( I_p \) be the \( p \)th current variable. Define sets of indices \( S_k \) as

\[
S_k \triangleq \begin{cases} 
\{ p | I_p \text{ is restricted by the } k\text{th equality constraint} \}, \\
1 \leq k \leq N_{ts} \times b; \\
\{ p | I_p \text{ is restricted by the } (k - N_{ts} \times b)\text{th global constraint} \}, N_{ts} \times b + 1 \leq k \leq N_{ts} \times (b + m),
\end{cases}
\]

where \( S_k, N_{ts} \times b + 1 \leq k \leq N_{ts} \times (b + m) \) also includes indices of current variables of the VDD network, because the current sources in the VDD network are indirectly restricted by global constraints through equality constraints. Every \( S_k \) corresponds to a set of current variables, and no equality constraints relates the current variables of
Figure 5.9. Equality constraints and global constraints at each time step. Equality constraints relate the current sources in both VDD and GND networks, they are non-overlapped and defined for individual circuit block. Global constraints are specified hierarchically in the GND network to bound the total current of circuit block(s).

disjoint sets. For each $S_k$, rewrite the corresponding LP problem (5.20) as

$$\text{Maximize } v_k = \sum_{p \in S_k} c_p I_p, \quad (5.22)$$
$$\text{subject to } I_{|I_p=0, \forall p \notin S_k} \in \mathcal{I}_F,$$

where $v_k$ is the voltage noise caused by the corresponding set of current variables, $c_p$ is the $p$th coefficient. The other current variables $I_p, \forall p \notin S_k$ are set to 0, and then the constraints can be simplified by removing the zero-valued variables. Nevertheless, solving (5.22) for each $S_k$ with standard LP solvers still takes long runtime, because there are $N_{ts} \times (b + m)$ LP problems with complicated constraints (i.e., local, global, equality, and transient constraints).

For fast estimation of the optimal value of (5.22), we compute its upper bound by solving simplified LP problems, which are obtained by removing equality constraints, transient constraints, and higher level global constraints. For each
\( S_k, 1 \leq k \leq N_{ts} \times b, \)

Maximize \( v_k = \sum_{p \in S_k} c_p I_p, \) \hspace{1cm} (5.23)

subject to local constraints only;

for each \( S_k, N_{ts} \times b + 1 \leq k \leq N_{ts} \times (b + m), \)

Maximize \( v_k = \sum_{p \in S_k} c_p I_p, \) \hspace{1cm} (5.24)

subject to local constraints and hierarchical global constraints

which only restrict the current variables \( I_p, p \in S_k. \)

The solution of (5.23) can be easily obtained by setting \( I_p \) to the maximum value defined by local constraints if \( c_p > 0, \) and 0 otherwise. (5.24) can be efficiently solved by the sorting-deletion algorithm [18] as follows. Each \( I_p \) is set to 0 if \( c_p \leq 0; \) the other \( I_p \)'s with \( c_p > 0 \) are sorted, such that their corresponding \( c_p \)'s are in non-increasing order, and then they are set to the maximum feasible value defined by local constraints and hierarchical global constraints sequentially.

Let \( v_k^* \) be the optimal value of (5.22), \( \hat{v}_k \) be the optimal value of (5.23) and (5.24). For each set \( S_k, \) define the average noise bound per variable

\[ \overline{v}_k \triangleq \frac{\hat{v}_k}{|S_k|}. \] \hspace{1cm} (5.25)

Let \( U \) be the index set of insignificant current variables, \( \hat{v}_{total} \) be the aggregate noise bound of selected sets. The variable reduction algorithm can be summarized as shown in Fig. 5.10. In order to identify sets of insignificant current variables, it first computes \( \hat{v}_k \) and \( \overline{v}_k \) for all sets \( S_k, \) and then iteratively selects the set with the minimum \( \overline{v}_k \)
among all feasible candidate sets, so that the total number of identified insignificant current variables can be maximized. Note that the supersets of the selected set must be updated during each iteration to reflect the status change.

Lemma 9 The variable reduction algorithm identifies insignificant current variables with error tolerance $\delta_{lp}$, and the resulting reduced-size LP problem satisfies (5.21).

Proof 9 According to Lemma 8 and (5.21), we only need to show that $v^*_2 \leq \delta_{lp}$. The identified set of insignificant current variables is a collection of pairwise disjoint sets of variables, which are chosen iteratively. According to the termination condition of the variable reduction procedure (i.e., line 8 of Fig. 5.10), these sets satisfy

$$\sum_{\forall \text{identified sets } S_k \subseteq U} \widehat{v}_k \leq \delta_{lp},$$

and can be viewed as a partition of insignificant current variables $I_2$. By generalizing the proof of Lemma 8 for the partition of $I_2$, it follows that

$$v^*_2 \leq \sum_{\forall \text{identified sets } S_k \subseteq U} v^*_k \leq \sum_{\forall \text{identified sets } S_k \subseteq U} \widehat{v}_k \leq \delta_{lp}.$$

Lemma 10 The time complexity for computing all $\widehat{v}_k$ and $\widehat{v}_k$, $\forall 1 \leq k \leq N_{ts} \times (b + m)$ takes $O(N_{ts} \times N_{cs}(TotalLevel + \log N_{cs}) + N_{ts} \times (b + m))$, where TotalLevel is the total level of hierarchical global constraints. Each iteration for identifying sets of insignificant current variables takes $O(N_{ts} \times (b + m) + TotalLevel)$ time.

Proof 10 Solving (5.23) to compute $\widehat{v}_k$, $\forall 1 \leq k \leq N_{ts} \times b$ takes $O(N_{ts} \times N_{cs})$ time. The sorting-deletion algorithm [18] is employed to compute $\widehat{v}_k$, $\forall N_{ts} \times b + 1 \leq k \leq N_{ts} \times (b + m)$. It takes $O(N_{ts} \times N_{cs} \log N_{cs})$ time to sort the current variables, and
Algorithm Variable Reduction

**Input:** coefficients $c$ in (5.18), constraints in (5.15), $\delta_{lp}$.

**Output:** the index set of insignificant variables $U$.

1. $U \leftarrow \emptyset$
2. $\hat{v}_{\text{total}} \leftarrow 0$
3. // Compute $\hat{v}_k$ and $\overline{v}_k$
4. Compute $\hat{v}_k, \forall 1 \leq k \leq N_{ts} \times b$ by solving (5.23);
5. Compute $\hat{v}_k, \forall N_{ts} \times b + 1 \leq k \leq N_{ts} \times (b + m)$ by solving (5.24) with the sorting-deletion algorithm;
6. Compute $\overline{v}_k, \forall 1 \leq k \leq N_{ts} \times (b + m)$;
7. // Identify sets of insignificant variables iteratively
8. while $\exists$ set(s) $S_k \not\subseteq U$ and $\hat{v}_{\text{total}} + \hat{v}_k \leq \delta_{lp}$ do
9. Find one such set $S_{k*}$ with the minimum $\overline{v}_{k*}$;
10. $U \leftarrow U \cup S_{k*}$;
11. $\hat{v}_{\text{total}} \leftarrow \hat{v}_{\text{total}} + \hat{v}_{k*}$;
12. // Update supersets of the selected set $S_{k*}$
13. for each $S_k \supset S_{k*}$ do
14. $S_k \leftarrow S_k \setminus S_{k*}$;
15. $\hat{v}_k \leftarrow \hat{v}_k - \hat{v}_{k*}$
16. Recompute $\overline{v}_k$;
17. end for
18. end while
19. Return $U$;

Figure 5.10. The variable reduction algorithm.
Figure 5.11. Runtime break down of noise optimization using the proposed variable reduction algorithm. “Setup” denotes the runtime of the variable reduction procedure; “Solve” represents the runtime for solving the reduced-size LP problems.

\[ O(N_{ts} \times N_{cs} \times \text{TotalLevel}) \] time to compute the optimal value of (5.24). Based on the pre-computed \( \hat{\nu}_k \), \( \nu_k \) can be calculated in \( O(N_{ts} \times (b+m)) \) time. Hence, the complexity for computing \( \hat{\nu}_k \) and \( \nu_k \) follows.

There are a total of \( N_{ts} \times (b+m) \) candidate sets \( S_k \), and each set has a maximum of \( \text{TotalLevel} \) supersets. Therefore, in each iteration, it takes \( O(N_{ts} \times (b+m)) \) time to identify a proper set \( S_k^* \), and \( O(\text{TotalLevel}) \) time to update its supersets. Then Lemma 10 follows.

5.5 Experimental Results

The proposed vectorless verification approach has been implemented in C++, and the LP problems are solved by MOSEK [70], a general optimization software. To evaluate the performance, we generate integrated RLC power grids with 4 metal layers, 1.2V VDD, and various C4 bumps/chip sizes/power consumptions. For each power grid, we extract local constraints from the grid description, generate transient constraints and equality constraints, each of which includes about 100 current sources in the GND network as well as the VDD network, and specify up to 10 global constraints hierarchically. All experiments are carried out on a 64-bit Linux server with 2.67GHz Intel X5650 processor and 64GB memory. Although the processor has multiple cores, only a single core is used for experiments.
Table 5.1. Average Runtime per Node for Vectorless Verification

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>Simulation</th>
<th>Standard LP Solver</th>
<th>Proposed Variable Reduction Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Nodes</td>
<td>$^1 N_{cs}$</td>
<td>$^2$Time(s)</td>
</tr>
<tr>
<td>pg1</td>
<td>3088</td>
<td>1352</td>
<td>0.12</td>
</tr>
<tr>
<td>pg2</td>
<td>11768</td>
<td>5202</td>
<td>0.46</td>
</tr>
<tr>
<td>pg3</td>
<td>45928</td>
<td>20402</td>
<td>1.65</td>
</tr>
<tr>
<td>pg4</td>
<td>71408</td>
<td>31752</td>
<td>2.66</td>
</tr>
<tr>
<td>pg5</td>
<td>102488</td>
<td>45602</td>
<td>3.80</td>
</tr>
<tr>
<td>pg6</td>
<td>181448</td>
<td>80802</td>
<td>6.84</td>
</tr>
<tr>
<td>pg7</td>
<td>282808</td>
<td>126002</td>
<td>10.98</td>
</tr>
<tr>
<td>pg8</td>
<td>1125608</td>
<td>502002</td>
<td>45.61</td>
</tr>
</tbody>
</table>

1 the number of current sources;  
2 the runtime of power grid transient simulation;  
3 the runtime of noise optimization by solving the LP problem (5.15) directly with the standard LP solver (i.e., MOSEK);  
4 the average number of variables in the reduced-size LP problems;  
5 the runtime of noise optimization by using the proposed algorithm, including both variable reduction and solving the reduced-size LP problems;  
6 the speedup of the proposed algorithm relative to the standard LP solver (i.e., MOSEK) for noise optimization;  
7 MOSEK fails to solve the LP problem (5.15) for “pg8”.  

Table 5.2. Worst-case Voltage Noises of a Random Node With and Without Transient Constraints

<table>
<thead>
<tr>
<th>Power Grids</th>
<th>Node Type</th>
<th>Without Transient Constraints</th>
<th>With Transient Constraints</th>
<th>Overestimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>overshoot/ground bounce (mV)</td>
<td>voltage drop/undershoot (mV)</td>
<td>overshoot/ground bounce (mV)</td>
</tr>
<tr>
<td>pg1</td>
<td>ground</td>
<td>39.75</td>
<td>12.39</td>
<td>22.55</td>
</tr>
<tr>
<td>pg2</td>
<td>supply</td>
<td>18.28</td>
<td>30.63</td>
<td>18.20</td>
</tr>
<tr>
<td>pg3</td>
<td>ground</td>
<td>14.04</td>
<td>10.51</td>
<td>11.26</td>
</tr>
<tr>
<td>pg4</td>
<td>supply</td>
<td>10.46</td>
<td>13.65</td>
<td>10.31</td>
</tr>
<tr>
<td>pg5</td>
<td>ground</td>
<td>15.27</td>
<td>12.26</td>
<td>12.87</td>
</tr>
<tr>
<td>pg6</td>
<td>supply</td>
<td>17.45</td>
<td>19.98</td>
<td>17.30</td>
</tr>
<tr>
<td>pg7</td>
<td>ground</td>
<td>20.32</td>
<td>18.02</td>
<td>18.43</td>
</tr>
<tr>
<td>pg8</td>
<td>supply</td>
<td>21.17</td>
<td>23.13</td>
<td>NA</td>
</tr>
</tbody>
</table>
We apply the proposed variable reduction algorithm with different error tolerances $\delta_{lp} = 5, 10, 20\text{mV}$ to verify synthetic power grids with time step $\Delta t = 10\text{ps}$ and the number of time steps $N_{ts} = 100$, i.e., we evaluate the worst-case voltage noises within 1ns. For performance comparison, we also compute the exact worst-case voltage noises by solving the LP problem (5.15) directly with MOSEK. As MOSEK allows to choose between the simplex method and the interior point method to solve LP problems, we experiment with both options but only report the results using the interior point method as it is often faster.

Table 5.1 presents the average runtime per node, which is an estimation from solving 100 random nodes for each power grid because it is too time-consuming to verify all the nodes. The runtime can be generally partitioned into two parts: the runtime for power grid transient simulation, and the runtime for noise optimization. It can be seen that the power grid simulation time is very small as the random-walk based PCG method is fairly efficient, but solving the LP problem (5.15) directly with the standard LP solver (i.e., MOSEK) takes a large amount of runtime, thus being the performance bottleneck of vectorless verification. Fortunately, the proposed algorithm achieves significant speedups over the standard LP solver for noise optimization because of variable reduction, and the resulting runtime for noise optimization is close to the transient simulation time. In general, the proposed algorithm tends to become more effective as the grid size increases, because the grid locality can be better exploited in larger power grids. For each power grid, as the error tolerance increases, the proposed algorithm removes more insignificant current variables, and provides better performance. For example, the LP problems (5.15) for verifying “pg7” contains $N_{ts} \times N_{cs} = 100 \times 126002 \approx 12.6\text{M}$ current variables. With 5mV error tolerance, the reduced-size LP problems only have less than 0.6M current variables, achieving about 115× speedup. With 20mV error tolerance, the variable count further decreases to about 0.2M, leading to 264× speedup.
Moreover, for each power grid, it is observed that the runtime of the variable reduction procedure with different error tolerance values are approximately the same, because most time complexity of variable reduction is due to the computation of $\hat{v}_k$ as summarized in Lemma 10. Fig. 5.11 shows the runtime break down of noise optimization using the proposed algorithm. The runtime of variable reduction tends to become increasingly important as grid size increases. In fact, the variable reduction runtime increases monotonically as the grid size becomes larger. With larger error tolerance, its percentage increases because the reduced-size LP problems can be solved more efficiently due to less current variables. The runtime and speedup of noise optimization shown in Table 5.1 and Fig. 5.11 do not show a well-defined trend, because the runtime for solving the reduced-size LP problems varies case by case.

Compared with the approaches in [3] and [6], which compute bounds of voltage noises under DC current constraints, the proposed vectorless verification approach takes more runtime to calculate the exact worst-case voltage noises (with user-specified error margin) based on both DC and transient current constraints. This is due to the fact that our exact approach is more compute-intensive than computing bounds according to [3] and [6], especially with transient constraints. In comparison with the verification approaches with hierarchical power constraints in [18] and [75], the proposed approach takes similar runtime to setup the LP problems (by transient simulation), while it consumes more runtime to solve the LP problems, since our current constraints do not have strict hierarchical structure and the LP problems cannot be solved by the sorting-deletion algorithm. Hence, the variable reduction algorithm is proposed in order to solve the LP problems efficiently.

To demonstrate that omitting transient constraints may result in pessimistic voltage noise prediction, we perform experiments without transient constraints for comparison. Table 5.2 shows the worst-case voltage noises of a random node with
and without transient constraints. Note that each node has two worst-case voltage noises, i.e. overshoot/voltage drop of a supply node, ground bounce/undershoot of a ground node. An interesting phenomenon is that omitting transient constraints leads to significant percentage of overestimation for the voltage drop of a supply node and the ground bounce of a ground node, while it has minor impact on overshoots and undershoots. This phenomenon is attributable to the fact that the worst-case current waveforms for overshots and undershouts nearly satisfy transient constraints over the verification time interval. Generally, with transient constraints, we can get more realistic voltage noise estimations, thus avoiding costly overdesigns of the power grid.

In summary, the proposed variable reduction algorithm largely accelerates vectorless verification with transient current constraints, and can be applied for verifying practical RLC power grids. One might complain that the synthetic power grids are relatively small, and the per node runtime is large. However, our algorithm is important for at least three reasons. First, the size of the RLC power grid model is dependent on the level of model extraction. In practice, our algorithm can be applied either to parts of the power grid, or to the top-level network of the grid. Second, our algorithm can be easily parallelized since each node is verified independently. Third, as discussed in [18], one can choose to verify a few risky nodes of the grid, e.g., the nodes that are farthest from voltage sources. To take full advantage of the proposed algorithm for better performance, one can start with a large error tolerance value (e.g., 20mV), and then switch to a smaller error tolerance (e.g., 5mV) if higher solution accuracy is required. There would be a performance gain if only a small portion of nodes need to be verified with higher accuracy.
CHAPTER 6
CONCLUSION AND FUTURE WORK

In this thesis work, we have studied power grid verification in the following aspects:

- **Parallel Transient Simulation:** We developed a parallel power grid transient simulator, which incorporates two approaches for parallelizing forward and back substitution. The first approach calculates independent variables in parallel based on given L/U matrices, i.e., a post-factorization method; the second approach uses a node ordering generated by nested dissection for factorization or preconditioning, i.e., a pre-factorization method, which enables more efficient parallelization of forward and back substitution. The parallel performance may be further improved through code optimization, and new preconditioning techniques may be developed by considering parallel forward and back substitution. As the parallelization techniques based on the spatial decomposition of power grids have been much studied, future work would be to explore more efficient parallelization techniques through temporal decomposition. The idea is to divide the transient current waveforms into some time intervals, perform simulation with the current excitation of each time interval independently, and then sum up the voltage responses. The feasibility of this approach has been confirmed by our preliminary experiments.

- **Steady-State Verification:** We proposed three approaches for vectorless steady-state verification, including the dual bound algorithm, the hierarchical matrix inversion algorithm, and the constraint abstraction approach. Specifically, the dual bound algorithm computes upper bound of voltage noise, and achieves significant speedup over the standard LP solver in solving the LP problem of steady-state verification (even when there are large number of global
The hierarchical matrix inversion algorithm calculates the rows (or columns) of the inverse of the power grid matrix by exploiting the row (or column) dependencies, and largely reduces the computation cost of matrix inversion, making the steady-state verification more efficient. The constraint abstraction approach uses boundary constraints to model the boundary condition of subgrids, and enables efficient computation of conservative bounds of voltage noises in a divide-and-conquer manner. Since these algorithms target different applications, and exploit different trade-offs between performance and solution accuracy, future work would be to build a steady-state verification framework based on these algorithms and their extensions.

- **Transient Verification:** We studied vectorless verification of power grids using an integrated RLC power grid model. Our study showed that the vectorless verification of RLC power grids can be divided into two phases: power grid transient simulation and noise optimization. We proposed novel transient constraints to restrict the waveform of each current source for sign-off verification, so that the worst-case voltage noise estimations can be more realistic. Moreover, in order to solve the noise optimization problem efficiently, we designed a variable reduction algorithm to generate reduced-size LP problems with a user-specified error tolerance. Results showed that our algorithm significantly speeds up vectorless verification with transient current constraints, making the verification of large-scale RLC power grids possible. Our current work only consider the verification within a small time period, and it is of great interest to solve the vectorless transient verification problem for all time $t$ in future work. Moreover, since a power grid design can be verified by either power grid simulation or vectorless verification approaches, a comparison study of these two verification methodologies based on standard power grid benchmarks would an interesting future work.
BIBLIOGRAPHY


