Friendly Fast Poisson Solver Preconditioning Technique for Power Grid Analysis

Jianlei Yang, *Student Member, IEEE*, Yici Cai, *Senior Member, IEEE*, Qiang Zhou, *Senior Member, IEEE*, and Jin Shi

***Abstract*— Robust and efficient algorithms for power grid analysis are crucial for both VLSI design and optimization. Due to the increasing size of power grids, IR drop analysis has become more computationally challenging both in runtime and memory consumption. This paper presents a Fast Poisson Solver (FPS) preconditioned method for unstructured power grids with unideal boundary conditions. Unstructured power grids are transformed to structured grids, which can be modeled as Poisson blocks by analytic formulation. The analytic formulation of transformed structured grids is adopted as an analytic preconditioner for original unstructured grids, in which the analytic preconditioner can be considered as a sparse approximate inverse technique. By combining this analytic preconditioner with robust conjugate gradient method, we demonstrate that this approach is totally robust for extremely large scale power grid simulations. Theoretical proof and experimental results show that iterations of our proposed method will hardly increase with the increasing of grid size as long as the pads density and the distribution range of metal conductance value have been decided. We demonstrate that the run efficiency of our approach is much higher than classical incomplete Cholesky factorization preconditioned conjugate gradient solver and random walk-based hybrid solver.**

***Index Terms*— Fast Poisson Solver (FPS), power grid analysis, preconditioning.**

I. Introduction

As VLSI manufacturing process advances beyond 45 nm, large digital systems can integrate more than several billion transistors. Also with the increasing of working frequency, the power consumption of a digital system increases exponentially. Power grid network distributes power and ground voltages from voltage sources to chip cells, but it will cause IR drops on the on-die power grid because of the metal resistance and the di/dt drops due to the inductance effect of the chip package. With technology advances, this effect becomes more and more significant. Excessive voltage drops in the power grid reduce switching speeds and noise margins of circuits, leading to functional failures. Hence, power delivery integrity verification is critical for silicon success.

The design and analysis of extremely large scale power grids are challenging tasks for VLSI design. A critical issue of power grid analysis is the large size of the network. For modern integrated circuits, such as microprocessors, such a network can easily include millions of nodes and even one billion nodes. Many contributions have been developed to run power grid simulations, including direct solvers and iterative solvers. Direct solvers are robust but not adequate for the tremendous amount of power grid nodes because of CPU speed and memory limitation. Iterative solvers are more memory efficient but unstable because of performance limitation by preconditioners [1]. In particular, there are some PDE-like solvers, such as random walk [2], multigrid methods [3], domain decomposition methods [4], and hierarchical methods [5], matrix techniques-based solvers, such as SPAI [6] and $\mathcal{H}$-Matrix [7]. However, directly using these methods reveal certain weaknesses in either efficiency or robustness when addressing very large industrial designs. Thus, sometimes all of these methods can be approximately used as preconditioners for iterative methods to improve the robustness. However, these iterative-related methods may hardly obtain an optimal convergence for extremely large scale power grids.

In addition, several special characteristics of power grids, such as locality effect [8] and pattern phenomenon [9], have been adopted to accelerate their solvers. There are also some closed-form expressions and related algorithms for fast power grid analysis [10], [11]. Recently, there has been quite increasing interest in parallel computation for power grid simulation. In [12], a friendly Fast Poisson Solver (FPS) using GPU-based FFT acceleration is proposed as an analytic direct method for solving 2-D structured power grids with the computation complexity of $\mathcal{O}(\mathcal{N}\log\mathcal{N})$. The highlight of the work is the proposed concept of Poisson block, which extends the locality effect of grid shell to more wide and practical cases. However, the FPS is mostly suitable for highly structured grids, which may limit its practical applications in general grids. Essentially, the closed-form expressions in [10] and [11] are somewhat equivalent to the analytic FPS in [12]. In [13], the FPS is adopted as an analytic preconditioner for unstructured power grid analysis. In [14], a HMD algorithm is proposed to solve 3-D irregular power grids. And in [15], a MGPCG solver is proposed to improve the robustness of HMD solver. However, the most noticeable problem with the above GPU-Multigrid solvers is the mapping method of 3-D irregular grids to 2-D regular grids, which ignores via resistances. Obviously it will result in considerable errors therefore slow convergence. In particular, as the industrial CMOS process become increasingly advanced, these approaches converge

Manuscript received April 29, 2012; revised December 3, 2012; accepted February 20, 2013. Date of publication April 18, 2013; date of current version March 18, 2014. This work was supported by the National Natural Science Foundation of China under Grant 61274031.

The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yjl09@mails.tsinghua.edu.cn; caiyc@mail.tsinghua.edu.cn; zhouqiang@mail.tsinghua.edu.cn; maplesj@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2252375

very slowly for power grids with unbalanced conductance distribution; that is, the resistance of vertical metal via is much larger than horizon metal wires.

In this paper, we propose a FPS preconditioned iterative method for general grids with unideal boundary conditions. In this approach, multilayer power grid is modeled as several single metal layers by treating vias as current sources. Then each metal layer is transformed into a structured grid so that all metal layers can be modeled by analytic formulation. Furthermore, by taking advantage of the analytic formulation of transformed grids, we propose an analytic preconditioner which can be considered as a sparse approximate inverse technique. Finally, an efficient and robust FPS preconditioned conjugate gradient (FPS-PCG) method is introduced to solve the original unstructured grids. By theoretical analysis on the sparse approximate inverse technique, we prove that this analytic preconditioner is very close to an exact inverse, which is only influenced by the distribution range of metal resistances value. Due to the certain regularity of real power grid designs, this distribution range is often so small that a good preconditioning performance can be guaranteed. Beneficially, we prove that the iterations will hardly increase with the increasing of grid size as long as the distribution range has been decided.

This paper is organized as follows. Section II presents the power grid analysis background and brief introduction of our proposed approach. Section III provides the analytic formulation for power grids. Section IV is the efficient implementation of proposed FPS-PCG method. Experimental results on large scale power grids are shown in Section V. Concluding remarks are given in Section VI.

II. Backgrounds and Overview

A. Power Grid Analysis Background

For DC simulation, power grid can be modeled as linear resistive network. By using the modified nodal analysis method, an n-node circuit network can be formulated as the following linear system equation [1]:

$$GV = I \tag{1}$$

where the conductance matrix $G \in \mathbb{R}^{n\times n}$ is a symmetric positive definite (s.p.d.) matrix, which represents the interconnecting relationship and resistor values, $V \in \mathbb{R}^{n\times 1}$ is an unknown vector of node voltages and $I \in \mathbb{R}^{n\times 1}$ is an input vector of node current sources. As the VLSI technology scaling associated with significantly increasing device numbers in a die, the number of nodes in the power grid may easily exceed many millions. The most accurate and stable methods for solving such huge linear systems are sparse direct solvers, such as SuperLU and Cholmod, but both of them are time expensive and memory inefficient. Another state-of-the-art approach is iterative methods, especially preconditioned iterative methods, which can be used to solve such linear systems with memory efficiently. However, preconditioned iterative methods are not stable in many cases because of either expensive cost or unsatisfactory performance of their preconditioners. There are also some fast and robust solvers for special regular and structured power grids, such as FPS [12], but its regularization limits its practical applications in general power grids. To overcome this limitation, we extend this fast solver on general power grids with little acceptable cost.

B. Prior Works and Proposed Approach

The most popular approach for solving such a symmetric positive definite system is conjugate gradient method, which is memory efficient for large scale problems. Regarding the convergence property of CG method, it can be shown that the required number of iterations can be bounded in terms of the spectral condition number [16]

$$\kappa_2(G) = \|G\|_2 \left\|G^{-1}\right\|_2 = \sqrt{\frac{\lambda_{\max}\left(G^T G\right)}{\lambda_{\min}\left(G G^T\right)}} \tag{2}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are maximum eigenvalue and minimum eigenvalue, respectively. That is to say, the convergence of CG method is affected by $\sqrt{\lambda_{\max}/\lambda_{\min}}$. For more detailed demonstration, please refer to Section IV-E.

An initial planning power grid is highly regular. The metal width and pitch have very small variations throughout the same metal layer but are totally distinct in different metal layers. Even though the grid is gradually modified, the typical distribution is affected so slightly that the regularity can still be exploited to improve the numerical characteristics. As shown in (2), the entries distribution of matrix G dominates the eigenvalues distribution, and consequently the condition number. It is obvious that large variations in conductance among different metal layers or vias will directly lead to a slow convergence rate of iteration methods. With the industrial CMOS process becoming more advanced, via resistance value exceeds the sheet resistance by several orders of magnitude. Thus, the numerical characteristic will be intensively affected by vias. All these unbalanced distributions cause the max-to-min ratio of eigenvalue to become larger.

On the other hand, aiming to take advantage of the geometry multigrid, the MGPCG method [14] was proposed for power grid analysis by compressing 3-D grid to 2-D grid but ignoring via resistance. Obviously this strategy will result in considerable error and hence slow convergence. In particular, for unbalanced conductance distribution which means that the resistance of vertical metal via is much larger than horizon metal wires, the above effect will be enlarged. And consequently the performance of this approach will be degraded.

Since neither ICCG nor MGPCG is a smart choice, we may draw some advantages from the divide and conquer strategy. Here, we use FPS as an analytic preconditioner for conjugate gradient method to handle general unstructured grids with unideal boundary conditions. For clarity, the flow of the proposed approach is shown in Fig. 1.

The original multilayer power grid is modeled as several single metal layers by treating vias as current sources. Because of grid regularity, the vias resistance almost never changes between two certain metal layers. For a certain metal layer, the width of all metal slices is calculated and the average value of them is adopted as the typical width of this metal layer. Then each metal layer is transformed into a structured grid with the typical width so that all metal layers can be modeled by

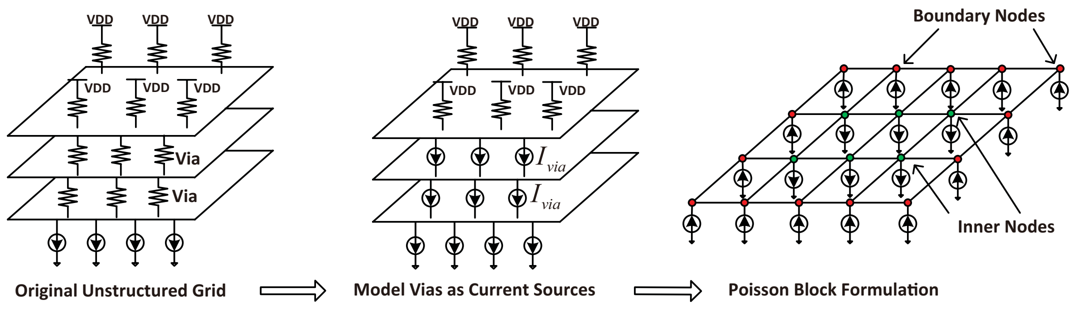

VDD
Via
I_{via}
Boundary Nodes
Inner Nodes
Original Unstructured Grid
Model Vias as Current Sources
Poisson Block Formulation

Fig. 1. Overall power grid analysis flow.

analytic formulation. By analytic formulation, the grids can be compressed as a Poisson block, which can be solved by FPS. This FPS is adopted as an analytic preconditioner, which can be considered as a sparse approximate inverse technique to accelerate the iterative method for original unstructured grids. The most advantage of this idea is that we just need to handle the single metal layer independently while the resistance value has very small variations throughout each metal layer. Thus, a good convergence rate is achieved. In summary, the convergence and efficiency of the proposed approach depend on the efficiency of FPS-PCG solver. We will demonstrate that our approach largely improves the robustness of power grid analysis.

III. Analytic Formulation

A concept of Poisson block was introduced in [12] to explore the particularity of power grid. The Poisson block corresponding to a finite difference discretization of a continuous Poisson problem on a 2-D rectangular homogeneous domain with Dirichlet conditions can be solved by FPS perfectly. For more general grids with unideal boundary conditions, new approach should be considered.

A. Poisson Block on Power Grid

The locality characteristic of power grid has been explored by many existing works [8], [17]–[19]. Locality means that the absorbing current sources only can trigger voltage drop within a local area around it. And beyond this area, the triggered voltage drop will attenuate to zero very fast. The work in [8] first proposes the concept of grid shell for large scale power grid with Flip-Chip package. The whole grid is partitioned as many sub-grids according to grid shell, which is not accurate but possible to be exploited for parallel power grid analysis. However, no obvious rule is proposed to determine the grid shell in [8]. Also, for Wire-Bond package type, grid shell is more ambiguous. Hence, the concept of Poisson block was proposed in [12] based on several definitions and assumptions, which can be easily satisfied by modern industrial chip designs. A Poisson block stands for a regular grid area, which satisfies three conditions: 1) this area does not contain any pads; 2) this area contains only current sources as active elements; and 3) the current flowing through the boundary is small enough. Power grid with Wire-Bond package can satisfy the definitions of Poisson block naturally. For Flip-Chip package model with locality property, pads are distributed

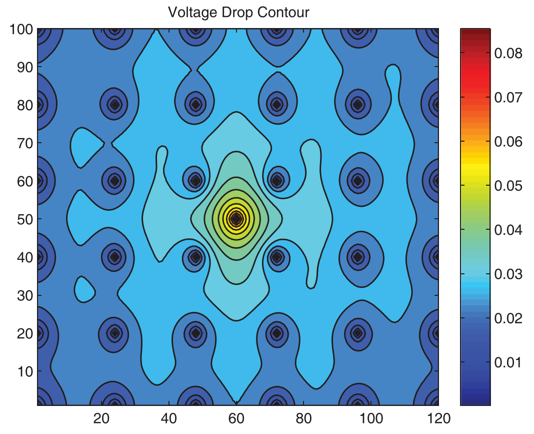


Fig. 2. Locality effect on Flip-Chip (a strong current source attached to the center node of a sample 100 $\times$ 120 power grid with 6 $\times$ 6 pads array).

evenly in the grid and they act as strong current drain points, which prevent the current from flowing far away from its source point. Thus, the local area among nearest pads in power grid with Flip-Chip package can also be treated as Poisson block.

Take a small grid with Flip-Chip package as an example, a strong current source is attached to the center node of a 100 $\times$ 120 grid with uniformly distributed 6 $\times$ 6 pads array while small current sources are attached to other nodes. Fig. 2 shows the voltage drop distribution triggered by this strong current source. The area that contains dominant voltage drop is around the center part of the grid. Meanwhile, the voltage drop decreases dramatically with the increasing distance from the center node. In particular, for the area outside the four nearest pads array, the voltage drop is affected slightly by the strong current source. For the area outside the second nearest pads array, the voltage drop is even much smaller. Hence, the area bounded by the nearest pads array can be considered as a local area. It is easy to find that it satisfies the definitions of Poisson block approximatively.

B. Fast Poisson Solver

Equation (1) is formulated as sparse format of matrix G while each row represents each circuit node and each entry of unknown voltage vector represents each node voltage [1]. Unlike the above traditional formulation, a dense matrix is adopted to describe the node voltages in this paper. With another two interesting matrix, Kirchhoff's Law can be satis-

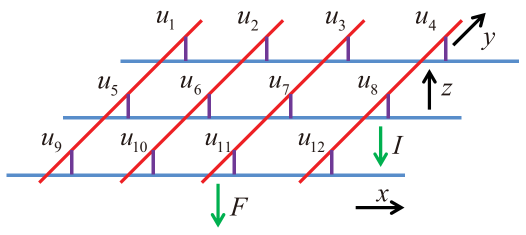


Fig. 3. Topology of a small power grid with two metal layers.

fied correctly. Considering a small power grid with two metal layers, the 3×4 grid nodes are shown in Fig. 3.

For the top metal layer in y-direction, voltage pads are attached to the boundary nodes. And for bottom metal layer in x-direction, current sources are attached to crossed nodes. Also the vertical metal segments in z-direction are modeled as metal vias. We suppose all metal stripes at bottom metal layer to have the same resistance r_1 and top metal layer to have the same resistances r_2 and vias to have the same resistances R because of the grid regularity. Then, we can use a dense matrix U_1 to describe the voltage of grid nodes at bottom metal layer and U_2 for top metal layer. Each entry in U_1 and U_2 represents voltage of each node

$$U_1 = \begin{pmatrix} u'_1 & u'_2 & u'_3 & u'_4 \\ u'_5 & u'_6 & u'_7 & u'_8 \\ u'_9 & u'_{10} & u'_{11} & u'_{12} \end{pmatrix}$$

$$U_2 = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 \\ u_5 & u_6 & u_7 & u_8 \\ u_9 & u_{10} & u_{11} & u_{12} \end{pmatrix}.$$

The current source distribution can be formulated as matrix F, in which each entry represents each current source loading

$$F = \begin{pmatrix} f_1 & f_2 & f_3 & f_4 \\ f_5 & f_6 & f_7 & f_8 \\ f_9 & f_{10} & f_{11} & f_{12} \end{pmatrix}.$$

Particularly, metal vias are modeled as current loadings from top metal layer to bottom metal layer which can be formulated as matrix I, in which each entry represents the current flowing on each via

$$I = \begin{pmatrix} i_1 & i_2 & i_3 & i_4 \\ i_5 & i_6 & i_7 & i_8 \\ i_9 & i_{10} & i_{11} & i_{12} \end{pmatrix}.$$

With two tridiagonal matrix T_1 and T_2, the Kirchhoff Current Law for top metal layer and bottom metal layer can be represented by two matrix equations

$$\frac{T_2 \cdot U_2}{r_2} = I, \qquad \frac{U_1 \cdot T_1}{r_1} = F - I \tag{3}$$

$$T_1 = \begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -1 \end{pmatrix}, \qquad T_2 = \begin{pmatrix} -1 & 1 & \\ 1 & -2 & 1 \\ & 1 & -1 \end{pmatrix}. \tag{4}$$

Meanwhile, there is an obvious relationship of current flowing on vias from top metal layer to bottom metal layer

$$\frac{U_2 - U_1}{R} = I. \tag{5}$$

By eliminating the matrix I from (3) and (5), we can obtain

$$U_1 = U_2 - \frac{R}{r_2} \cdot T_2 \cdot U_2. \tag{6}$$

As shown in (6), the solution of U_1 can be directly obtained by substitution as long as U_2 has been solved. Aiming to obtain the solution of U_2, U_1 can be eliminated from (3), (5), and (6). Then we can obtain

$$\frac{U_2 \cdot T_1}{r_1} + \frac{T_2 \cdot U_2}{r_2} - \frac{R}{r_1 r_2} \cdot T_2 \cdot U_2 \cdot T_1 = F. \tag{7}$$

If we define a matrix D_2 to represent the voltage drop of grid nodes on top metal layer and the standard supply voltage is denoted as Vcc, we can obtain

$$U_2 = Vcc \cdot E - D_2 = V - D_2 \tag{8}$$

where E is unit matrix. Equation (7) can be represented as

$$\frac{(V - D_2) \cdot T_1}{r_1} + \frac{T_2 \cdot (V - D_2)}{r_2} - \frac{R}{r_1 r_2} \cdot T_2 \cdot (V - D_2) \cdot T_1 = F. \tag{9}$$

Notice that $V \cdot T_1 = 0$, $T_2 \cdot V = 0$ and $T_2 \cdot V \cdot T_1 = 0$ due to the special characteristic of T_1 and T_2, and by sufficiently exploiting the properties of Poisson block, where the boundary voltage drop of a Poisson block is equal to zero, the final matrix equation can be obtained

$$F = \frac{D_2 \cdot P_1}{r_1} + \frac{P_2 \cdot D_2}{r_2} + \frac{R}{r_1 r_2} \cdot P_2 \cdot D_2 \cdot P_1 \tag{10}$$

$$P_1 = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{pmatrix}. \tag{11}$$

The matrix P is typical tridiagonal Toeplitz matrix, which has analytic eigendecomposition $P = z \cdot \Delta \cdot z^T$, where z is a symmetry orthogonal dense matrix and Δ is a diagonal matrix, which is given by

$$z(i,j) = \sqrt{\frac{2}{n+1}} \sin\left(\frac{i \cdot j \cdot \pi}{n+1}\right)$$
$$\Delta(i,i) = 2\left(1 - \cos\frac{i \cdot \pi}{n+1}\right). \tag{12}$$

Let $X_2 = z_2^T \cdot D_2 \cdot z_1$, substitute this analytic eigendecomposition to (10), remembering that matrix z_1 and z_2 are symmetry orthogonal matrices. The system matrix equation can be expressed as an analytic form

$$X_2 = \left(z_2^T \cdot F \cdot z_1\right) \odot W$$
$$W(i,j) = \left(\frac{\Delta_1(j,j)}{r_1} + \frac{\Delta_2(i,i)}{r_2} + \frac{R}{r_1 r_2} \cdot \Delta_1(j,j) \cdot \Delta_2(i,i)\right)^{-1} \tag{13}$$

where the operator $\odot$ means that the result matrix in brackets performs Hadamard matrix multiplication with matrix W. Finally, we can obtain the analytic voltage drop solution as shown below

$$D_2 = z_2 \cdot X_2 \cdot z_1^T = z_2 \cdot [(z_2 \cdot F \cdot z_1) \odot W] \cdot z_1. \tag{14}$$

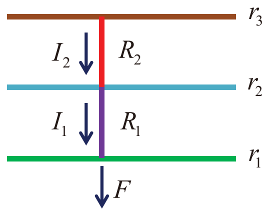


Fig. 4. Demonstration of a power grid with three metal layers.

Consequently, the voltage drop distribution D_2 of the top metal layer can be obtained as long as the current loadings distribution matrix F is known. Finally, the voltage drop distribution D_1 of bottom metal layer can be obtained by substitution as shown in (6).

Similar analytic solution can be obtained in the same manner for multilayer power grid analysis. As shown in Fig. 4, a power grid with three metal layers is taken as an example for clarity, where r_1, r_2, and r_3 are to represent the typical resistance of each metal layer, matrix F is to represent the current source loadings for transistors, R_1 and R_2 are to represent the typical resistance of vertical metal vias, which are modeled as the current source matrix I_1 and I_2. By the same manner, the voltage solution X_3 of the top metal layer can be first obtained. And then the voltage solution X_2 of the middle metal layer can be obtained by substitution. Finally, the voltage solution X_1 of the bottom metal layer can also be obtained by substitution. The only distinction we have to emphasize is we need to reformulate the dense matrix W

$$X_3 = \left(z_2^T \cdot F \cdot z_1\right) \odot W$$

$$W(i,j) = \begin{pmatrix} \frac{\Delta_1(j,j)}{r_1} + \frac{\Delta_2(i,i)}{r_2} + \frac{\Delta_2(j,j)}{r_3} \\ +\frac{R_1}{r_1 r_2} \cdot \Delta_1(j,j) \cdot \Delta_2(i,i) \\ +\frac{R_2}{r_2 r_3} \cdot \Delta_2(i,i) \cdot \Delta_3(j,j) \\ +\frac{R_1+R_2}{r_1 r_3} \cdot \Delta_1(j,j) \cdot \Delta_3(j,j) \\ +\frac{R_1 R_2}{r_1 r_2 r_3} \cdot \Delta_1(j,j) \cdot \Delta_2(i,i) \cdot \Delta_3(j,j) \end{pmatrix}^{-1}. \tag{15}$$

Thus, the solution of middle and bottom metal layer can be obtained by substitution as long as the voltage distribution of the top metal layer is solved.

C. Simulation Method for Multilayer Power Grid

As demonstrated in Section II-B, the numerical characteristics are intensely sensitive to the distinction of metal resistance value among different metal layers and vias. For physical design of modern chips, the metal width and pitch have very small variations throughout the same metal layer but totally distinct in different metal layers. This characteristic will lead to a bad condition system, which will converge slowly for iterative solvers if we handle the whole power grids together. By exploiting the analytic formulation of multilayer power grids, each single metal layer can be solved independently to avoid solving the bad condition system because the resistance value of each metal wire has very small variations throughout each metal layer.

First, each metal layer is transformed as a structured metal layer and then these several structured metal layers can be compressed as a single metal layer by analytic formulation as shown in (15). The proposed FPS method [12] can be adopted to obtain the node voltage distribution on top metal layer efficiently. The voltage distribution of other metal layers can be obtained by substitution since the voltage solution on top metal layer has been known. Thus, the whole power grid analysis can be performed by analytic formulation, which is very efficient and robust for large scale problem size.

D. Poisson Block on Different Package Types

Since the analytic solution of voltage drops has been obtained, we have to reconsider the existence of Poisson block both in Flip-Chip package and Wire-Bond package. As revealed in [8], the current on the grid with regular pads distribution is localized among vias/pads regardless of whether what package type is used. For irregular pads distribution, the definitions of the Poisson block cannot be satisfied because the current flowing through the grid edge is not tiny enough. That is to say, the Poisson block is only located between pad shell where the current flowing through the boundary is extremely small. The shape of the Poisson block will be different as long as the pads distribution has been changed. If the power grid is still partitioned as Poisson blocks in the manner of regular approach and then solved independently, some relative solution error will be introduced due to the unsatisfied boundary conditions.

The power grid with Wire-Bond package can be formulated as a naturally approximate Poisson block because the pads are only connected to the boundary nodes. And for Flip-Chip package with regular pads distribution, there exist many small approximate Poisson blocks where each Poisson block is located at each neighboring pads array. But for Flip-Chip package with irregular pads distribution, more considerations should be taken to identify the certain Poisson blocks. Take a power grid with 140×230 nodes and 11×13 pads array as an example, several certain pads are removed from the array due to the irregularity. Its voltage drop contour is shown in Fig. 5. As shown in Fig. 5(a), the voltage drop of the grid area with several pads removed is larger than the other grid area. It is necessary to identify each proper Poisson block for this kind of power grid with irregular pads distribution.

Among the grid area with regular pads distribution, each sub-grid area surrounded by each four neighbor pads can be modeled as a small Poisson block due to the locality effect. And among the grid area with several pads removed, the sub-grid area surrounded by several neighbor pads can also be modeled as a larger Poisson block. As shown in Fig. 5(b), the Poisson block PB1 is surrounded by 12 pads, PB2 is surrounded by eight pads, and PB3 is surrounded by ten pads. These larger Poisson blocks are different from the small Poisson block surrounded by four pads. Thus, there are many Poisson blocks with different shape and size in power grid with irregular pads distribution. These Poisson blocks can be adopted as different analytic preconditioners. Even though the shape of Poisson block may not be a rectangle, it can be

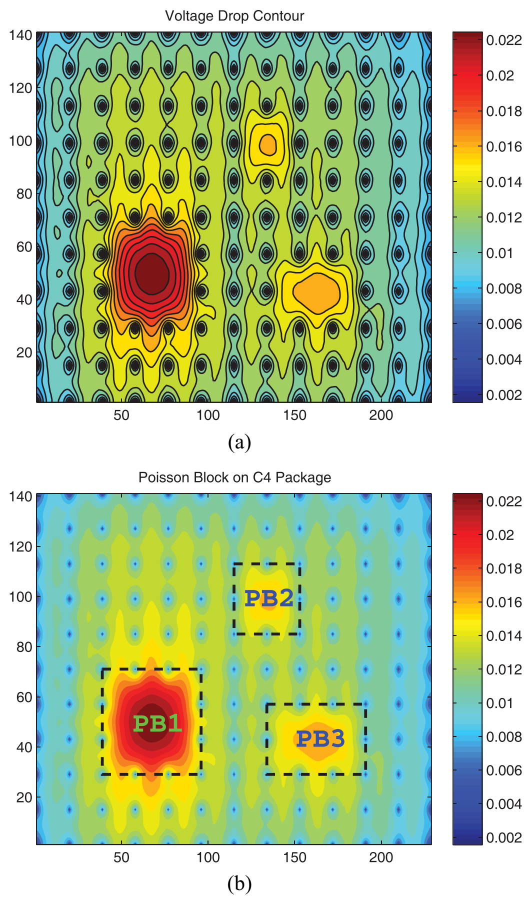


Fig. 5. Poisson blocks on irregular pads distribution with Flip-Chip package. (a) Voltage drop on power grid with Flip-Chip package. (b) Larger Poisson blocks area.

partitioned as several minimum rectangles which can cover the grid area without pads. And then by treating each rectangle above as an approximate Poisson block, a preconditioner can be constructed for each Poisson block as well. Even though these Poisson blocks are obtained approximately, we just adopt them as preconditioners for iterative methods as long as their convergence can be properly guaranteed.

As described in (14), the current source matrix F contains a number of entries which are resulted by pad nodes but the current drawn on each pad is unknown. In this paper, the pad node is modeled as a voltage source Vcc with a resistance in series. By Norton's Theorem, the pad model can be transformed as an equivalent circuit of a current source with a resistance in parallel. Once the current drawn by pads has been directly obtained, we just need to update the conductance matrix G with resistance of pad and to update the current source matrix F with equivalent independent current source. Essentially, however, the unideal boundary conditions on Poisson block will introduce corresponding solution errors due to the asymmetric pads distribution of Wire-Bond package or Flip-Chip package. We will demonstrate that the solution error of this approach is mainly located near the pad nodes. Within several iterations, the solution error will be cut down to a satisfactory accuracy level.

IV. FPS-PCG Solver

For large scale linear problems, the classical Krylov-subspace iterative methods lack for fast convergence so preconditioning technologies are developed. However, it is difficult to obtain a good preconditioner because of either solution accuracy lost or badly memory footprint. Some effective preconditioners are based on deep insight into the structure of the problem. For partial differential equations, where it is shown that certain discretized second-order elliptic problems on simple geometries can be very well preconditioned with FPS [20]. A domain-decomposed FPS on a rectangle was developed for parallel implementation in [21]. Several preconditioners involved by fast Fourier transform were proposed in order to improve the convergence of iterative methods [22]. The effectiveness of such a preconditioner has been analyzed in [23] and some of the many ways to implement the solver efficiently are discussed in [24]. Similarly in VLSI physical design community, a class of eigendecomposition-based FPS is proposed for chip level thermal analysis [25]. The proposed FPS method can leverage the preconditioned conjugate gradient method to solve non-rectangle 3-D domains with mixed boundary conditions efficiently. All of these approaches can be regarded as a very strong demonstration for adopting FPS as an analytic preconditioner for iterative methods. Since the power grid analysis and the thermal grid analysis belong to discretized second-order elliptic problems, this similarity can be exploited to adopt FPS as a preconditioner for power grid analysis.

A. FPS-Preconditioning Method

For clarity, the structured power grid and unstructured power grid are first compared without considering metal vias. As shown in [12], the structured power grid can be formulated as a standard Poisson block. If we define

$$\Sigma_1 = g_x \cdot I_1 \quad \Sigma_2 = g_y \cdot I_2 \tag{16}$$

where $\Sigma_1 \in \mathbb{R}^{m\times m}$, $\Sigma_2 \in \mathbb{R}^{n\times n}$ are diagonal matrix and $I_1 \in \mathbb{R}^{m\times m}$, $I_2 \in \mathbb{R}^{n\times n}$ are identity matrix, g_x is the metal sheet conductance value in x-direction (row direction), and g_y is the metal sheet conductance value in y-direction (column direction) on the power grid mesh. In the same manner of general discretization for Poisson equation on a rectangle with the Dirichlet stencil, the conductance matrix $G \in \mathbb{R}^{mn\times mn}$ can be formulated as

$$G = \Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2 \tag{17}$$

where the operator $\otimes$ is to perform Kronecker product [26]–[28], $P_1 \in \mathbb{R}^{m\times m}$ and $P_2 \in \mathbb{R}^{n\times n}$ are typical tridiagonal Toeplitz matrices, which have analytic eigen decompositions as shown in (12). It is easy to find that the above sparse formulation is equivalent to the dense form

$$P_1 \cdot U + U \cdot P_2 = F \tag{18}$$

which is similar to the formulations in Section III-B.

In the same manner, the conductance matrix of unstructured power grid can also be formulated as

$$G' = \alpha \otimes P_2 + P_1 \otimes \beta \tag{19}$$

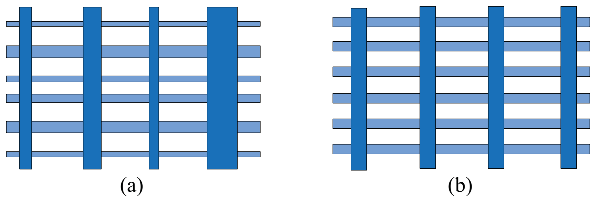


Fig. 6. Grid transformation. (a) Original unstructured grid. (b) Transformed structured grid.

where $\alpha \in \mathbb{R}^{m\times m}$ is a diagonal matrix $\operatorname{diag}(\alpha_1, \alpha_2 \ldots, \alpha_m)$ whose entries α_i $(i = 1, 2, \ldots, m)$ represent the metal sheet conductance values of each row (x-direction), $\beta \in \mathbb{R}^{n\times n}$ is a diagonal matrix $\operatorname{diag}(\beta_1, \beta_2 \ldots, \beta_n)$ whose entries β_j $(j = 1, 2, \ldots, n)$ represent the metal sheet conductance values of each column (y-direction). Also it can be reformulated as dense form

$$P_1 \cdot U \cdot \beta + \alpha \cdot U \cdot P_2 = F. \tag{20}$$

Due to α and β are diagonal matrices, we can reformulate it as

$$\alpha^{-1} \cdot P_1 \cdot U + U \cdot P_2 \cdot \beta^{-1} = \alpha^{-1} \cdot F \cdot \beta^{-1}. \tag{21}$$

This dense form formulation is same as Sylvester Equation, which is commonly encountered in control theory. Sylvester Equation is a kind of matrix equation, which can be represented as $AX + XB = C$, where A, B, X, and C are $n \times n$ matrices, A, B, and C are known, and the problem is to find a solution of X. A classical algorithm for the numerical solution of Sylvester equation is Bartels–Stewart algorithm [29]. It first transforms A and B into Schur form by a QR algorithm, and then solves the resulting triangular system by back-substitution. The complexity of this algorithm is relative high, which cannot be adopted for solving our problems. However, a deep investigation on the special structure of tridiagonal Toeplitz matrix P_1 and P_2 should be exploited for acceleration.

As demonstrated in Fig. 6, the basic idea behind FPS preconditioning is to perform regularization in Poisson block. For a certain metal layer, the width of all metal slices is calculated and the average value of them is adopted as the typical width of this metal layer. Then the original irregular metal layer is transformed into a regular metal layer with the typical width. For x-direction, it calculates the average value of α as g^{α}_{avg}. As for the y-direction, it calculates the average value of β as g^{β}_{avg}. Then we have $g_x = g^{\alpha}_{\text{avg}}$ and $g_y = g^{\beta}_{\text{avg}}$ for regularization.

The proposed preconditioning is to adopt the FPS on the regular grid to precondition the irregular grid. The detailed demonstration of preconditioning algorithm is shown in Algorithm 1. First, the dense matrix W is formulated according to the transformed structured grid and the vector of residual r is reshaped as dense form F. Then the dense form solution D is computed by FPS, which can be reshaped as a preconditioned vector z. The solution residual vector r is taken after each CG iteration and the preconditioned vector z is adopted by next CG iteration.

Algorithm 1 FPS Preconditioning Algorithm

```
Input: The average resistance value $r_{average}$, metal via
       resistance $R_{via}$, the residual vector of voltage
       solution $r$, the grid size $m \times n$, $z_1$, $z_2$, $\Delta_1$, $\Delta_2$,
       the $reshape(X, M, N)$ function is similar to
       Matlab which returns the $M$-by-$N$ matrix whose
       elements are taken column-wise from $X$.
Output: The preconditioned vector $z$.
1 Formulate dense matrix $W$ by $r_{average}$, $R_{via}$, $\Delta_1$, $\Delta_2$;
2 Formulate current loadings matrix
  $F = reshape(r, n, m)$;
3 Obtain the voltage drop $D = z_2 \cdot ((z_2 \cdot F \cdot z_1) \odot W) \cdot z_1$;
4 Return $z = reshape(D^T, m \cdot n, 1)$.
```

It should be emphasized that this analytic preconditioning method can be very efficient for unstructured grid. In another view, this analytic method is a direct solver for structured grid with ideal boundary conditions. And for unstructured grid with unideal boundary conditions, the preconditioning algorithm can provide an approximate voltage distribution, which can be regarded as the low frequency errors have been reduced and then several CG iterations are performed to smooth the global errors all around the grid, which can be regarded as the high frequency errors have been smoothed. Obviously this error smoothing strategy is the essence of the proposed FPS preconditioning technique.

B. Preconditioning Performance

This analytic preconditioner can be regarded as an approximate inverse of conductance matrix G. We will analyze its preconditioning performance theoretically. For a linear system equation $Ax = b$, preconditioning technique is to apply iterative algorithm to $M^{-1}A$, where M is chosen so that $M^{-1}A$ is better conditioned and the systems of form $Mz = y$ are easily solved.

As described in Section IV-A, the conductance matrix $G \in \mathbb{R}^{mn\times mn}$ can be formulated as

$$G = \Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2. \tag{22}$$

Using properties of the Kronecker product and remembering that matrix z_1 and z_2 are symmetry orthogonal matrices, G can be reformulated as

$$\begin{aligned} G &= \Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2 \\ &= (z_1 \cdot \Sigma_1 \cdot z_1) \otimes (z_2 \cdot \Delta_2 \cdot z_2) \\ &\quad + (z_1 \cdot \Delta_1 \cdot z_1) \otimes (z_2 \cdot \Sigma_2 \cdot z_2) \\ &= (z_1 \otimes z_2) \cdot (\Sigma_1 \otimes \Delta_2) \cdot (z_1 \otimes z_2) \\ &\quad + (z_1 \otimes z_2) \cdot (\Delta_1 \otimes \Sigma_2) \cdot (z_1 \otimes z_2) \\ &= (z_1 \otimes z_2) \cdot (\Sigma_1 \otimes \Delta_2 + \Delta_1 \otimes \Sigma_2) \cdot (z_1 \otimes z_2) \end{aligned} \tag{23}$$

where $(\Sigma_1 \otimes \Delta_2 + \Delta_1 \otimes \Sigma_2)$ is diagonal. Observing that the middle expression is a diagonal matrix, if we define

$$\Lambda = (\Sigma_1 \otimes \Delta_2 + \Delta_1 \otimes \Sigma_2) \tag{24}$$

and also remembering that matrix z_1 and z_2 are symmetry orthogonal matrices, the inverse of matrix G can be obtained directly by

$$\begin{aligned} G^{-1} &= [(z_1 \otimes z_2) \cdot (\Sigma_1 \otimes \Delta_2 + \Delta_1 \otimes \Sigma_2) \cdot (z_1 \otimes z_2)]^{-1} \\ &= (z_1 \otimes z_2)^{-1} \cdot (\Sigma_1 \otimes \Delta_2 + \Delta_1 \otimes \Sigma_2)^{-1} \cdot (z_1 \otimes z_2)^{-1} \\ &= (z_1 \otimes z_2) \cdot \Lambda^{-1} \cdot (z_1 \otimes z_2). \end{aligned} \quad (25)$$

That is to say, the analytic solution of structured grid can be obtained by the matrix inverse of closed form. It is easy to find that the closed formulation is equivalent to the FPS on dense form. And for unstructured case, we first transform it as a structured grid and then adopt its analytic inverse to precondition the original unstructured grid. Take a small 12×15 unstructured grid with Wire-Bond package as an example, we compare the sparse pattern of matrix inverse between its accurate inverse and the equivalent analytic inverse. As shown in Fig. 7, their inverse patterns are so similar that the proposed preconditioning strategy should be sufficient.

It is easy to find that the closed formulation (25) is equivalent to the step 3 in Algorithm 1. Because the transformed structured power grid is adopted as an analytic preconditioner, the difference between original unstructured grid and transformed structured grid will affect the preconditioning performance. However, the theoretical proof and convergence analysis will demonstrate that the preconditioning performance will hardly be affected by the increasing of grid size as long as the resistances value range of metal sheet has been decided. Thus, the robustness of proposed preconditioning technique is guaranteed by the analytic preconditioner. For more detailed demonstration, please refer to Section IV-E.

C. FPS-PCG Algorithm

The most classical preconditioner is incomplete Cholesky (IC) factorization. But it is either too expensive or inefficient for larger grids due to the tradeoff between the elements fill-in and the performance of preconditioning. In particular, for unbalanced metal width distribution, which locates at a broad range, the condition number of conductance matrix G is too large. Hence, the robustness of IC preconditioner cannot be guaranteed. By taking advantage of the analytic preconditioner of FPS, a highly robust preconditioning strategy is proposed for efficient power grid analysis.

Aiming to improve the convergence, FPS is adopted as an implicit preconditioner for CG iterative method. That is to say, FPS is adopted to precondition the residual of each CG iteration, which is shown in Algorithm 1. The proposed FPS-PCG algorithm has been described in Algorithm 2 for more details. The FPS is introduced as a preconditioner for conjugate gradient method. The voltage solution is obtained by several CG iterations, while FPS is just adopted as the inner preconditioner. As long as the residual tolerance of CG iterations can be satisfied, the solution accuracy should be guaranteed naturally. The only consideration is focused on the performance of the FPS preconditioner while the convergence of FPS preconditioner is theoretically illustrated in Section IV-E.

Experimental results have shown that the number of iterations required by FPS-PCG method is much less than traditional conjugate gradient method. The computation cost of each FPS-PCG iteration mainly comes from FPS preconditioning step. More specifically, the computation cost of each FPS preconditioning stage lies in matrix multiplication from (14). The detailed preconditioning implementation will be discussed in Section IV-D.

Algorithm 2 FPS-PCG Algorithm

```
Input: The sparse conductance matrix $A \in {}^{n\times n}$, the FPS
       preconditioner $z = FPSPrecond(r)$, the right
       hand side vector $b \in \mathbb{R}^{n\times 1}$ and the residual
       tolerance $tol$.
Output: The voltage solution $x \in \mathbb{R}^{n\times 1}$.
1  $x_0 = FPSPrecond(b)$;
2  $r_0 = b - Ax_0$;
3  $z_0 = FPSPrecond(r_0)$;
4  $p_0 = z_0$;
5  for $k = 0, 1, \ldots,$ until converge to tol do
6      $\alpha_k = \frac{r_k^T z_k}{p_k^T A p_k}$;
7      $x_{k+1} = x_k + \alpha_k p_k$;
8      $r_{k+1} = r_k - \alpha_k A p_k$;
9      $z_{k+1} = FPSPrecond(r_{k+1})$;
10     $\beta_k = \frac{r_{k+1}^T z_{k+1}}{r_k^T z_k}$;
11     $p_{k+1} = z_{k+1} + \beta_k p_k$;
12 end
13 Return the solution $x_{k+1}$.
```

D. Preconditioning Implementation

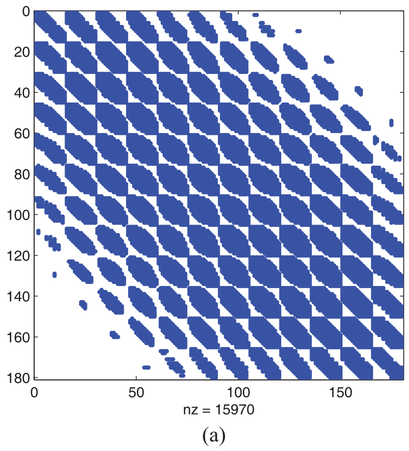
As demonstrated in Section IV-C, the main computation cost is the preconditioning stage of each iteration. Obviously it will be time-consuming to perform matrix multiplication directly from (14) among preconditioning stage. But by FPS [12], the matrix multiplication can be computed by fast Fourier transform (FFT) due to the special structure of matrix z_1 and z_2. As denoted in (12), all entries in matrix z_1 and z_2 are results of different sine functions. The calculation of $R = z_1 \cdot F$, in which F can be regarded as $F = (f_1, f_2, \ldots, f_n)$, can be computed by $R = (z_1 \cdot f_1, z_1 \cdot f_2, \ldots, z_1 \cdot f_n)$. For the kth column of R, $R_k = z_1 \cdot f_k$, considering that z_1's definition in (12), it can be written as following formulation:

$$R_k = \sum_{j=1}^{m} f_{k,j} \cdot z_1(k, j) = \sqrt{\frac{2}{n+1}} \sum_{j=1}^{m} f_{k,j} \cdot \sin\left(\frac{k \cdot j \cdot \pi}{n+1}\right) \quad (26)$$

Noticed that the above formulation is performing m points discrete sine transform (DST) on sequence f_k, denoted as $DST_m(f_k)$, which is related to discrete Fourier transform (DFT). Further, a m points DST can be equivalently obtained by performing a $2m + 2$ points DFT [30] according to basic property of DFT. Thus, we can obtain the following formulation:

$$\mathrm{DST}_m(f_k) = \mathrm{DFT}_{2m+2}(g_k), \quad g_k = [0, f_k, 0, -f_k]. \quad (27)$$

By performing DFT on each column of matrix F, we can obtain the result of $R = z_1 \cdot F$. Similarly we can obtain the result of $Q = F \cdot z_1$ by performing DFT on each row of matrix F. Moreover, DFT can be obtained by fast Fourier



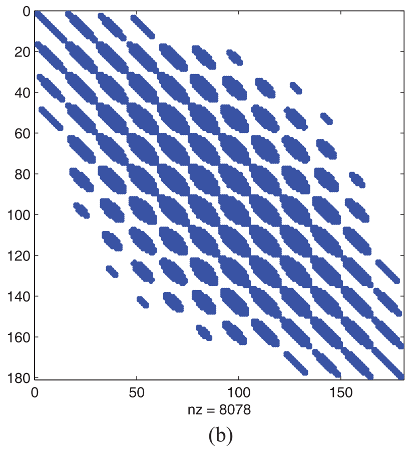


Fig. 7. Comparison of sparse pattern by different inverse methods (a sample 12×15 power grid with Wire-Bond package). (a) Accurate inverse of original unstructured power grid. (b) Equivalent analytic inverse by FPS of transformed power grid.

transform (FFT) with the complexity of $\mathcal{O}(\mathcal{N}\log\mathcal{N})$. Since the matrix multiplication operation is the most time-consuming task in FPS preconditioner, the computation complexity of preconditioner can be bounded by $\mathcal{O}(\mathcal{N}\log\mathcal{N})$. Notice that the complexity of traditional solvers is about $\mathcal{O}(\mathcal{N}^{1.5})$, such as ICCG solver. Even though without any parallel implementation for acceleration, the proposed analytic preconditioner is still much more efficient than the traditional solver.

E. Convergence Analysis

For the symmetric positive definite system $Ax = b$, the convergence property of conjugate gradient method has been proved that [16, p. 192]

$$\left\|x_k - x^*\right\|_A \le 2\left(\frac{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}\right)^k \left\|x_0 - x^*\right\|_A \qquad (28)$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are maximum eigenvalue and minimum eigenvalue of A, respectively, x^* is the exact solution and x_k is the solution after kth iteration. It implies that one way to guarantee that CG method converges rapidly is to reduce the ratio of $\sqrt{\lambda_{\max}}/\sqrt{\lambda_{\min}}$, that is to ensure that A has a condition number approximately equal to 1.

There are many preconditioning techniques for conjugate gradient methods. It is really worth to mention the optimal preconditioners, which is very similar to FPS preconditioner. The interest of optimal preconditioner is drawn from preliminary experience of Kronecker product preconditioners in the conjugate gradient settings [31]. Suppose $A \in \mathbb{R}^{m\times n}$ with $m = m_1 m_2$ and $n = n_1 n_2$, it considers the problem of finding $B \in \mathbb{R}^{m_1\times n_1}$ and $C \in \mathbb{R}^{m_2\times n_2}$ so that

$$\phi_A(B, C) = \|A - B \otimes C\|_F \qquad (29)$$

is minimized. For this solution process to be successful, the preconditioner should capture the essence of matrix A as much as possible subject to the constraint that a linear system $Mz \equiv (B \otimes C)z = r$ is easy to solve. There are some works on finding good preconditioners through an appropriately constrained minimization of $\phi_A(B, C)$ [31], especial for the case when A is Toeplitz matrix [32] or Toeplitz blocks [33]. This minimization needs to perform additional computations, such as singular value decomposition which may lead to more complex problems [31] and cannot be adopted for solving large systems. Luckily, for our power grid analysis which is related to discretized Poisson problems, FPS preconditioner can be introduced without performing additional computations because of analytic preconditioning process.

As demonstrated in Section IV-A, the conductance matrix of structured power grid (standard Poisson block) can be formulated as (17), that is

$$G = \Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2 \qquad (30)$$

and similarly, the conductance matrix of unstructured power grid can also be formulated as (19), that is

$$G' = \alpha \otimes P_2 + P_1 \otimes \beta. \qquad (31)$$

Similar to the optimal preconditioner problem like (29), we choose the (17) as a preconditioner for solving $G'x = b$ because both G and G' has the same Kronecker product structures.

As we have known that the ratio of $\sqrt{\lambda_{\max}}/\sqrt{\lambda_{\min}}$ directly affects the convergence performance of conjugate gradient methods, we will analyze the convergence when using FPS as a preconditioner. We have

Theorem 1: Let $G'x = b$, where G' is defined in (19), when using (17) as a preconditioner for solving $G'x = b$ with iterative methods. Once $\alpha \in \mathbb{R}^{m\times m}$ and $\beta \in \mathbb{R}^{n\times n}$ are bounded by two certain range, that is to say, $g^{\alpha}_{\min} \le \alpha_i \le g^{\alpha}_{\max}$ $(i = 1, 2, \ldots, m)$ and $g^{\beta}_{\min} \le \beta_j \le g^{\beta}_{\max}$ $(j = 1, 2, \ldots, n)$, then the number of iterations remains roughly a constant.

Proof: When using (17) as a preconditioner for solving $G'x = b$, we obtain the preconditioned system $G^{-1}G'$ which can be described as

$$G^{-1}G' = (\Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2)^{-1}(\alpha \otimes P_2 + P_1 \otimes \beta) \qquad (32)$$

where $(\Sigma_1 \otimes P_2 + P_1 \otimes \Sigma_2)^{-1}$ can be directly obtained as shown in (25). And as described in (12), the tridiagonal Toeplitz matrices P_1 and P_2 have known eigenvalues

$$\Delta_1(i, i) = 2\left(1 - \cos\frac{i\cdot\pi}{m+1}\right), \quad i = 1, 2, \ldots, m$$

TABLE I
EVALUATION FOR CONVERGENCE RATE

Example	$[g^{\alpha}_{\min}, g^{\alpha}_{\max}]$	$[g^{\beta}_{\min}, g^{\beta}_{\max}]$	$\sqrt{\frac{g^{\alpha}_{\min}+g^{\beta}_{\min}}{g^{\alpha}_{\max}+g^{\beta}_{\max}}}$	$\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$
No. 1	[3, 12]	[4, 15]	1.96396	1.99992
No. 2	[0.1, 1.05]	[0.2, 1.16]	2.71416	3.23163
No. 3	[3, 102]	[4, 123]	5.66947	5.78548

$$\Delta_2(j,j) = 2\left(1 - \cos\frac{j\cdot\pi}{n+1}\right), \quad j = 1,2,\ldots,n. \tag{33}$$

Using this result, and from the point of view of optimal preconditioner problem, we can consider that

$$\begin{aligned} &\|G' - G\|_F^2 \\ &= \|(\alpha\otimes P_2 + P_1\otimes\beta) - (\Sigma_1\otimes P_2 + P_1\otimes\Sigma_2)\|_F^2 \\ &= \|(\alpha\otimes\Delta_2 + \Delta_1\otimes\beta) - (\Sigma_1\otimes\Delta_2 + \Delta_1\otimes\Sigma_2)\|_F^2 \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}\left[\begin{matrix}(\alpha_i\Delta_2(j,j) + \Delta_1(i,i)\beta_j) \\ -(g_x\Delta_2(j,j) + \Delta_1(i,i)g_y)\end{matrix}\right]. \end{aligned} \tag{34}$$

The eigenvalue distribution of $G^{-1}G'$, which is crucial to the success of using $G = \Sigma_1\otimes P_2 + P_1\otimes\Sigma_2$ as a good preconditioner, can also be examined by closed form

$$\lambda_{ij}\left(G^{-1}G'\right) = \frac{\alpha_i\Delta_2(j,j) + \Delta_1(i,i)\beta_j}{g_x\Delta_2(j,j) + g_y\Delta_1(i,i)}. \tag{35}$$

Even though the function characteristics of $\Delta_1(i,i)$ and $\Delta_2(j,j)$ are known, we cannot understand the exact distribution range of α and β. Thus, it is not easy to decide the exact lower bound and upper bound of (35). Because the values of $\Delta_1(i,i)$ and $\Delta_2(j,j)$ are located between $(0,4)$ and appear both in numerator part and denominator part, we can consider that $[g^{\alpha}_{\min}, g^{\alpha}_{\max}]$ and $[g^{\beta}_{\min}, g^{\beta}_{\max}]$ will dominate the lower bound and upper bound of (35). If we define two scale factors μ_1 and μ_2, then its approximative minimum eigenvalue and maximum eigenvalue can be given by

$$\begin{aligned} \left[\lambda_{ij}\left(G^{-1}G'\right)\right]_{\min} &\approx \mu_1\frac{g^{\alpha}_{\min} + g^{\beta}_{\min}}{g_x + g_y} \\ \left[\lambda_{ij}\left(G^{-1}G'\right)\right]_{\max} &\approx \mu_2\frac{g^{\alpha}_{\max} + g^{\beta}_{\max}}{g_x + g_y}. \end{aligned} \tag{36}$$

Remembering $g_x = g^{\alpha}_{\text{avg}}$ and $g_y = g^{\beta}_{\text{avg}}$, the eigenvalue distribution of $G^{-1}G'$ can be bounded as

$$\mu_1\frac{g^{\alpha}_{\min} + g^{\beta}_{\min}}{g^{\alpha}_{\text{avg}} + g^{\beta}_{\text{avg}}} \le \lambda_{ij}\left(G^{-1}G'\right) \le \mu_2\frac{g^{\alpha}_{\max} + g^{\beta}_{\max}}{g^{\alpha}_{\text{avg}} + g^{\beta}_{\text{avg}}}. \tag{37}$$

Using this result, the ratio of $\sqrt{\lambda_{\max}}/\sqrt{\lambda_{\min}}$ which is important to convergence can be obtained by

$$\frac{\sqrt{\left[\lambda_{ij}\left(G^{-1}G'\right)\right]_{\max}}}{\sqrt{\left[\lambda_{ij}\left(G^{-1}G'\right)\right]_{\min}}} \approx \sqrt{\frac{\mu_2}{\mu_1}\cdot\frac{g^{\alpha}_{\max} + g^{\beta}_{\max}}{g^{\alpha}_{\min} + g^{\beta}_{\min}}}. \tag{38}$$

Hence, the convergence of the proposed preconditioner is only affected by the conductance distribution range $[g^{\alpha}_{\min}, g^{\alpha}_{\max}]$ and $[g^{\beta}_{\min}, g^{\beta}_{\max}]$. That is to say, the number of iterations remains roughly a constant as long as the conductance distribution range has been decided. ■

Theorem 1 has shown that when using FPS as preconditioner, the convergence is very robust which will not vary with the increasing of problem size. Aiming to verify the promising property, we take three conductance distribution ranges as example. For each example, the conductance distribution range is decided and several power grids are generated with different grid size. As shown in Table I, $[g^{\alpha}_{\min}, g^{\alpha}_{\max}]$ is for x-direction metal lines, $[g^{\beta}_{\min}, g^{\beta}_{\max}]$ is for y-direction metal lines. Without loss of generality, the computed ratio by (37) is roughly

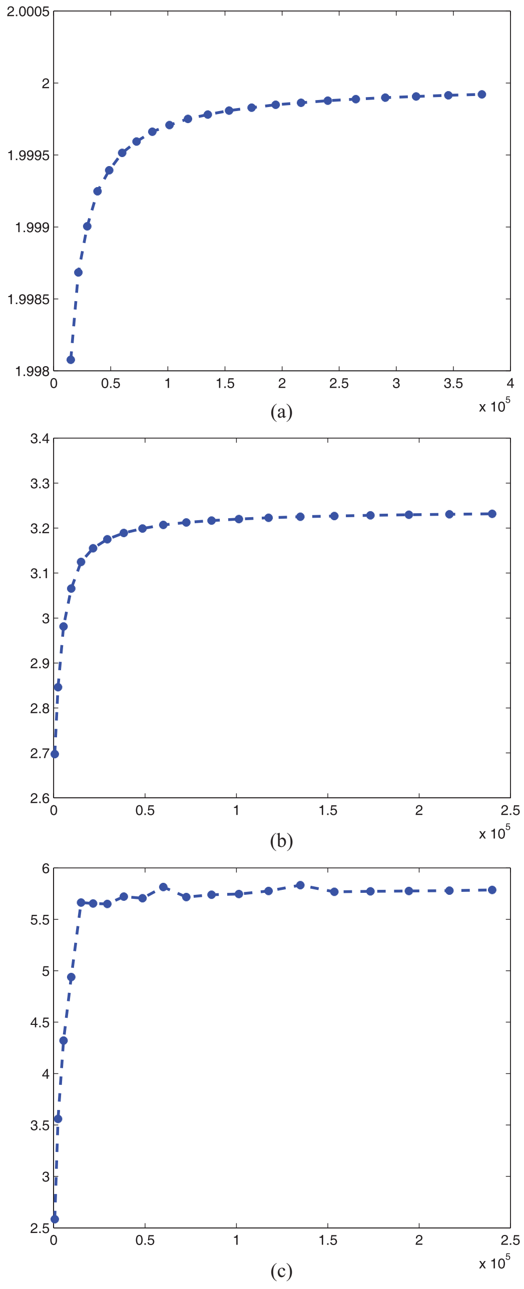


Fig. 8. Trend of the computed ratio with the increasing of grid size. The horizontal axis is to represent the increasing of grid size. The vertical axis is to represent the computed ratio $\sqrt{(g^{\alpha}_{\min} + g^{\beta}_{\min})/(g^{\alpha}_{\max} + g^{\beta}_{\max})}$. (a) Example #1 ($3 \le g^{\alpha} \le 12$ and $4 \le g^{\beta} \le 15$). (b) Example #2 ($0.1 \le g^{\alpha} \le 1.05$ and $0.2 \le g^{\beta} \le 1.16$). (c) Example #3 ($3 \le g^{\alpha} \le 102$ and $4 \le g^{\beta} \le 123$).

denoted as a predicted ratio $\sqrt{(g_{\min}^{\alpha}+g_{\min}^{\beta})/(g_{\max}^{\alpha}+g_{\max}^{\beta})}$. Meanwhile, the actual ratio $\sqrt{\lambda_{\max}/\lambda_{\min}}$ is also computed for $G^{-1}G'$ and listed on the last column in Table I. It is obvious that the predicted ratio is very close to the actual ratio, which is related to the condition number. And as shown in Fig. 8, the trend of each predicted ratio is nearly a constant with the increasing of grid size. Since the condition number is almost a constant for different grid size, the convergence should be robust for very large scale power grid analysis.

V. Experimental Results

Various experiments are carried out to validate the promising performance of the proposed FPS-PCG algorithm. The regular topology of power grid is adopted in this paper because it is really common in use for early design stage. All power grids are constructed by unstructured metal stripes whose resistance values are randomly generated in a certain range. This range is among 0.01 and 1 Ω, which is close to real industrial designs. Also, the pads distribution is unideal for boundary conditions which are similar to real industrial designs. There are 10% grid nodes on boundaries connected to pads for Wire-Bond package. The resistance value of pad contact is set to 5 Ω and their connected standard voltage sources are 1.8 V. The current loadings of power grid are also generated reasonably and randomly.

All algorithms have been implemented using C/C++ language and the performance of different solving techniques is compared under the same environments. The simulation platform is a 64-Bit Linux Server with 2 Quad-Core Intel Xeon E5506 CPU@2.13 GHz and 24-GB RAM. All running times are measured in seconds. The stopping criteria for the FPS-PCG method are defined as relative residual

$$\text{tol}=\frac{\|r_k\|_2}{\|b\|_2} \tag{39}$$

where $\|r_k\|_2$ is the two-norm of the residual vector after k-th iterations, $\|b\|_2$ is the two-norm of the right-hand side vector, and tol is set to 10^{-6}, which is reliable for convergence. The FFTW3 library [34] is adopted in FPS preconditioning steps. The Hybrid solver [35] which is a well developed random walk linear solver [36] is also evaluated for comparison.

A. Preconditioning Observations

As demonstrated above, structured grid with ideal boundary conditions can satisfy the definitions of Poisson block strictly. Hence, it can be solved by FPS analytically. In this paper, we propose the idea of using FPS as preconditioning technique for unstructured grid with unideal boundary conditions. First, we demonstrate the ability of the proposed preconditioner to smooth global errors. Take a 150 × 150 grid with Wire-Bond package as an example; the residual distribution of first guess has been observed to be somewhat similar to the final voltage distribution due to the analytic solution of FPS. And just with one iteration with FPS as preconditioner, the residual distribution is reduced to about 10^{-3}, which is relatively small. It can be observed that the residual mainly exists on boundary nodes so that more iterations should be performed. After ten iterations with FPS as preconditioner, the residual distribution is about 10^{-4}. And later the final solutions are obtained after 32 iterations and the solution errors are below 10^{-6}. Thus, the proposed analytic preconditioner of FPS is observed to be very sufficient for power grid analysis.

B. Comparing FPS-PCG With Hybrid Solver and ICCG

The comprehensive results of FPS-PCG solver, Hybrid solver, and ICCG solver for unstructured power grids with unideal boundary conditions are shown in Table II. The Hybrid solver is a well developed Random Walk-based linear solver [35], but it cannot report the number of iterations and residual value. The implemented ICCG solver adopts IC(0) as preconditioner, which is based on incomplete Cholesky factorization with zero fill-in by the threshold of zero for drop tolerance. The runtime of ICCG includes the factorization time and solving time. The preconditioning step of FPS-PCG solver is implemented by FFT acceleration, which is shown in Section IV-D. Several power grid benchmarks (from 15 K to 71.4 M) are evaluated by these three solvers and results are shown in Table II. Hybrid solver cannot be used for solving the largest two benchmarks because of memory limited. Experimental results have shown that the solution errors of FPS-PCG methods are located near the pad nodes. But the ICCG solver is different. This phenomenon can be explained by the essence of these two preconditioners. The incomplete Cholesky factorization preconditioner is limited by its threshold of elements fill-in so that the performance of preconditioning is badly weakened all around the power grid. But for analytic FPS preconditioner, one or two preconditioning can give an approximate voltage distribution for global nodes and then several iterations are required to smooth the global errors all around the power grid especially for pad nodes.

As can be observed in Table II, the number of iterations of ICCG solver increases with the increasing of grid size but it is surprisingly almost a constant for FPS-PCG solver, which is not varying significantly with the increasing of grid size. Compared with ICCG and Hybrid solver, a significant reduction in the runtime is observed on all power grid designs when using FPS preconditioning technique. As shown in Fig. 9(a), FPS-PCG solver can achieve about 10× ∼ 20× speedups compared with ICCG solver and about 2× speedups compared with Hybrid solver. For clarity, the runtime T_K for solving each thousand nodes is shown in Table II. The runtime of FPS-PCG for solving each thousand nodes is about from 0.02 to 0.04 s with the increasing of grid size. But for Hybrid solver, it is about from 0.03 to 0.1 s which is more than the FPS-PCG solver. Also for ICCG solver, this runtime is about from 0.02 to 0.7 s which is much more than FPS-PCG solver.

With the continuously increasing of power grid size, the memory usage of simulation seems to be more challenge than runtime by some point of view. The comparison of peak memory usage is listed in Table II. As shown in Fig. 9(b), the peak memory of ICCG solver is lowest among the three solvers because it adopts the very simple preconditioner IC(0). And the peak memory of FPS-PCG solver is about half

TABLE II
COMPARISON BETWEEN FPS-PCG SOLVER, HYBRID SOLVER, AND ICCG SOLVER. FIRST COLUMN IS THE GRID SIZE. $Iter$ IS THE NUMBER OF ITERATIONS FOR EACH SOLVER. T IS CPU RUNTIME OF EACH SOLVER, WHICH IS MEASURED IN SECONDS. Mem IS THE PEAK MEMORY USAGE OF EACH SOLVER, WHICH IS MEASURED BY MEMTIME. T_K IS THE RUNTIME FOR SOLVING EACH THOUSAND NODES. Mem_K IS THE PEAK MEMORY USAGE FOR SOLVING EACH THOUSAND NODES IN WHICH IT IS MEASURED BY KB. $Resid$ IS THE RESIDUAL OF ITERATION METHODS

Size	FPS-PCG						Hybrid				ICCG					
	$Iter$	T	T_K	Mem	Mem_K	$Resid$	T	T_K	Mem	Mem_K	$Iter$	T	T_K	Mem	Mem_K	$Resid$
15K	48	0.48	0.032	7.67M	0.52	8E-7	0.43	0.029	11M	0.79	166	0.38	0.025	4.92M	0.34	9E-7
200K	59	4.72	0.024	77M	0.39	9E-7	7.43	0.037	132M	0.68	468	15.27	0.076	42M	0.22	9E-7
560K	64	14.66	0.026	177M	0.32	9E-7	23.66	0.042	386M	0.67	710	66.17	0.118	115M	0.21	9E-7
1.2M	69	30.64	0.026	336M	0.29	9E-7	54.07	0.045	787M	0.67	998	189.64	0.158	244M	0.21	9E-7
1.82M	70	58.09	0.032	536M	0.30	9E-7	86.62	0.048	1.16G	0.67	1209	368.05	0.202	369M	0.21	9E-7
2.88M	71	87.54	0.030	712M	0.25	9E-7	173.61	0.060	1.82G	0.66	1449	694.96	0.241	584M	0.21	9E-7
3.99M	72	138.26	0.035	985M	0.25	9E-7	208.09	0.052	2.55G	0.67	1564	952.58	0.239	808M	0.21	9E-7
5.28M	73	152.68	0.029	1.27G	0.25	8E-7	296.33	0.056	3.36G	0.67	1760	1493.61	0.283	1.04G	0.21	9E-7
7M	74	260.79	0.037	1.68G	0.25	9E-7	504.21	0.072	4.45G	0.67	1965	2294.10	0.328	1.38G	0.21	9E-7
9M	75	288.94	0.032	2.16G	0.25	9E-7	543.99	0.060	5.69G	0.66	2130	3213.42	0.357	1.78G	0.21	9E-7
11.2M	76	358.61	0.032	2.68G	0.25	9E-7	738.99	0.066	7.08G	0.66	2381	4465.22	0.399	2.21G	0.21	9E-7
14.4M	77	507.90	0.035	3.45G	0.25	9E-7	1154.35	0.080	9.11G	0.66	2622	5895.89	0.409	2.84G	0.21	9E-7
18.9M	78	682.86	0.036	4.53G	0.25	8E-7	1574.15	0.083	11.96G	0.66	2984	8531.40	0.451	3.73G	0.21	9E-7
24M	79	861.65	0.035	5.74G	0.25	8E-7	2259.27	0.094	15.18G	0.66	3331	11733.30	0.489	4.74G	0.21	9E-7
33M	79	1320.47	0.040	7.89G	0.25	9E-7	3529.94	0.107	20.91G	0.66	3507	17205.20	0.521	6.52G	0.21	9E-7
44.8M	81	1903.22	0.042	10.71G	0.25	9E-7	4688.98	0.105	23.44G	0.55	3964	26353.70	0.588	8.84G	0.21	9E-7
60M	82	2173.75	0.036	14.34G	0.25	8E-7	–	–	–	–	4537	39552.00	0.659	11.85G	0.21	9E-7
71.4M	82	2960.79	0.041	17.06G	0.25	9E-7	–	–	–	–	4933	50710.90	0.710	14.10G	0.21	9E-7

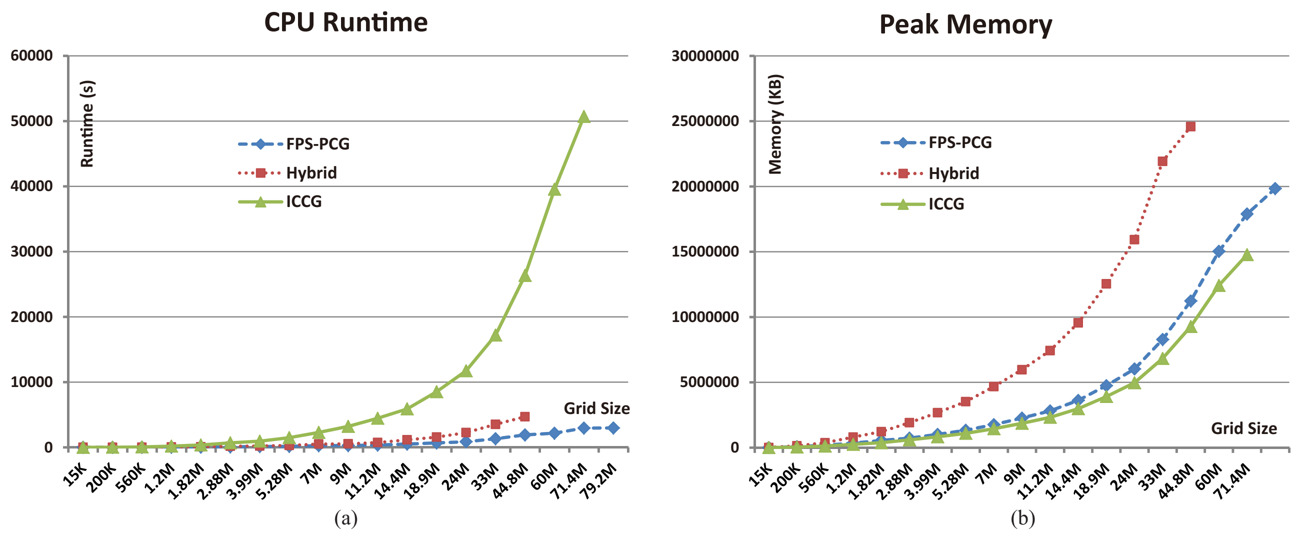


Fig. 9. Run efficiency comparison. (a) Runtime comparison among three solvers. (b) Peak memory comparison between three solvers.

of Hybrid solver. Since there is no need to construct an explicit preconditioner for FPS-PCG solver while the memory consumption mainly lies in the FFT procedures. For clarity, the peak memory usage Mem_K for solving each thousand nodes is also listed in Table II. The peak memory usage of FPS-PCG for solving each thousand nodes is about 0.25 kB with the increasing of grid size. And for Hybrid solver, it is about 0.66 kB, which is more than FPS-PCG solver. Also for ICCG solver, the peak memory usage is about from 0.21 kB, which is a little less than FPS-PCG solver.

For the comparison of computational complexity, ICCG solver is about $\mathcal{O}(\mathcal{N}^{1.5})$ where N is the number of nodes. The computational complexity of our proposed method has been discussed in Section IV. Since we have proved that the number of iterations for FPS-PCG solver is almost a constant, we just need to analyze the complexity of each iteration. For each iteration process, a FPS preconditioning step and a sparse matrix vector multiplication are performed with the complexity of $\mathcal{O}(\mathcal{N}\log\mathcal{N})$ and $\mathcal{O}(\mathcal{N})$, respectively. Accordingly, the total computational complexity of the proposed approach is

TABLE III
COMPARISON BETWEEN FPS AND FPS-PCG. T IS THE CPU RUNTIME OF FPS METHOD USING BOUNDARY ITERATION PROCESS [12]. *Iter* IS THE ITERATIONS OF EACH SOLVER. *Time* IS THE CPU RUNTIME OF EACH SOLVER, WHICH IS MEASURED IN SECONDS

Size	T [12]†	FPS-PCG†			FPS-PCG*		
		Iter	*Time*	*Resid*	*Iter*	*Time*	*Resid*
1*M*	15.64	33	9.37	7E-7	67	20.11	9E-7
1.2*M*	20.93	34	14.84	8E-7	69	29.54	9E-7
1.4*M*	27.16	35	18.94	7E-7	70	36.88	8E-7
1.6*M*	36.33	35	21.26	8E-7	70	38.88	9E-7
1.9*M*	44.87	36	26.68	7E-7	70	50.21	9E-7
2.3*M*	54.90	36	28.38	8E-7	71	59.16	7E-7
2.5*M*	66.27	33	34.72	8E-7	71	73.46	9E-7
2.9*M*	79.16	34	42.56	9E-7	69	84.94	9E-7
3.2*M*	94.30	35	45.11	9E-7	71	92.37	8E-7
3.6*M*	110.51	34	49.15	8E-7	71	100.46	8E-7
4*M*	125.63	37	56.96	9E-7	72	112.05	8E-7

† On structured power grids.
* On unstructured power grids.

between $\mathcal{O}(\mathcal{N})$ and $\mathcal{O}(\mathcal{N}\log\mathcal{N})$, which is a considerable improvement. Meanwhile, the proposed analytic preconditioner has been validated as a good sparse approximate inverse technique, which is extremely effective and robust for large scale power grid analysis.

C. Comparing FPS-PCG With FPS [12]

By taking advantage of the Norton's Theorem which is demonstrated in Section III-D, there is no need to carry out the boundary iteration process [12]. Table III illustrates the runtime comparison between FPS with boundary iteration process [12] and our proposed FPS-PCG solver. First, these two approaches are evaluated on solving structured power grid. As shown in Table III, FPS-PCG solver eventually achieves approximately 2× speedup than FPS with boundary iteration on structured grid. The following insightful experiments are also conducted. FPS-PCG solver is evaluated for solving structured power grids and unstructured power grids, respectively. According to the results listed in Table III, the additional iterations when solving unstructured grids are resulted by irregular metal conductance distribution.

VI. CONCLUSION

In this paper, an efficient and robust FPS-PCG method was proposed for large scale power grid analysis. The proposed analytic formulation was improved the numerical characteristics for multilayer power grids with large variations in conductance among different metal layers and vias. The idea of transforming unstructured grid to structured grid was proposed to properly tackle the unstructured grids with unideal boundary conditions. Thus, a friendly robust analytic preconditioner was adopted to improve the convergence of conjugate gradient methods while the FPS provided an approximate voltage distribution. Then, the solution residual was smoothed to a satisfactory level quickly. The greatest benefit of this approach was that the number of iterations is insensitive to the increasing of grid size. Moreover, this approach can also be accelerated on GPU platforms for parallelization due to the analytic formulation, which can be realized by fast Fourier transform method.

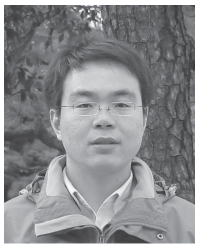
ACKNOWLEDGMENT

The authors would like to thank Dr. H. Qian from Thomas J. Watson Research Center, of IBM, for providing the Hybrid solver library to us.

REFERENCES

[1] T.-H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. IEEE/ACM Design Autom. Conf.*, 2001, pp. 559–562.
[2] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 8, pp. 1204–1224, Jul. 2005.
[3] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2001, pp. 480–487.
[4] K. Sun, Q. Zhou, K. Mohanram, and D. C. Sorensen, "Parallel domain decomposition for simulation of large-scale power grids," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 54–59.
[5] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw, "Hierarchical analysis of power distribution networks," in *Proc. IEEE/ACM Design Autom. Conf.*, Jan. 2000, pp. 150–155.
[6] S. Cauley, V. Balakrishnan, and C.-K. Koh, "A parallel direct solver for the simulation of large-scale power/ground networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 4, pp. 636–641, Apr. 2010.
[7] J. M. S. Silva, J. R. Phillips, and L. M. Silveira, "Efficient simulation of power grids," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 10, pp. 1523–1532, Oct. 2010.
[8] E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2004, pp. 485–488.
[9] J. Shi, Y. Cai, S. X.-D. Tan, and X. Hong, "High accurate pattern based precondition method for extremely large power/ground grid analysis," in *Proc. IEEE/ACM Int. Symp. Phys. Design*, Apr. 2006, pp. 108–113.
[10] S. Köse and E. G. Friedman, "Fast algorithms for IR voltage drop analysis exploiting locality," in *Proc. 48th Design Autom. Conf.*, Jun. 2011, pp. 996–1001.
[11] S. Köse and E. G. Friedman, "Efficient algorithms for fast IR drop analysis exploiting locality," *Integr. VLSI J.*, vol. 45, no. 2, pp. 149–161, Mar. 2012.
[12] J. Shi, Y. Cai, W. Hou, L. Ma, S. X.-D. Tan, X.-D. Sheldon, P.-H. Ho, and X. Wang, "GPU friendly fast Poisson solver for structured power grid network analysis," in *Proc. 46th IEEE/ACM Design Autom. Conf.*, Jul. 2009, pp. 178–183.
[13] J. Yang, Y. Cai, Q. Zhou, and J. Shi, "Fast Poisson solver preconditioned method for robust power grid analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2011, pp. 531–536.
[14] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Dec. 2007, pp. 647–654.
[15] Z. Feng and Z. Zeng, "Parallel multigrid preconditioning on graphics processing units GPUs for robust power grid analysis," in *Proc. 47th IEEE/ACM Design Autom. Conf.*, Jun. 2010, pp. 661–666.
[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 1995.
[17] Z. Zeng and P. Li, "Locality-driven parallel power grid optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1190–1200, Aug. 2009.
[18] Z. Zeng, Z. Feng, P. Li, and V. Sarin, "Locality-driven parallel static analysis for power delivery networks," *ACM Trans. Design Autom. Electron. Syst.*, vol. 16, no. 31, pp. 28:1–28:17, Jun. 2011.
[19] R. Achar, M. S. Nakhla, H. S. Dhindsa, A. R. Sridhar, D. Paul, and N. M. Nakhla, "Parallel and scalable transient simulator for power grids via waveform relaxation (PTS-PWR)," *IEEE Trans. Very Large Scale Intergr. (VLSI) Syst.*, vol. 19, no. 2, pp. 319–332, Feb. 2011.
[20] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia, PA, USA: SIAM, Sep. 1995, pp. 101–125.

[21] T. F. Chan and D. C. Resasco, "A domain-decomposed fast Poisson solver on a rectangle," *SIAM J. Sci. Stat. Comput.*, vol. 8, no. 1, pp. 14–26, Jan. 1987.
[22] C. C. Christara and K. S. Ng, "Fast fourier transform solvers and preconditioners for quadratic spline collocation," *BIT Numer. Math.*, vol. 42, no. 4, pp. 702–739, Dec. 2002.
[23] T. A. Manteuffel and S. V. Parter, "Preconditioning and boundary conditions," *SIAM J. Numer. Anal.*, vol. 27, no. 3, pp. 656–694, Jun. 1990.
[24] P. N. Swarztrauber, "The methods of cyclic reduction, fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle," *SIAM Rev.*, vol. 19, no. 3, pp. 490–501, Jul. 1977.
[25] H. Qian and S. S. Sapatnekar, "Fast poisson solvers for thermal analysis," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2010, pp. 698–702.
[26] A. J. Laub, *Matrix Analysis for Scientists and Engineers, Kronecker Products*. Philadelphia, PA, USA: SIAM, 2004, ch. 13.
[27] R. A. Horn and R. A. Horn, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1991.
[28] C. V. Loan, *Computational Frameworks for the Fast Fourier Transform*. Philadelphia, PA, USA: SIAM, 1992.
[29] G. H. Golub, S. Nash, and C. V. Loan, "A Hessenberg-Schur method for the problem $\mathrm{AX} + \mathrm{XB} = \mathrm{C}$," *IEEE Trans. Autom. Control*, vol. 24, no. 6, pp. 909–913, Dec. 1979.
[30] T. Lyche. (2006, Sep.). *Fast Poisson Solvers and FFT* [Online]. Available: http://heim.ifi.uio.no/~tom/fastpoissonslides.pdf
[31] C. V. Loan and N. Pitsianis, "Approximation with kronecker products," in *Linear Algebra for Large Scale and Real Time Applications*. Boston, MA, USA: Kluwer, 1993, pp. 293–314.
[32] T. F. Chan, "An optimal circulant preconditioner for toeplitz systems," *SIAM J. Sci. Stat. Comput.*, vol. 9, no. 4, pp. 766–771, 1988.
[33] R. H. Chan and X.-Q. Jin, "A family of block preconditioners for block systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 5, pp. 1218–1235, 1992.
[34] M. Frigo and S. G. Johnson. (2012, Feb.). *FFTW 3.3.1* [Online]. Available: http://fftw.org/
[35] H. Qian and S. S. Sapatnekar, "Stochastic preconditioning for diagonally dominant matrices," *SIAM J. Sci. Comput.*, vol. 30, no. 3, pp. 1178–1204, 2008.
[36] H. Qian, S. S. Sapatnekar. (2012, Aug.). *Random Walk-Based Hybrid Linear Equation Solver* [Online]. Available: http://www.ece.umn.edu/users/sachin/software/hybridsolver/index.html



Jianlei Yang (S'12) received the B.S. degree in microelectronics from Xidian University, Xi'an, China, in 2009. He is currently pursuing the Ph.D. degree in computer science and technology with Tsinghua University, Beijing, China, and conducting research with the EDA Laboratory.

His current research interests include numerical algorithms for VLSI power grid analysis and verification.

Mr. Yang's team was the recipient of the first place of the TAU Power Grid Simulation Contest in 2011 and the second place of the TAU Power Grid Transient Simulation Contest in 2012.

Yici Cai (M'04–SM'10) received the B.S. degree in electronic engineering and the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 1983 and 1986, respectively, and the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2007.

She has been a Professor with the Department of Computer Science and Technology, Tsinghua University. Her current research interests include design automation for VLSI integrated circuits algorithms and theory, power/ground distribution network analysis and optimization, high performance clock synthesis and low power physical design.

Qiang Zhou (M'04–SM'10) received the B.S degree in computer science and technology from the University of Science and Technology of China, Hefei, China, the M.S degree in computer science and technology from Tsinghua University, Beijing, China, and the Ph.D. degree in control theory and control engineering from the Chinese University of Mining and Technology, Beijing, in 1983, 1986, and 2002, respectively.

He has been a Professor with the Department of Computer Science and Technology, Tsinghua University. Beijing. His current research interests include VLSI layout theory and algorithms.

Jin Shi received the B.S. degree in computer science and technology from the University of Electronic Science and Technology of China, Chengdu, China, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2003 and 2008, respectively.

His current research interests include power/ground network design, analysis and optimization and numeric methods in circuit simulation.