PowerRush: An Efficient Simulator for Static Power Grid Analysis

Jianlei Yang, *Student Member, IEEE*, Zuowei Li, Yici Cai, *Senior Member, IEEE*, and Qiang Zhou, *Senior Member, IEEE*

***Abstract*— Efficient power grid analysis is critical for modern very large scale integration design but is computationally challenging in runtime and memory consumption because of the increasing size of power grids. PowerRush is proposed as an efficient IR-drop simulator, which includes an efficient SPICE parser, a robust circuit builder, and a linear solver Algebraic MultiGrid Preconditioned Conjugate Gradient. The proposed AMG-PCG solver is a pure algebraic method, which can provide stable convergence without geometric information. Aggregation-based AMG with K-cycle acceleration is adopted as a preconditioner to improve the scalability of iterative method. In multigrid scheme, double pairwise aggregation technique is applied to matrix graph in coarsening to ensure low setup cost and memory requirement. Furthermore, K-cycle multigrid scheme is adopted to provide Krylov subspace acceleration at each level to guarantee enhanced robustness and scalability. The experimental results for large-scale power grids have shown that PowerRush has remarkable scalability both in runtime and memory consumption. DC analysis of power grid with 60-million nodes can be solved by PowerRush for 0.01 mV accuracy within 150 s and 21.99 GB total memory used. Moreover, the proposed AMG-PCG solver can perform much better than widely used direct solver Cholmod and well-developed Hybrid solver both on runtime and memory consumption.**

***Index Terms*— Aggregation, algebraic multigrid (AMG), K-cycle, power grid, PowerRush.**

I. Introduction

MODERN VLSI circuits are rapidly becoming more and more power intensive due to the scaling impacts. In [1], it reports that IBM POWER7 microprocessor can consume 230 W within 567 mm^2 area, which means that it has a current density of tens of $Amps/cm^2$. The supply voltage observed by a chip cell is known to vary with its dynamic current drawn due to the impedance of the power delivery network. Fig. 1 shows logic density and supply voltage value projection from the 2011 International Technology Roadmap for Semiconductors (ITRS) report. The trend shows that the voltage profile is going to get worse and power consumption is increasingly prominent in advanced technology nodes, which means that the effect of voltage fluctuations on power grid becomes more and more significant. With the decreased VDD value, how to retain the switching speeds and satisfy the noise margin is becoming more and more challengeable for high performance chip designs. A critical issue in the analysis of power grids is the large size of the network. For recent power grid analysis of IBM processors [1]–[5], the problem size of a ground network on a single core can reach hundreds of millions nodes. Such a large-scale problem can only be solved by highly tuned algorithms with supercomputing resources. Therefore, the design and analysis of extremely large-scale power grids are a very computationally challenging task for VLSI design.

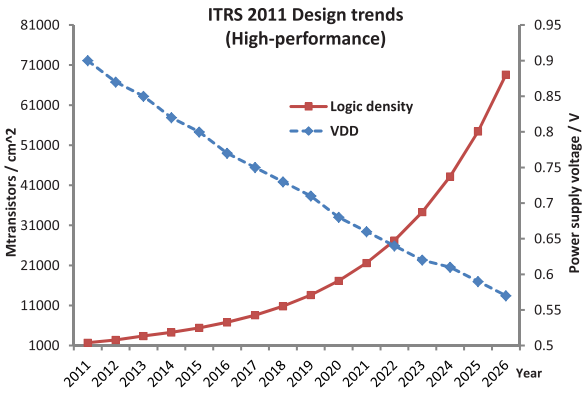


Fig. 1. High-performance design trends of ITRS 2011.

Many contributions have been developed for efficient power grid analysis, which includes direct solvers, iterative solvers, and many other specialized solvers. Direct solvers such as KLU [6] and Cholmod [7] are usually adopted for transient simulation with a fixed time step because the factorized triangular matrix can be reused in the simulation of each time step. However, direct approaches cannot scale well with the problem size for larger scale power grids. Furthermore, the reusability of matrix factorization will be impracticable for simulation with variable time step, which is more appreciated for practical use. Therefore, iterative approaches are relatively competitive than direct approaches in many cases.

Iterative solvers are usually developed for static simulation, such as Krylov subspace method [8] with all sorts of preconditioners, i.e., random walk [9], support graphs [10], and so forth. Among them the random walk-based Hybrid solver has been well developed as a stochastic preconditioner [11] and proved to be very efficient for power grid analysis [9].

Manuscript received January 4, 2013; revised July 5, 2013; accepted September 12, 2013. Date of publication October 2, 2013; date of current version September 23, 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; lizuoweirain@gmail.com; caiyc@mail.tsinghua.edu.cn; zhouqiang@mail.tsinghua.edu.cn).

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.2013.2282418

Especially, there are many specialized methods proposed for efficient power grid analysis, which includes hierarchical and macromodeling methods [12], domain decomposition [13], and some matrix-oriented algebraic methods such as SPAI [14] and $\mathcal{H}$-matrix [15] as well as HSS methods [16], [17].

Due to the similarity between the power grids and discretization of Laplacian equations, multigrid methods are introduced for fast power grid simulation. Geometric-based multigrid like technique [18] was first applied to power grid analysis but its grid coarsening strategy is based on the topology of power grids, which may limit its practical use. To further handle irregular power grids, algebraic multigrid (AMG)-based techniques [19], [20] were also developed. In parallel view, a GPU-based HMD algorithm [21] is proposed to improve the solving efficiency and MGPCG solver [22], [23] is further introduced to improve the robustness of HMD solver. However, the most noticeable problem between 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 and therefore slow convergence.

For AMG approaches, there exists a tradeoff between the setup cost and efficiency of error components reduction. The grid reduction methods in some AMG-like multigrid approaches [19] are also somewhat geometrically based, which may degrade the efficiency due to the irregularity. A pure AMG method [24] was proposed to improve the efficiency whose grid reduction was based on matrix graph. However, its denser prolongation matrices increase setup costs and memory requirements. To overcome the bottleneck of limited convergence by controlling the number of coarsening levels, smoothed aggregation method [25] is developed recently, which is robust and efficient over a wide variety of problems. Aggregation-based AMG is also introduced to power grid analysis [26]. In [27], Fourier analysis has been explored for several aggregation-based two-grid schemes for a model anisotropic problem. However, in practice, it is too difficult to achieve optimal order convergence with V-cycle or even with standard W-cycle. The work in [28] proposed an aggregation-based AMG preconditioned CG method with Krylov subspace acceleration for power grid analysis, which has been proved to be very close to optimal convergence.

In summary, there is a tradeoff between runtime and memory consumption among the existing methods. Some direct solvers can be very robust but require much more memory resources, while some iterative solvers can be acted as memory efficient but lack of stable convergence rate when simulating extremely large-scale power grids. Some particular fast solvers can be utilized for special or structured power grid simulation such as [29], but mostly suitable for highly structured grids, which may limit its practical applications in general grids. Aiming to break the tradeoff between runtime and memory consumption and obtain a robust convergence with low setup cost, in this paper, we propose an efficient simulator PowerRush with pure AMG solver for static IR-drop analysis of large-scale power grids.

The major contributions of this paper are as follows. Section III provides the simulation flow of our simulator PowerRush, detailed implementations of SPICE parser, and circuit builder, which should have considerable practical value to this community. Section IV presents the proposed AMG preconditioned conjugate gradient (AMG-PCG) solver, which is a pure algebraic method and we have demonstrated how it can improve the convergence robustness without geometric information. In multigrid scheme, double pairwise aggregation technique is applied to matrix in coarsening to ensure low setup cost and memory requirement. Further, K-cycle multigrid scheme is adopted to provide Krylov subspace acceleration at each level to guarantee enhanced robustness and scalability. In particular, Section V presents the simulation performance of large-scale power grids.

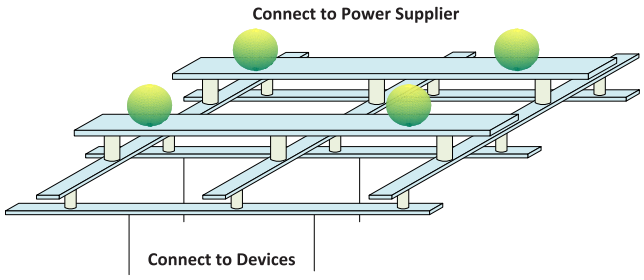


Fig. 2. Power grid model.

II. Background

A. Power Grid Analysis

In this section, we give the basic modeling and analysis techniques for efficient and accurate analysis. Typically, power grid on die is designed from top-level metal layer, which is connected to the package, down through interlayer vias, and finally to the active devices, as shown in Fig. 2. For dc simulation, power grid can be modeled as linear resistive network system. Using modified nodal analysis (MNA) method, a n-node circuit network can be formulated as a linear system [8]. After reformulation by *Norton's law*, this system can result in a symmetric, positive definite problem whose system matrix is a nonsingular $\mathcal{M}$-matrix [19]

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

where $u \in \mathbb{R}^{n\times 1}$ is an unknown vector of node voltages, $G \in \mathbb{R}^{n\times n}$ is the conductance matrix, $I \in \mathbb{R}^{n\times 1}$ is a vector of node current sources. The diagonal entries of matrix G are defined by $g_{ii} = \sum_{j\in N_i} |g_{ij}|$, where $N_i = \{j \,|\, g_{ij} \neq 0\}$ is the set of neighbors of node i, g_{ij} defines the conductance between the two neighboring nodes i and j, thus $g_{ij} = g_{ji}$, which results in the G matrix being symmetric.

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 KLU [6] and Cholmod [7], but both of them are time expensive and memory inefficient for very large scale problems. Another state-of-the-art approach is iterative methods especially preconditioned

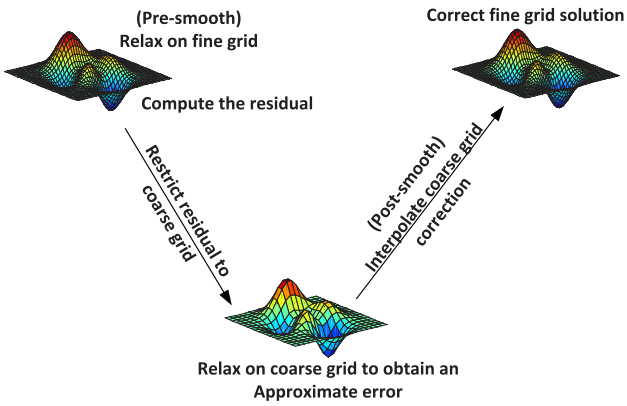


Fig. 3. Essence of multigrid.

iterative methods, which can be utilized to solve such linear systems with memory efficiently. However, preconditioned iterative methods are not stable for many cases because of either expensive cost or unsatisfactory performance of their preconditioners [30].

B. AMG Method

The main idea of multigrid is to improve the convergence of a basic iterative method by global correction on each level, performed by solving a coarse problem. Multigrid methods [31] are considered to be scalable, which are expected to solve a linear system within linear complexity. Multigrid methods achieve optimality through the effect of a smoother and a coarse grid correction. In multigrid scheme, as shown in Fig. 3, the smoother is fixed and generally based on a simple iterative relaxation method. The coarse grid correction involves transferring information to a coarse grid through restriction and computing an approximate solution to the residual equation on a coarser grid, which is to say, solving a linear system of smaller size. This solution is then transferred back to the original grid using an appropriate interpolation, which is also described as prolongation. In the classical multigrid setting, smoothing reduces high frequency error, whereas coarse grid correction eliminates low frequency error.

Multigrid methods are divided into two typical categories, geometric multigrid (GMD) and AMG [32]. GMD can be easily implemented if the geometric problem information is known. AMG methods construct their hierarchy of operators directly from the system matrix, and the levels of the hierarchy are simply subsets of unknowns without any geometric interpretation. Thus, AMG methods become true black-box solvers for sparse matrices. However, AMG is regarded as advantageous mainly where GMD is too difficult to apply.

Due to the irregularity of real power grid designs [33], AMG methods have been well developed in power grid simulation area. AMG is a pure matrix-based method for solving linear equations based on multigrid principles, but requires no explicit knowledge of the problem geometry. It determines coarse grids, intergrid transfer operators, and coarse-grid equations based solely on the matrix entries. With AMG methods, linear systems $Ax = b$ are solved, where A is a real $n \times n$ matrix and x and b are vectors in $\mathbb{R}^n$. For clarity, we assume that A is symmetric positive definite matrix. Recall that the two main components of multigrid are smoothing and coarse grid correction. Coarse grid correction involves operators that transfer information between fine and coarse grids, which are denoted in linear algebra terms simply as the vector space $\mathbb{R}^n$ and the lower dimensional vector space $\mathbb{R}^{n_c}$. Interpolation (prolongation) maps the coarse grid to the fine grid and is just the $n \times n_c$ matrix $P : \mathbb{R}^{n_c} \rightarrow \mathbb{R}^n$. Restriction maps the fine grid to the coarse grid and operators matrix is the transpose of interpolation P^T. The typical two-grid method for solving the above equation can be defined as follows:

1) obtain approximate solution $\tilde{x}$ by presmoothing on fine grid $Ax = b$;
2) compute residual $r = b - A\tilde{x} = Ae$;
3) restrict residual to coarse grid by $r_c = P^T r$;
4) obtain an approximate error e_c by relax on coarse grid $A_c e_c = r_c$;
5) coarse grid correction $x = \tilde{x} + Pe_c$;
6) postsmoothing on the fine grid $Ax = b$.

The error vector e in step 2 reflects the difference between the exact solution and the current iteration. After restricting the residual to coarse grid, the approximate error on coarse grid can be obtained. In practice, we solve the coarse system in step 4 by recursively reapplying the two-grid method, yielding a hierarchy of coarse grids, transfer operators, and coarse grid systems. Because AMG is only based on the matrix A, we only need to define the coarse system A_c with low setup cost for operators P, and then the total AMG framework can be performed efficiently.

Although the classical AMG methods work remarkably well for a wide variety of problems, some of the assumptions made in its derivation usually limit its applicability. In this paper, we introduce an aggregation-based AMG with K-cycle acceleration as a high quality preconditioner for CG iteration to perform efficient static power grid analysis. The smoothed aggregation method [25] is a highly successfully AMG method that is robust and efficient over a wide variety of problems. The most interesting aspect of aggregation-based AMG is its approach to define interpolation. The aggregation algorithm first partitions the grid by aggregation grid points into small disjoint sets and then builds a preliminary interpolation operator to coarsen power grids by system matrix level. Thus, the prolongation matrices with at most one nonzero entry per row are much sparser than the ones obtained by the classical AMG approach. Numerical analysis has shown that for 2-D anisotropic model problem, aggregation-based two-grid methods [27] may have optimal order convergence properties. The constructions of interpolation operator and prolongation operator discussed in Section IV-A will show that how smoothed aggregation method reduces the setup cost. Meanwhile, Krylov subspace acceleration is introduced as cycling strategy of multigrid scheme. The Krylov subspace smoothing is performed at the end of every recursive cycle on each level to reduce the residual. Theoretical analysis has been shown that aggregation-based multigrid with K-cycle can achieve a guaranteed convergence, which is independent or near independent of the number of levels.

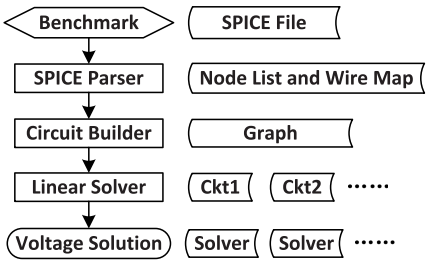


Fig. 4. Simulation flow of PowerRush.

III. Simulation Flow

This paper presents a friendly efficient simulator with remarkable scalability for static power grid analysis. The simulator consists of a smart SPICE parser, a robust circuit builder, and a linear solver. The detailed simulation flow of our simulator PowerRush is shown in Fig. 4. The SPICE format netlist is parsed as nodes list and wires map. Subsequently, it builds them as a topology graph. After extracting them into conductance matrix, the linear solver is called to obtain the voltage solutions by solving the involved linear equations.

A. SPICE Parser

An efficient/robust SPICE parser is critical for circuit simulation especially for large-scale power grids because it is extremely time-consuming to parse a very large scale R/RC/RLC network. For each element of the SPICE netlist, we need to identify how it is connected to the circuit. It is necessary to build a hash table of all circuit nodes when querying them [34], [35].

For each benchmark it reads the netlist twice. The first reading is to count the total number of resistors as N_R and then estimates the total number of circuit nodes to initialize hash table T for all nodes set. The second reading is to hash each circuit node to its index in hash table T by the node name string given in the SPICE netlist. For a node i with name string K_i, we define a string hash function

$$H(K_i) = \sum_{j=1}^{L_i} \mathrm{mod}\left(K_i^j \times M^{(L_i - j)}, N_R\right) \qquad (2)$$

where L_i is the name string length of node i, K_i^j is the jth character of name string K_i, and M is a constant base number, which is selected to be optimal for the above hash function. The idea behind this is that we try to use as many bits as we can while converting strings to integers so that we can obtain a unique integer as far as possible. Of course, we are still not able to obtain unique integers for all the strings and undoubtedly there will be collisions, however, we can try to minimize the collisions as much as possible. According to the properties of the node name strings in SPICE netlist, the 7-bit encoding (or a 128-base number) can be adopted for the above hash function. After properly testing for real benchmarks, we choose an optimal constant base $M = 131$ so that the average clustering in hash table is about 1.16. That is to say, it can ensure the average search length of each node is about 1.16 and the maximum search length is no more than 6.

As we know the complexity of hash method to find an element in a table is expected to $\mathcal{O}(1)$. Therefore, the total complexity of our parser is expected to $\mathcal{O}(N)$, where N is the number of grid nodes.

TABLE I
Grid Reduction Scale for IBM PG Benchmarks

Benchmark	$N_{original}$	$N_{equivalent}$	$\frac{N_{equivalent}}{N_{original}}$
ibmpg3	851584	286299	33.62%
ibmpg4	953583	610103	63.98%
ibmpg5	1079310	540497	50.08%
ibmpg6	1670494	834633	49.96%

B. Circuit Builder

The key step of building circuit is to create a graph to handle all the wires map and nodes list. For more detailed implementations, the reader can refer to Appendix A. In addition, there are some short paths or vertical vias between each two neighboring metal layers with extremely small or zero resistance. Many special considerations should be taken for handling these particular cases to avoid solving the involved ill-conditioned linear equations. Please refer to Appendix B for more details.

Noticing that the number of unknowns can be reduced by merging nodes as short paths (vias with very small or zero resistance value), a highly efficient equivalence-based merging technique is adopted in PowerRush. The disjoint-set data structure is utilized to find and union nodes sets. A disjoint-set data structure is a data structure that keeps track of a set of elements partitioned into a number of disjoint (nonoverlapping) subsets. A union-find algorithm is an algorithm that performs two useful operations on such a data structure. After placing the two nodes between the short path into a same disjoint set, both the find and union operation can be finished within a complexity of $\mathcal{O}(logN)$. Taking IBM power grid benchmarks, for example, the grid reduction scale is listed in Table I. N_{original} is the number of original grid nodes. $N_{\text{equivalent}}$ is the number of equivalent grid nodes. $N_{\text{equivalent}}$ can be about 30%–60% of N_{original} after merging short paths and metal vias with extremely small resistance value. This grid reduction scale can lead to a smaller linear system, which is easier to be solved.

It is noticeable that there exist several separated subnets for each benchmark in real power grid designs [33]. Our circuit builder can identify all the separated subnets by depth-first search (DFS) algorithm with complexity of $\mathcal{O}(N)$ and then solve each subnet independently. Therefore, the total solving cost can be added linearly by the solving cost of each subnet.

C. Linear Solver

After building the circuit graph, the conductance matrix G can be formulated by MNA method as well as the corresponding right-hand side vector. Then, the AMG-PCG method is

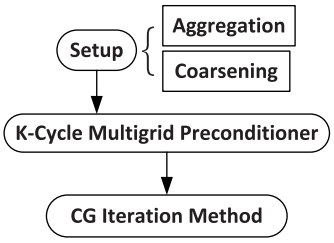


Fig. 5. Framework of AMG-PCG solver.

utilized to solve the resulted linear equations. Among AMG-PCG solver, aggregation-based AMG with K-cycle acceleration is adopted as a preconditioner to improve the robustness of conjugate gradient iterative method [36].

As shown in Fig. 5, the application of AMG-PCG is implemented as a three-part process. The first part, which is a fully automatic setup phase, consists of recursively choosing the coarser levels and defining the transfer and coarse-grid operators. The second part, which is the preconditioning phase, just uses the resulting components to perform acceleration recursively on all the levels by the use of K-cycle scheme. The third part, aggregation-based AMG with K-cycle accelerating is adopted as an implicit preconditioner for CG iterative method to solve the involved linear equations. The detailed implementations of AMG-PCG solver are demonstrated in Section IV.

IV. AMG-PCG Solver

In this paper, we use aggregation-based AMG method with K-cycle acceleration to improve the convergence. This procedure is fully algebraic, that is, it works with the information present in the system matrix only.

A. Aggregation Coarsening

As shown in Fig. 5, the aggregation scheme contains two steps. Initially, we need to partition the unknowns into disjoint subsets to generate the prolongation matrices. Then, the prolongation matrices are utilized to formulate the coarse matrices.

As described in [37], we focus on the schemes that use coarsening by aggregation where the strongest connection is favored in forming pairs. For each unmarked node i, according to some priority rule designed so as to favor a regular covering of the matrix graph. Then, attempt to group this node with another unmarked node with which is most strongly negative coupled, that is, the unmarked node j for which a_{ij} is minimal. For circuit representation, strong coupling means a large-conductance value between each circuit nodes, which has been discussed in [24] and [26]. Since the diagonal entry of conductance matrix $a_{ij}\big|_{j=i}$ is denoted as positive value, the off-diagonal entry $a_{ij}\big|_{j\neq i}$ representing the conductance between each two connected nodes is denoted as negative value. Therefore, the coupling relationship is described as negative coupling. A threshold β is defined to decide whether it is strong or weak coupling, and then define the set of nodes S_i to which i is strongly negative coupled by

$$S_i = \left\{ j \neq i \,\middle|\, a_{\mathrm{ij}} < -\beta \max_{a_{\mathrm{ik}}<0} |a_{\mathrm{ik}}| \right\} \tag{3}$$

which means that the node j is formed as a pair with node i only and if only j satisfies this requirement. As illustrated in Algorithm 1, the pairwise aggregation strategy selects the nodes, which have most of the influence on their neighboring nodes as coarse nodes.

Algorithm 1: Pairwise Aggregation PWA($A_{n\times n}, \beta$)

Input: Conductance matrix $A_{n\times n}$; Strong/Weak coupling threshold β, default as 0.25.
Output: Number of coarse variables n_c and disjoint subset/aggregates G_i, $i = 1, ..., n_c$ which satisfies $G_i \cap G_j = \emptyset$ for each $i \neq j$.
Definition: For each unmarked node i, it satisfies $S_i = \left\{ j \neq i \,\middle|\, a_{ij} < -\beta \max_{a_{ik}<0} |a_{ik}| \right\}$.
Initialize: Define all unmarked nodes as set $U = [1, n]$; For all i, $m_i = |\{j \in U \,|i \in S_i\}|$; $n_c = 0$.

```
Begin
  while U ≠ ∅ do
    Select i ∈ U with minimal m_i
    n_c = n_c + 1
    Select j ∈ U such that a_ij = min_{k∈U} a_ik
    if j ∈ S_i then
      G_{n_c} = {i, j}
    else
      G_{n_c} = {i}
    end
    Remove G_{n_c} from U
    For all k ∈ G_{n_c}, update m_l = m_l − 1 for l ∈ S_k
  end
End
```

With aggregation scheme above, the fine grid with n unknown variables is grouped into coarse grid with n_c disjoint subsets G_i, $i = 1, 2, \ldots, n_c$, and each such subset is associated to a unique coarse level unknown. Prolongation from coarse level to fine level is a vector defined on the coarse variable set by assigning the value at a given coarse variable to all fine grid variables associated to it. The prolongation operator P is a $n \times n_c$ Boolean matrix with exactly one nonzero entry in each row and piecewise constant in each column, that is

$$P_{\mathrm{ij}} = \begin{cases} 1 & \text{if } i \in G_j, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, n; \qquad j = 1, \ldots, n_c \tag{4}$$

and restriction operator R is chosen to be the transpose of the prolongation matrix P.

The coarse grid matrices are then cheap to compute using the variational property of the Galerkin coarse grid operator [31] and generally as sparse as the original fine grid matrix. As we have denoted the conductance matrix G as fine grid matrix A_f, which is symmetric and positive definite, the coarse grid

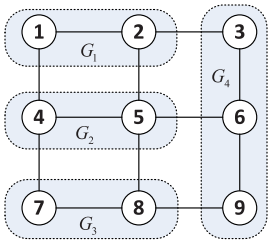


Fig. 6. Sample grid with nine nodes is grouped into four aggregates.

Algorithm 2: Double Pairwise Aggregation $DPWA\,(A_{n\times n}, \beta)$

Input: Conductance matrix $A_{n\times n}$; Strong/Weak coupling threshold β, default as 0.25.
Output: Number of coarse variables n_c and disjoint subset/aggregates G_i, $i = 1, ..., n_c$ which satisfies $G_i \cap G_j = \emptyset$ for each $i \neq j$.

1 **Begin**
2 $G_1^{(1)}, ..., G_{n_{c_1}}^{(1)} = PWA\,(A, \beta)$
3 Compute $A^{(1)}_{n_{c_1}\times n_{c_1}}$ by $a_{ij}^{(1)} = \sum_{k\in G_i^{(1)}} \sum_{l\in G_j^{(1)}} a_{kl}$
4 $G_1^{(2)}, ..., G_{n_{c_1}}^{(2)} = PWA\left(A^{(1)}, \beta\right)$
5 For all $i = 1, ..., n_c$, compute $G_i = \cup_{j\in G_i^{(2)}} G_j^{(1)}$
6 **End**

matrix is computed from the Galerkin formula

$$A_c = RA_fP = P^TA_fP \tag{5}$$

which implies that A_c is cheap to construct using

$$(A_c)_{\text{ij}} = \sum_{k\in G_i} \sum_{l\in G_j} a_{\text{kl}}. \tag{6}$$

That is, the entries in the coarse grid matrix are obtained by summing the entries in A_f that connect the different aggregates. Taking Fig. 6 as an example, the grid with nine nodes is aggregated as a coarse grid with four disjoint sets G_1, G_2, G_3, and G_4, then its prolongation operator P can be as follows:

$$P^T = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{matrix} \leftarrow G_1 \\ \leftarrow G_2 \\ \leftarrow G_3 \\ \leftarrow G_4 \end{matrix}. \tag{7}$$

The above simple pairwise aggregation coarsening is still relatively slow, which cannot guarantee an optimal performance of multilevel methods. This paper adopts the double pairwise aggregation algorithm, which begins by forming pairs in the scaled problem matrix [36]. This fast coarsening technique is implemented in our simulator by repeating the simple pairwise aggregation process, defining aggregates by forming pairs of pairs. The details of double pairwise aggregation algorithm are listed in Algorithm 2.

Furthermore, preliminary numerical results show that aggregation-based multigrid methods may then indeed exhibit convergence that is independent or near independent of the number of levels [27].

Algorithm 3: AMG as Preconditioner at Level k, $z_k = AMGprecond\,(r_k, k)$

Input: The residual r_k of level k
Output: The preconditioned vector z_k

1 **Pre-smoothing**: relax the input residual r_k to v_k by using smoother of several iterations on $A_kv_k = r_k$;
2 **Restriction**: compute a new residual $\tilde{r}_k = r_k - A_kv_k$ and restrict it to coarser grid by $r_{k-1} = P_k^T\tilde{r}_k$;
3 Compute an (approximate) solution y_{k-1} on coarser grid by $A_{k-1}y_{k-1} = r_{k-1}$ which is recursively obtained by $y_{k-1} = AMGprecond\,(r_{k-1}, k-1)$ until the coarsest level with $k = 1$;
4 **Interpolation**: interpolate coarse grid correction by $y_k = P_ky_{k-1}$ and compute new residual $\bar{r}_k = \tilde{r}_k - A_ky_k$;
5 **Post-smoothing**: relax the new residual $\bar{r}_k$ to w_k by using smoother of several iterations on $A_kw_k = \bar{r}_k$;
6 Obtain preconditioned vector $z_k = v_k + y_k + w_k$;

B. Multilevel Preconditioning

Although AMG strategy has been tried to capture all relevant influences by accurate coarsening and interpolation, its interpolation will hardly ever be optimal. Multilevel error reduction technique can smooth the majority of the error components very quickly but inefficiency for just a few exceptional error components. Therefore, its total convergence property is degraded by this nonideal phenomenon. To improve the robustness of our approaches, AMG is adopted as an implicit preconditioner for conjugate gradient method instead of standalone solver [38]. This behavior of scalability enhancement can be understood when we investigate the eigenvalue spectrum of the Richardson iteration matrix. Most of the eigenvalues are generally clustered around zero, only the largest eigenvalues are outside the clustering. As we know the spectral radius determines the convergence of multigrid as a standalone solver. This spectral radius increases on finer grids but the eigenvectors belonging to the larger eigenvalues are very soon captured into the Krylov subspace when multigrid is adopted as a preconditioner, and accordingly the convergence is accelerated considerably.

The AMG preconditioner at each level on a given residual vector is computed according to Algorithm 3, which includes presmoothing, restriction, interpolation, and postsmoothing [36]. As general implementation, symmetric Gauss–Seidel smoothing method is adopted for presmoothing and postsmoothing step. The distinguishing characteristics of our AMG preconditioner are the intergrid operators and multigrid cycling strategy. The intergrid operators, which related to restriction and interpolation steps have been illustrated in Section IV-A, while the cycling strategy will be demonstrated in Section IV-C.

The multigrid preconditioning is called by the main iteration routine at the top level $k = \mathcal{K}$ (fine grid), and it recursively calls itself in step 3 with a smaller index until the coarsest level. We stop the coarsening when the coarse grid matrix has 400 rows, allowing fast direct solver with sparse Cholesky

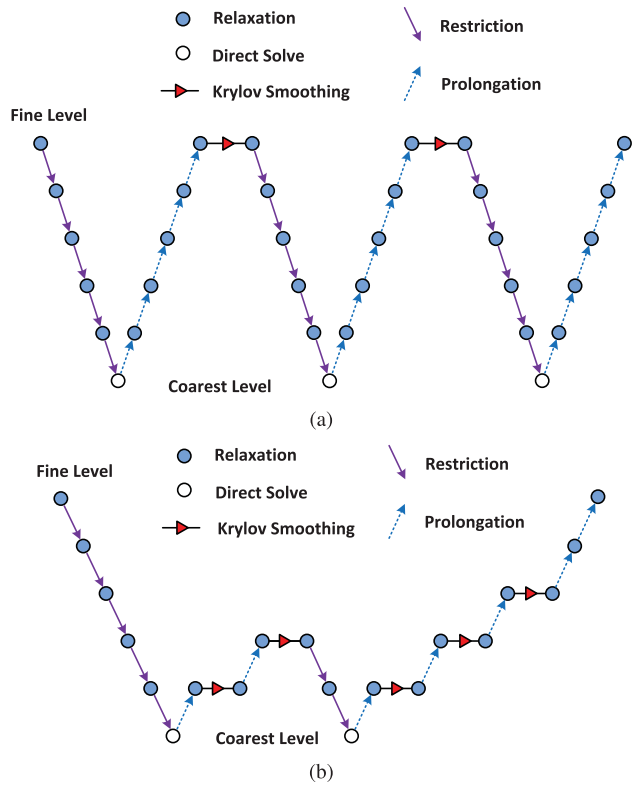


Fig. 7. Schematic description of multigrid cycling strategy. The cycling proceeds from left to right and from top (find level grid) to bottom (coarsest level grid). (a) Multigrid as a preconditioner for a Krylov subspace method. (b) K-cycle: Krylov smoothing is performed on each grid level.

factorization. For $k > 1$, the AMG preconditioner at level k computes y_{k-1} approximately using AMG preconditioner at level $k-1$. The way this is done defines the cycling strategy, which will be discussed in detail in the following section.

Aiming to enhance the convergence robustness and scalability of iterative method, the above aggregation-based AMG is adopted as an implicit preconditioner for conjugate gradient method, which is considered as main (outer) iteration routine. This AMG preconditioner is called at the top level (fine grid) $\mathcal{K}$ among the outline of AMG-PCG solver.

C. K-Cycle Multigrid

The scalability of the general multigrid method can be enhanced or accelerated recursively on all levels by the use of K-cycle strategy [39]. The K-cycle method is so named because the acceleration of multigrid is typically performed by Krylov subspace methods. Typically, accelerating multigrid by a Krylov subspace method at the top level is equivalent to adopting multigrid as a preconditioner in connection with Krylov subspace methods. General K-cycle method can be better understood if we consider the top level acceleration case first. Fig. 7(a) considers to use standard V-cycle multigrid as a preconditioner for Krylov subspace method. After each V-cycle, a Krylov subspace smoothing step is utilized in an attempt to reduce the residual, and that means that $\|A \cdot \tilde{x}_k\|$ should be less than $\|A \cdot x_k\|$ where $\tilde{x}_k$ is the approximate solution computed using the Krylov subspace method. The number of V-cycle iterations needed to reach convergence is hopefully reduced using Krylov smoothing. Although most of the acceleration methods only consider acceleration at the top level there is no reason why acceleration cannot be utilized within each grid level. Not only Krylov smoothing on each coarse grid level is much less time-consuming than Krylov smoothing on top level (fine grid), but also it can improve the scalability of convergence, which has been analyzed in [39].

Fig. 7(b) shows one iteration of K-cycle multigrid strategy, which is presented to accelerate W-cycle. The details of AMG as preconditioner at level k with Krylov subspace acceleration are shown in Algorithm 4, in which the coarse grid system is recursively solved by K-cycle style. This scheme corresponds the one implementation of the preconditioner at the next

Algorithm 4: AMG as Preconditioner at Level k With Krylov Subspace Acceleration, $z_k = AMGprecond\,(r_k, k)$

Input: The residual r_k of level k; The residual threshold t for inner iteration, default as 0.25.
Output: The preconditioned vector z_k
Pre-smoothing: relax the input residual r_k to v_k by using smoother of several iterations on $A_k v_k = r_k$;
Restriction: compute a new residual $\tilde{r}_k = r_k - A_k v_k$ and restrict it to coarser grid by $r_{k-1} = P_k^T \tilde{r}_k$;
Compute an (approximate) solution y_{k-1} on coarser grid by $A_{k-1} y_{k-1} = r_{k-1}$ which is **recursively** obtained by $y_{k-1} = AMGprecond\,(r_{k-1}, k-1)$ until the coarsest level with $k = 1$:
  if $k = 1$ **then**
    directly solve $A_{k-1} y_{k-1} = r_{k-1}$ by $y_{k-1} = cholmod\,(A_{k-1}, r_{k-1})$
  else
    K-Cycle, perform one or two iterations with Multigrid precondition:
      $p_{k-1} = AMGprecond\,(r_{k-1}, k-1)$
      $u_{k-1} = A_{k-1} p_{k-1}$
      $\rho_1 = p_{k-1}^T u_{k-1}$
      $\alpha_1 = r_{k-1}^T r_{k-1}$
      $\tilde{r}_{k-1} = r_{k-1} - \frac{\alpha_1}{\rho_1} u_{k-1}$
      if $\|\tilde{r}_{k-1}\| \le t \|r_{k-1}\|$ **then**
        $y_{k-1} = \frac{\alpha_1}{\rho_1} p_{k-1}$
      else
        $q_{k-1} = AMGprecond\,(\tilde{r}_{k-1}, k-1)$
        $d_{k-1} = A_{k-1} q_{k-1}$
        $\gamma = q_{k-1}^T u_{k-1}$
        $\beta = q_{k-1}^T d_{k-1}$
        $\alpha_2 = \tilde{r}_{k-1}^T \tilde{r}_{k-1}$
        $\rho_2 = \beta - \frac{\gamma^2}{\rho_1}$
        $y_{k-1} = \left(\alpha_1 - \frac{\gamma}{\rho_2}\alpha_2\right) \frac{1}{\rho_1} p_{k-1} + \frac{\alpha_2}{\rho_2} q_{k-1}$
      end
  end
Interpolation: interpolate coarse grid correction by $y_k = P_k y_{k-1}$ and compute new residual $\bar{r}_k = \tilde{r}_k - A_k y_k$;
Post-smoothing: relax the new residual $\bar{r}_k$ to w_k by using smoother of several iterations on $A_k w_k = \bar{r}_k$;
Obtain preconditioned vector $z_k = v_k + y_k + w_k$;

coarser level, whereas a few steps of Krylov subspace iterative smoothing are performed with K-cycle (steps 7–23). According to the theoretical analysis [39], at most two iterations are allowed, and the second iteration (steps 16–22) will be skipped if the relative residual error is less than the threshold t after the first iteration (steps 8–12 and 14). Practically, we set $t = 0.25$ as default.

For the sake of clarity, Fig. 7(b) shows an example of Krylov subspace acceleration recursively on each level. It first performs smoothing and restriction on the top level grid ($k = \mathcal{K}$) five times to obtain the residual of coarsest level grid, then obtains coarse grid correction by directly solving residual equation on coarsest level grid. After interpolating coarse grid correction from coarsest level grid to coarse grid, it performs one or two steps of Krylov subspace smoothing on coarse grid. The Krylov subspace smoothing is performed at the end of every recursive cycle on each level to reduce the residual, in which it is described as inner iteration. In this scheme, the residual on coarse grid system is smoothed by several steps of a Krylov subspace iterative method. This approach is followed recursively until the coarsest level where the solution is obtained through application of the direct solver Cholmod [7]. In practice, K-cycle with only two inner iterations at each level is observed to be optimal.

Theoretical analysis [27], [39] has shown that aggregation-based multigrid with K-cycle can achieve a guaranteed convergence, which is independent or near independent of the number of levels. K-cycle multigrid appears to be more robust than V-cycle or even standard W-cycle. The enhanced robustness is obtained nearly for free because K-cycle has a roughly same computational complexity as V- or W-cycle.

D. Complexity Analysis

PowerRush is designed as a scalable simulator for extremely large-scale power grid analysis. The parser step with hashing method has a linear complexity, and the circuit builder with disjoint set and DFS method has a complexity between $\mathcal{O}(logN)$ and $\mathcal{O}(N)$. That is to say, if our linear solver AMPPCG has an approximate linear complexity, the promised scalable simulator will be hold by PowerRush.

The double pairwise aggregation is adopted in grid coarsening to ensure a relative effective reduction. Because double pairwise means forming pairs of pairs, which means that each four nodes are reduced as one node from one level to the next level, the reduction rate is theoretically 1/4. Numerical analysis has shown that if the grid reduction rate is guaranteed, K-cycle formulation can be utilized at every level while keeping the overall cost bounded [36]. In AMG-PCG solver, the cost of each main (outer) iteration is bounded by

$$Cost \leq \frac{2\sigma}{1-2\sigma}\mathcal{C}nnz\,(A) \tag{8}$$

where $\mathcal{C}$ is a constant such that the cost of one iteration with the multigrid preconditioner at a certain level, $nnz\,(\cdot)$ stands for the number on nonzeros, A is the top level matrix, σ is the grid reduction rate, which is close to 1/4. If the grid reduction rate is 1/4, the cost of each main iteration is bounded by $\mathcal{C}nnz\,(A)$. For a not good grid reduction rate of $\sigma = 3/10$, the cost of each main iteration is at most $3/2\mathcal{C}nnz\,(A)$. Even for a very slow coarsening with the grid reduction rate of $\sigma = 3/8$, the cost of each top level iteration is bounded by $3\mathcal{C}nnz\,(A)$.

It is easy to find that the constant $\mathcal{C}$ is linearly increased with the problem size increasing. And the number of nonzeros of matrix A is also linearly increased with the problem size increasing. Therefore, the total complexity is only effected by the grid reduction rate σ. Once the reduction rate σ is stably performed as a relative small constant, the total complexity bound can be guaranteed as an approximate linear curve.

V. Experimental Results

All algorithms in PowerRush are implemented by C/C++ language with single thread. The simulation platform is a 64-bit Redhat Enterprise Linux with 2 Quad-Core Intel Xeon E5506 CPU@2.13 GHz and 24GB RAM on HP ProLiant DL380 G6 Server. The Hybrid solver [11], which is a well-developed random walk linear solver [40] is also evaluated for comparison. An efficient direct solver Cholmod [7], which is a part of SuiteSparse 3.7.0 [41] is also evaluated for comparison. Due to the symmetric property when performing static analysis, other nonsymmetrical matrix solvers such as SuperLU or KLU will not be evaluated in this paper.

The runtime in simulation flow is measured by system clock counter. The peak memory usage is measured by a friendly tool memtime whose memory usage is fetched from a PID information of /proc/[PID]/stat by sampling [42]. We have to emphasize that, the reported peak memory usage is the total memory used for each simulator, which includes SPICE parser, circuit builder, and linear solver, not only for linear solver.

This paper is focused on efficient algorithms and simulators for static analysis and consequently only dc simulation is performed. There are two sets of benchmarks to evaluate, which include IBM and THU power grid benchmarks. IBM power grid benchmarks are industrial power grids, which are drawn from real chip designs [33]. The smallest one of THU power grid benchmarks is synthesized for a test chip design using Synopsys Astro based on TSMC 65nm technology. The remaining THU benchmarks are extended according to the above synthesized grid by our power grid planner without loss of generality [43]. The matrix conditioning nature of THU benchmarks is properly preserved because we adopt a small real power grid as a seed to create larger cases. All of THU benchmarks are extracted from the above power grid designs as SPICE format, without any special consideration we can benefit from. The details of these benchmarks are listed in Table II. The top six are IBM power grid benchmarks and the last ten are THU power grid benchmarks. The second column is the original grid size, which is from 851 K to 60 million. The third column is the number of equivalent nodes for each benchmark after merging the short paths or vias with extremely small resistance value. The last six columns list the physical parameters for each benchmark. It should be emphasized that the shorts listed in column 7 are the number of zero resistance in SPICE netlist, which is different from the metal vias with extremely small resistance. The ninth column lists the number of subnets existed in each benchmark.

TABLE II

DETAILS OF POWER GRID BENCHMARKS. EQUIVALENT NODES ARE THE NUMBER OF NODES AFTER MERGING SHORT PATHS OR RESISTORS WITH EXTREMELY SMALL VALUE. SUBNET IS THE NUMBER OF ISOLATED NETS, WHICH MEANS NO ELECTRICAL CONNECTIVITY BETWEEN THEM

Benchmark	Nodes	Equivalent Nodes	Resistors	Current Sources	Voltage Sources	Shorts	Subnets	Metal Layers
ibmpg3	851584	286299	1401572	201054	955	0	3	5
ibmpg4	953583	610103	1560645	276976	962	0	2	6
ibmpg5	1079310	540497	1076848	540800	277	538810	5	3
ibmpg6	1670494	834633	1649002	761484	381	835858	2	3
ibmpgnew1	1461041	531777	1422830	357930	955	929261	3	7
ibmpgnew2	1461041	531777	2352355	357930	955	0	3	7
thupg1	4974439	3261850	8248158	315950	261	0	1	3
thupg2	8989132	6014504	15042090	484424	350	0	1	3
thupg3	11778121	7856200	19698081	610610	395	0	1	3
thupg4	15209208	10259771	25566793	747741	434	0	1	3
thupg5	19231049	13627506	33000795	835113	466	0	1	3
thupg6	23505915	17216192	42430212	926447	535	0	1	3
thupg7	28468261	18690184	47430073	1391500	621	0	1	3
thupg8	39784463	27661623	67904272	1703905	703	0	1	3
thupg9	51810571	31518068	83992792	2793853	911	0	1	3
thupg10	60351149	38415188	99645735	3013079	925	0	1	3

TABLE III

DC SIMULATION RESULTS OF IBM POWER GRID BENCHMARKS. THE SETUP COST IS FOR THE THREE SOLVERS AT THE SAME TIME

Benchmark	Setup		PowerRush				Hybrid Solver				Cholmod			
	Parse (Sec)	Build (Sec)	Solve (Sec)	Memory (MB)	E_{max} (mV)	E_{ave} (mV)	Solve (Sec)	Memory (MB)	E_{max} (mV)	E_{ave} (mV)	Solve (Sec)	Memory (MB)	E_{max} (mV)	E_{ave} (mV)
ibmpg3	2.98	0.54	1.70	239.95	0.07	0.0124	10.80	326.02	0.07	0.0127	1.58	314.01	0.05	0.0124
ibmpg4	2.97	0.93	2.04	319.56	0.23	0.0290	12.21	515.71	0.22	0.0272	3.41	480.50	0.23	0.0290
ibmpg5	3.05	0.82	1.75	329.37	0.06	0.0117	25.26	449.31	0.09	0.0162	2.51	466.33	0.05	0.0117
ibmpg6	5.07	1.48	4.14	475.70	0.09	0.0122	44.20	743.18	0.11	0.0245	3.64	664.35	0.05	0.0122
ibmpgnew1	4.29	1.31	3.28	387.97	0.07	0.0124	19.98	577.42	0.07	0.0128	3.53	570.96	0.05	0.0124
ibmpgnew2	5.07	1.06	3.26	407.01	0.07	0.0124	20.17	587.32	0.07	0.0128	3.51	552.88	0.05	0.0123

Many efforts of detail implementations are taken into the parser step and the circuit building step of PowerRush to guarantee a stable simulator for more general power grid designs. Each benchmark is carefully checked in parser procedure to merge the short paths as equivalent nodes and subsequently its original grid size is reduced to a smaller one. Also, the resistors with extremely small resistance value are properly neglected, and vertical vias between each two neighboring layers are handled adaptively. In our simulator, the stopping criteria of linear solver is defined as relative residual

$$tol = \frac{\|r_k\|_2}{\|b\|_2} \tag{9}$$

where $\|r_k\|_2$ is the 2-norm of the residual vector after k iterations, $\|b\|_2$ is the 2-norm of the right-hand side vector, and tol is set to 10^{-6}, which is reliable to guarantee the required voltage solution accuracy.

A. Simulation of Industrial Power Grids

Initially, we carried our various experiments on industrial power grids, which are drawn from real designs [33] to validate the promising performance of the proposed simulator PowerRush. The details of these benchmarks have been listed in Table II. PowerRush is evaluated on these benchmarks compared with Cholmod and Hybrid solver in which their results are shown in Table III. The setup cost that includes parser and builder is all the same for the three solvers. The parser and builder time are accordingly listed in columns 2 and 3. $E_{\max}$ and E_{ave} is to represent the max and average voltage error, respectively. The measured memory is the total memory used for all simulator, not only for linear solver, but also including SPICE parser and circuit builder. The max and average voltage errors of PowerRush are almost the same as the direct solver Cholmod, which is accurate enough for practical analysis.

The runtime of PowerRush is also somewhat like Cholmod solver, from which we can observe that a well-developed direct solver is also a good choice when solving not very large scale power grids. Due to the matrix factorization in Cholmod, the memory consumption of Cholmod solver is much more than PowerRush, which can be found in columns 5 and 13. However, Cholmod has more potential competitive advantage when performing transient simulation with fixed time step because the factorized triangular matrix can be reused on each time point. For runtime comparison, Hybrid solver uses much more CPU time than PowerRush and Cholmod solver. Therefore, both the runtime and memory usage of PowerRush are relatively small and increase slowly with the grid size increasing. As listed in Table IV, the performance of proposed AMG-PCG solver is also compared with GMD solver [21]. T_{GMD} and T_{PR} are the runtime of GMD solver [21] and AMG-PCG solver in PowerRush, respectively. Subsequently, $E_{\text{ave}}^{\text{GMD}}$ and $E_{\text{ave}}^{\text{PR}}$ are the average solution error when using GMD solver [21] and when using AMG-PCG solver in PowerRush, respectively. For GMD solver, the listed runtime and average solution error are listed in [21] of Table I. Even if the working frequency of our used microprocessor is slower than [21], AMG-PCG solver in PowerRush can still achieve

TABLE IV
RUNTIME AND ACCURACY FOR DC SIMULATION COMPARED WITH GMD SOLVER [21]

Benchmark	T_{GMD} (Sec)	T_{PR} (Sec)	Speedup	E_{ave}^{GMD} (mV)	E_{ave}^{PR} (mV)
ibmpg3	16.1	1.70	9X	4.2	0.0124
ibmpg4	28.5	2.04	14X	0.1	0.0290
ibmpg5	25.8	1.75	15X	1.5	0.0117
ibmpg6	48.5	4.14	12X	3.6	0.0122

about 10 speedups with better accuracy than GMD [21]. The experimental results have shown that PowerRush is effective and robust with high accuracy for real power grids analysis.

B. Simulation of Larger Power Grids

Aiming to present the promising enhanced robustness and scalability of PowerRush, a serious of larger scale power grid benchmarks [43] are generated by our power grid planner, in which the grid size is from 5 to 60 million, as listed in Table II. Among them, there exists only one single net in each benchmark, which means that there are no separated nets in a same benchmark. That is to say, it cannot take the advantage of divide-and-conquer strategy in circuit building process, as described in Section III-B.

In Table V, it lists the behavior of our proposed simulator PowerRush on our generated power grids. T_C, T_H, and T_{PR} are the runtime of Cholmod solver, Hybrid solver, and AMG-PCG solver in PowerRush, respectively. M_C, M_H, and M_{PR} are the total memory usage when simulation with Cholmod solver, with Hybrid solver and with AMG-PCG solver, respectively. E_{ave} is to represent average voltage error. The solution of Cholmod solver is adopted as a standard for comparison because there are no golden solutions for the generated benchmarks. T_C/T_{PR} is the speedups of AMG-PCG than Cholmod. T_H/T_{PR} is the speedups of AMG-PCG than Hybrid solver. M_{PR}/M_C is the total memory usage reduction of AMG-PCG than Cholmod. M_{PR}/M_H is the total memory usage reduction of AMG-PCG than Hybrid solver. The setup cost is listed in columns 2 and 3. The runtime of each part in our simulator is shown in Fig. 8. The parsing time, building time, and solving time are very scalable with the increasing size of power grids. The outer iterations of AMG-PCG solver are listed in column 9, which is very robust for large-scale power grid analysis because the robust convergence is based on the double pairwise aggregation and Krylov subspace acceleration techniques.

As listed in Table V, Cholmod solver failed for the largest four benchmarks and Hybrid solver also failed for the largest three benchmarks because of limited memory. However, AMG-PCG solver can simulate all benchmarks correctly. The runtime comparison of three solvers is shown in Fig. 9(a) from which we can observe that AMG-PCG solver is much more efficient than Cholmod and Hybrid solver. The speedups of AMG-PCG solver than Cholmod are listed in column 13, which can obtain about 4X–8X speedups. And speedups of AMG-PCG solver than Hybrid solver are listed in column 14, which can obtain more than 20X or 30X speedups.

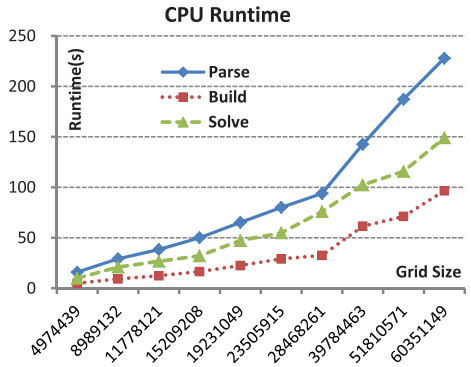


Fig. 8. Runtime of SPICE parser, circuit builder, and AMG-PCG solver in PowerRush simulator.

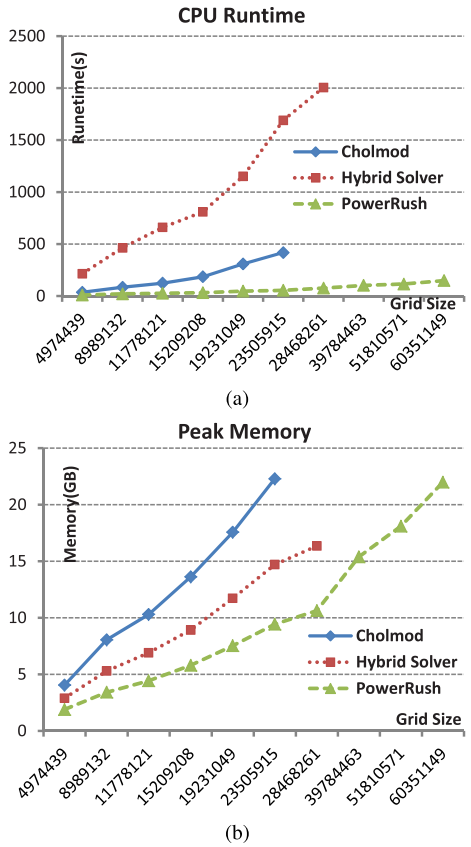


Fig. 9. Runtime and memory complexity of three solvers. (a) Runtime. (b) Memory.

The total memory consumption comparison of three solvers is shown in Fig. 9(b) from which we can observe that the memory usage of AMG-PCG solver is much less than Cholmod and Hybrid solver. AMG-PCG solver can extremely reduce the memory usage using the double pairwise aggregation technique. The memory usage reduction performance is listed in columns 15 and 16. The simulator with AMG-PCG solver just uses about 42% memory than with Cholmod and about 64% memory than with Hybrid solver.

Obviously, there is a tradeoff between Cholmod and Hybrid solver whereas Cholmod is faster than Hybrid solver but

TABLE V

DC SIMULATION RESULTS OF POWERRUSH ON THU POWER GRID BENCHMARKS WHEN COMPARED WITH HYBRID SOLVER AND CHOLMOD SOLVER

Grid	Setup		Cholmod		Hybrid Solver			PowerRush				Speedup		MemReduce	
	Parse (Sec)	Build (Sec)	T_C (Sec)	M_C (GB)	T_H (Sec)	M_H (GB)	E_{ave} (mV)	$Iter$	T_{PR} (Sec)	M_{PR} (GB)	E_{ave} (mV)	$\frac{T_C}{T_{PR}}$	$\frac{T_H}{T_{PR}}$	$\frac{M_{PR}}{M_C}$	$\frac{M_{PR}}{M_H}$
thupg1	15.92	4.96	36.41	4.04	214.17	2.89	1.47	6	9.91	1.89	0.05	4X	22X	47%	65%
thupg2	29.17	9.13	85.11	8.05	464.11	5.29	2.80	6	20.93	3.41	0.06	4X	22X	42%	64%
thupg3	38.37	12.40	123.64	10.29	660.10	6.90	3.61	7	26.73	4.43	0.21	5X	25X	43%	64%
thupg4	50.01	16.54	184.96	13.62	809.94	8.92	4.79	6	32.20	5.79	0.03	6X	25X	43%	65%
thupg5	65.22	22.49	308.95	17.56	1151.04	11.71	7.00	7	47.29	7.53	0.02	7X	24X	43%	64%
thupg6	79.94	29.13	418.35	22.29	1689.16	14.70	9.27	6	54.80	9.43	0.05	8X	31X	42%	64%
thupg7	93.83	32.48	-	-	2005.46	16.35	-	7	75.88	10.63	-	-	26X	-	65%
thupg8	142.64	64.31	-	-	-	-	-	6	102.42	15.39	-	-	-	-	-
thupg9	187.22	71.12	-	-	-	-	-	6	115.75	18.10	-	-	-	-	-
thupg10	227.93	96.49	-	-	-	-	-	6	148.91	21.99	-	-	-	-	-

TABLE VI

GRID REDUCTION EFFICIENCY OF DOUBLE PAIRWISE AGGREGATION SCHEME. THE INITIAL SIZE IS THE EQUIVALENT NODE NUMBER AFTER MERGING THE SHORT PATHS OR VERY SMALL RESISTORS

Benchmark	Initial Size	Reduction Level									Reduction Rate*
		# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	
ibmpg3/GND†	151148	42535	11817	3250	880	239	-	-	-	-	0.2753
ibmpg4/GND†	303712	78876	21527	5822	1552	407	105	-	-	-	0.2650
ibmpg5/GND†	291559	75523	20095	5320	1412	377	-	-	-	-	0.2645
ibmpg6/GND†	430586	114936	31314	8657	2421	663	174	-	-	-	0.2720
ibmpgnew/GND†	272655	75804	21113	5804	1574	416	108	-	-	-	0.2711
thupg1	3261850	1016250	302693	82384	21244	5404	1369	347	-	-	0.2715
thupg2	6014504	1876044	559178	152303	39264	9973	2524	640	163	-	0.2694
thupg3	7856200	2448528	729549	198959	51293	13032	3310	838	213	-	0.2694
thupg4	10259771	3198126	952769	259647	66915	16990	4303	1089	275	-	0.2691
thupg5	13627506	4246482	1265492	344802	88799	22548	5720	1445	366	-	0.2691
thupg6	17216192	5365326	1598894	435519	112230	28500	7207	1824	460	117	0.2674
thupg7	18690184	5824651	1735814	473105	121978	30992	7852	1985	502	128	0.2676
thupg8	27661623	8620564	2568323	700007	180413	45840	11607	2941	744	188	0.2674
thupg9	31518068	9817025	2925390	797496	205484	52169	13205	3342	847	215	0.2675
thupg10	38415188	11967133	3566144	972293	250721	63664	16109	4074	1030	264	0.2677

† GND means that only the single net of ground network is used.

* Average reduction rate of all reduction levels

Cholmod is memory inefficient than Hybrid solver. Because Cholmod is a direct solver, which requires more memory resources while Hybrid solver just needs to construct a preconditioner whose fill-in can be largely controlled. There is no doubt that the quality of preconditioner will naturally affect the preconditioning performance of iterative solvers and consequently require more runtime. However, our proposed AMG-PCG solver can reduce the solution residual by a multilevel fashion while it can not only improve the solving efficiency, but also reduce the memory consumption. In addition, the voltage error when using AMG-PCG is much smaller than with Hybrid solver, as shown in columns 8 and 12. Since the multilevel approach is essentially better than random walk approach, the solution accuracy of AMG-PCG solver is much more scalable than Hybrid solver when simulating larger scale power grids.

PowerRush is also evaluated in first annual TAU power grid simulation contest [44]. The contest results show that PowerRush not only has a better solving efficiency, but also is extremely memory efficient. In addition, PowerRush is expected to explore a widely practical use in industrial world. For more detailed results, the readers can refer to [28] and [45]–[47].

C. Grid Reduction Efficiency

Also, in Table VI, it lists the grid reduction efficiency of double pairwise aggregation strategy. For all benchmarks, the total average reduction rate is about 0.2689, which is very close to $\sigma = 1/4$ while the scale factor is about 3.7184. Therefore, the complexity is bounded by

$$\frac{2\sigma}{1-2\sigma}\mathcal{C}nnz\,(A) = 1.1636\mathcal{C}nnz\,(A)\,. \tag{10}$$

Considering a scaling rule for this reduction scheme with

$$400 \times (3.7184)^{10} \approx 202125475 \tag{11}$$

it can reduce a power grid of about 200-million nodes easily to 400 nodes within 11 levels. Thus, the double pairwise aggregation technique reduces the memory consumption rapidly, which ensures the scalability in memory usage.

VI. Conclusion

We have presented the detailed implementations of PowerRush. Aggregation-based AMG with K-cycle accelerating is adopted as an implicit preconditioner for CG iterative method to solve the involved linear equations for static power grid simulation. Double pairwise aggregation scheme adopts two passes of a pairwise matching algorithm to ensure low setup cost both in runtime and memory usage. K-cycle scheme is a recursively accelerated W-cycle where acceleration is performed by finding the optimal linear combination of two iterations (inner and outer iteration). Thus, a friendly scalable solver is realized fully with algebraic information on the involved system matrix only. Also, by taking the advantage of effective SPICE parser and robust circuit builder, PowerRush has shown to be an effective and robust simulator with excellent scalability for real power grids analysis.

Appendix A
Additional Implementation of SPICE Parser and Circuit Builder

The SPICE parser first builds each kind of elements as a set, that is to say, all circuit nodes set as S_{node}, all resistors set as S_{wire}, all pins, which are connected to current sources as S_{pin} and all pad nodes, which are connected to voltage sources as S_{pad}. By traversing on these sets, the power grid circuit is stored as nodes list and wires map, which are linked to present their topologies. Each circuit component is pointed to its two side nodes, such as for a resistor $w \in S_{\text{wire}}$ with two side nodes $p \in S_{\text{node}}$ and $q \in S_{\text{node}}$, we have $q \leftarrow w \rightarrow p$, and thus we can store the relationships among all circuit components and nodes as a link table $L_{\text{wire}\rightarrow\text{node}}$.

Based on the link table $L_{\text{wire}\rightarrow\text{node}}$, we can build the circuit topology graph G. For each wire $w \in S_{\text{wire}}$, which is not ignored, the wire w is pointed to its two side equivalent nodes, and also each equivalent node is pointed to the wire, which is connected to it. Thus, the circuit topology is constructed by a two-way mapping relationship by the wires map and nodes list. The above two-way mapping relationship is denoted as graph G, which can be easily extracted as a conductance matrix for simulation.

By merging the short paths, the metal wires on short paths are removed from S_{wire} and the components, which are pointed to the original nodes will be pointed to the equivalent nodes. As shown in Fig. 10, wire 1 is pointed to node b and c. If there is a short path between node a and b, it will merge them and choose node b as their equivalent node without loss of generality. Then, node a is removed from the nodes list and the voltage solution of node a can be obtained from its equivalent node b after solving the linear system. Meanwhile, node b and c are pointed to wire 1.

Appendix B
On the Perspective of Preconditioning for Power Grid Network

The involved ill-conditioned problems are arisen from the dramatically conductance variations in different metal layers or vias, which are similar to the ill-conditioned finite element stiffness matrices [48]. With the industrial CMOS process becoming more advanced, via resistance value will exceed the sheet resistance by several orders of magnitude, and consequently the max to min ratio of eigenvalues of conductance matrix becomes too large to converge when using iterative methods.

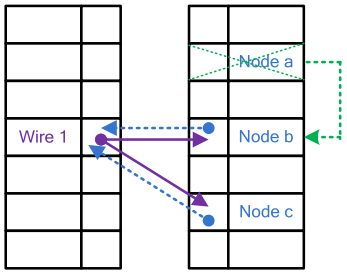


Fig. 10. Link table for wire map and nodes list.

From the perspective of preconditioning for power grid network [33], R_{via} (typical resistance value of vertical vias) is much smaller than R_{horizon} (in horizontal metal layers), as shown in Fig. 11. For ibmpg3, R_{horizon} is about 10^{-2} ohm, whereas R_{via} is about 10^{-8} ohm. Accordingly, the condition number of system conductance matrix is too large to obtain a convergence solution for iterative method. If directly ignoring R_{via} by merging the two nodes as an equivalent node it will result in some considerable voltage error in final solutions. Subsequently, the error becomes more and more significant along with R_{via} increasing. For ibmpg3, the nodes number of GND net is 441 076, which can be reduced to 151 148 after merging the vias with extremely small resistance value (about 10^{-8} ohm) and then it leads to a well-conditioned system to be solved. For clarity, several experiments are carried out to demonstrate the influence on numerical characteristic due to the difference between R_{via} and R_{horizon}. Initially, R_{horizon} is fixed, and then R_{via} varies from 10^{-8} to 10^{-1} ohm. Subsequently, we monitor the IR-drop on vertical vias, which are listed in Table VII. The condition number is computed by condest function in MATLAB. Solution unreliable means that the solution is not reliable because of large condition number. The condition number is almost infinity when R_{via} is too small, and decreases with R_{via} increasing because of R_{via} is becoming more and more close to R_{horizon}. Also, the IR-drop on vertical vias is increasing when R_{via} is increasing. Obviously, there exists a tradeoff between the solution accuracy and the condition number. We should set a proper threshold to decide when R_{via} can be ignored under the condition of solution accuracy requirement.

An approximation strategy is proposed to tackle this ill-conditioned problem in our simulator. The idea of adaptively ignoring vertical vias is based on the magnitude of the voltage drop on via. That is, the voltage drop on via is very small when g_{via} is very large, and subsequently the via can be treated as a short path. Thus, the negligible voltage drop can properly be ignored while avoiding to solve the ill-conditioned problem. Before building the conductance matrix, we try to estimate an average current flow through the vias between

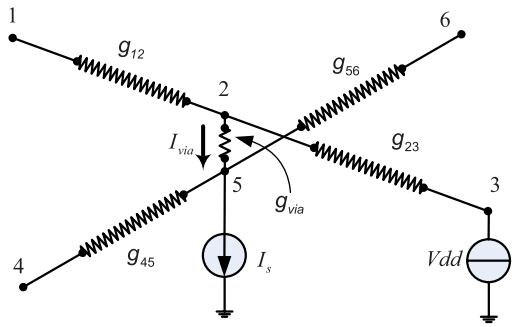


Fig. 11. Via between neighboring metal layers.

TABLE VII
COMPARISON OF CONDITION NUMBER AND IR-DROP ON VIA WITH DIFFERENT RESISTANCE VALUE OF VERTICAL VIAS FOR IBMPG3 CASE

R_{via}	Condition Number	IR-drop on Via
10^{-8}	Inf.	Solution unreliable
10^{-7}	Inf.	Solution unreliable
10^{-6}	$\sim 10^{9}$	Solution unreliable
10^{-5}	$\sim 10^{8}$	Solution unreliable
10^{-4}	$\sim 10^{7}$	$\sim 0.01\ mV$
10^{-3}	$\sim 10^{6}$	$\sim 0.1\ mV$
10^{-2}	$\sim 10^{5}$	$\sim 1\ mV$
10^{-1}	$\sim 10^{4}$	$\sim 10\ mV$

each two neighboring metal layers. Because the current sinks are only attached to the nodes on bottom metal layer, we can calculate the total current sinks by adding them together as I_{total}. Also, we count the total number of vias between every two neighboring metal layers as N_k^l between metal layer k and l. By assuming that the current flow through the vias between certain two layers are somewhat equilibration, then the average current flow through each via between layer k and l can be estimated as $I_{\text{via}}^{kl} = I_{\text{total}}/N_k^l$. Assuming the required accuracy of voltage solution is set to ε and the typical value of vias resistance is estimated as r_{typical}, whether the vias resistance should be ignored or not can be decided by this formula with a threshold μ

$$\begin{cases} r_{\text{typical}} \times I_{\text{via}}^{kl} \le \mu \times \varepsilon & \text{ignore} \\ r_{\text{typical}} \times I_{\text{via}}^{kl} > \mu \times \varepsilon & \text{do not ignore} \end{cases} \tag{12}$$

where $\mu \approx 0.01$ is a rough factor to scale the solution accuracy ε. The solution accuracy ε is usually specified by chip designers, such as defining ε as 1 mV. By scaling it with $\mu = 0.01$ or even smaller, that is, the corresponding solution error is scaled to 0.01 mV as a threshold, then to decide whether the vias will be ignored or not. The parameter μ is just a rough factor, which is usually chosen by scaling estimation. For IBM power grid benchmarks, it is reasonable to set μ as about 0.01 by our observations.

This formula means that if the typical resistance value of vias is small enough to ignore the voltage drop on them we can merge them as equivalent nodes instead of solving a very ill-conditioned system. Experiments will show that this estimation strategy is proper and effective. It can not only avoid to solve the ill-conditioned problem, but also reduce the problem size by merging nodes as equivalent node on a large scale.

ACKNOWLEDGMENT

The authors would like to thank Dr. H. Qian from the T. J. Watson Research Center of IBM for providing them with the Hybrid solver library.

REFERENCES

[1] H. Smith, "Power delivery and analysis in IBM," IBM Corporation, Armonk, NY, USA, Keynote Rep., 2011.

[2] S. R. Nassif, "Power grid simulation—The chip view," IBM Corporation, Armonk, NY, USA, Tech. Rep., 2010.

[3] R. Berridge, R. M. Averill, III, A. E. Barish, M. A. Bowen, P. J. Camporese, J. DiLullo, P. E. Dudley, J. Keinert, D. W. Lewis, R. D. Morel, T. Rosser, N. S. Schwartz, P. Shephard, H. H. Smith, D. Thomas, P. J. Restle, J. R. Ripley, S. L. Runyon, and P. M. Williams, "IBM POWER6 microprocessor physical design and design methodology," *IBM J. Res. Develop.*, vol. 51, no. 6, pp. 685–714, Nov. 2007.

[4] J. Friedrich, R. Puri, U. Brandt, M. Buehler, J. DiLullo, J. Hopkins, M. Hossain, M. Kazda, J. Keinert, Z. M. Kurzum, D. Lamb, A. Lee, F. Musante, J. Noack, P. J. Osler, S. Posluszny, H. Qian, S. Ramji, V. Rao, L. N. Reddy, H. Ren, T. Rosser, B. R. Russell, C. Sze, and G. Tellez, "Design methodology for the IBM POWER7 microprocessor," *IBM J. Res. Develop.*, vol. 55, no. 3, pp. 9:1–9:14, May/Jun. 2011.

[5] V. Zyuban, J. Friedrich, C. J. Gonzalez, R. Rao, M. D. Brown, M. M. Ziegler, H. Jacobson, S. Islam, S. Chu, P. Kartschoke, G. Fiorenza, M. Boersma, and J. A. Culp, "Power optimization methodology for the IBM POWER7 microprocessor," *IBM J. Res. Develop.*, vol. 55, no. 3, pp. 7:1–7:9, May/Jun. 2011.

[6] T. A. Davis and S. Rajamanickam, "Algorithm 907: KLU, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.*, vol. 37, no. 3, pp. 36–52, Sep. 2010.

[7] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: Cholmod, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22–35, Oct. 2008.

[8] T. H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. 38th IEEE/ACM DAC*, Jun. 2001, pp. 559–562.

[9] 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, Aug. 2005.

[10] X. Zhao, J. Wang, Z. Feng, and S. Hu, "Power grid analysis with hierarchical support graphs," in *Proc. IEEE/ACM ICCAD*, Nov. 2011, pp. 543–547.

[11] H. Qian and S. S. Sapatnekar, "Stochastic preconditioning for diagonally dominant matrices," *SIAM J. Sci. Comput.*, vol. 30, no. 3, pp. 1178–1204, 2008.

[12] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw, "Hierarchical analysis of power distribution networks," in *Proc. 37th IEEE/ACM DAC*, Jun. 2000, pp. 150–155.

[13] K. Sun, Q. Zhou, K. Mohanram, and D. C. Sorensen, "Parallel domain decomposition for simulation of large-scale power grids," in *Proc. IEEE/ACM ICCAD*, Nov. 2007, pp. 54–59.

[14] 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.

[15] 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.

[16] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, "A fast solver for HSS representations via sparse matrices," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 1, pp. 67–81, 2006.

[17] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numer. Linear Algebra Appl.*, vol. 17, pp. 953–976, Dec. 2010.

[18] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "Fast power grid simulation," in *Proc. IEEE/ACM DAC*, Jun. 2000, pp. 156–161.

[19] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 10, pp. 1148–1160, Oct. 2002.

[20] H. Su, E. Acar, and S. R. Nassif, "Power grid reduction based on algebraic multigrid principles," in *Proc. 40th IEEE/ACM DAC*, Jun. 2003, pp. 109–112.

[21] Z. Feng and P. Li, "Multigrid on GPU: Tackling power grid analysis on parallel SIMT platforms," in *Proc. IEEE/ACM ICCAD*, Nov. 2008, pp. 647–654.

[22] Z. Feng and Z. Zeng, "Parallel multigrid preconditioning on graphics processing units (GPUs) for robust power grid analysis," in *Proc. 47th IEEE/ACM DAC*, Jun. 2010, pp. 661–666.

[23] Z. Feng, X. Zhao, and Z. Zeng, "Robust parallel preconditioned power grid simulation on GPU with adaptive runtime performance modeling and optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4, pp. 562–573, Apr. 2011.

[24] C. Zhuo, J. Hu, M. Zhao, and K. Chen, "Power grid analysis and optimization using algebraic multigrid," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 738–751, Apr. 2008.

[25] P. Vaněk, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing*, vol. 56, no. 3, pp. 179–196, 1996.

[26] P. Y. Huang, H.-Y. Chou, and Y.-M. Lee, "An aggregation-based algebraic multigrid method for power grid analysis," in *Proc. 8th IEEE ISQED*, Mar. 2007, pp. 159–164.

[27] Y. C. Muresan and Y. Notay, "Analysis of aggregation-based multigrid," *SIAM J. Sci. Comput.*, vol. 30, no. 2, pp. 1082–1103, 2008.

[28] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "PowerRush: A linear simulator for power grid," in *Proc. IEEE/ACM ICCAD*, Nov. 2011, pp. 482–487.

[29] J. Shi, Y. Cai, W. Hou, L. Ma, S. X.-D. Tan, P.-H. Ho, and X. Wang, "GPU friendly fast Poisson solver for structured power grid network analysis," in *Proc. IEEE/ACM DAC*, Jul. 2009, pp. 178–183.

[30] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.

[31] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*. San Francisco, CA, USA: Academic, 2001.

[32] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphia, PA, USA: SIAM, 2000.

[33] S. R. Nassif. (2011, Aug.). *IBM Power Grid Benchmarks* [Online]. Available: http://dropzone.tamu.edu/~pli/PGBench

[34] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA, USA: Addison-Wesley, 1974, pp. 111–113.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009, pp. 219–232.

[36] Y. Notay, "An aggregation-based algebraic multigrid method," *Electron. Trans. Numer. Anal.*, vol. 37, no. 6, pp. 123–146, 2010.

[37] Y. Notay, "Aggregation-based algebraic multilevel preconditioning," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 4, pp. 998–1018, 2006.

[38] C. W. Oosterlee and T. Washio, "On the use of multigrid as a preconditioner," in *Proc. 9th Int. Conf. Domain Decomposit. Methods*, 1996, pp. 441–448.

[39] Y. Notay and P. S. Vassilevski, "Recursive Krylov-based multigrid cycles," *Numer. Linear Algebra Appl.*, vol. 15, no. 5, pp. 473–487, 2006.

[40] H. Qian and S. S. Sapatnekar. (2012, Aug.). *Random Walk Based Hybrid Solver 1.2.1* [Online]. Available: http://www.ece.umn.edu/users/sachin/software/hybridsolver/index.html

[41] T. A. Davis. (2011, Dec.). *Suitesparse 3.7.0* [Online]. Available: http://www.cise.ufl.edu/research/sparse/SuiteSparse

[42] J. Yang and Z. Li. (2012, Jul.). *Memtime* [Online]. Available: http://tiger.cs.tsinghua.edu.cn/Students/yangjl/memtime

[43] J. Yang and Z. Li. (2012, Jul.). *THU Power Grid Benchmarks* [Online]. Available: http://tiger.cs.tsinghua.edu.cn/PGBench

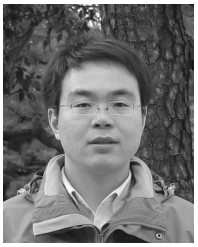
[44] Z. Li. (2012, Apr.). *TAU 2011 Power Grid Analysis Contest* [Online]. Available: http://www.tauworkshop.com/PREVIOUS/tau_2011_contest.pdf

[45] Z. Li, R. Balasubramanian, F. Liu, and S. Nassif, "2011 TAU power grid simulation contest: Benchmark suite and results," in *Proc. IEEE/ACM ICCAD*, Nov. 2011, pp. 478–481.

[46] C.-H. Chou, N.-Y. Tsai, H. Yu, C.-R. Lee, Y. Shi, and S.-C. Chang, "On the preconditioner of conjugate gradient method—A power grid simulation perspective," in *Proc. IEEE/ACM ICCAD*, Nov. 2011, pp. 494–497.

[47] Z. Zeng, T. Xu, Z. Feng, and P. Li, "Fast static analysis of power grids: Algorithms and implementations," in *Proc. IEEE/ACM ICCAD*, 2011, pp. 488–493.

[48] S. Gen-Hua, "Direct-iterative solution of ill-conditioned finite element stiffness matrices," *Int. J. Numerical Methods Eng.*, vol. 18, no. 2, pp. 181–194, Feb. 1982.



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.

He is involved in research with the EDA Laboratory. His current research interests include numerical algorithms for VLSI power grid analysis and verification.

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

Zuowei Li received the B.S. degree in computer science and technology from the Wuhan University of Science and Technology, Wuhan, China, in 2009, and the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012.

He has been with Nimbus Automation Technologies, Shanghai, China, since 2012.

Mr. Li was a recipient of the first place award of the TAU Power Grid Simulation Contest in 2011 and the second place award 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. His current research interests include VLSI layout theory and algorithms.