

Parallel domain decomposition simulation for large-scale power grid

Kai Sun[†], Quming Zhou[‡], Kartik Mohanram[‡], Danny C. Sorensen[†]

[†]Department of Computational and Applied Mathematics, Rice University, Houston, USA
{kleinsun, sorensen}@rice.edu

[‡]Department of Electrical and Computer Engineering, Rice University, Houston, USA
{quming, kmram}@rice.edu

ABSTRACT

This paper presents a fully parallelized domain decomposition (DD) technique for efficient simulation of large-scale linear circuits such as power grids. Domain decompositions using non-overlapping and overlapping partitions are developed to conquer the numerical difficulties in simulation. Experimental results show that by the proposed parallel DD framework, existing linear circuit simulators can be improved for large-scale power grids. Detailed analysis indicate that DD combining multi-grid is suitable for DC simulation and that DD combining LU decomposition performs best for transient simulations.

Categories and Subject Descriptors: B.7.2 [Integrated circuits]: Design aids—simulation

General Terms: Algorithms, design

Keywords: Power grid, domain decomposition, large scale system

1. INTRODUCTION

With increasing power consumption and faster operating frequency of microprocessors, the design of effective power distribution networks has emerged as a critical design challenge. The parasitics associated with the power distribution network and time-varying circuit currents induce supply voltage variations across the integrated circuit. Such power supply variations may impact circuit performance and compromise noise immunity. Extensive simulations are usually performed to identify and correct such instances during design. The power distribution network typically has a grid structure and is commonly referred to as the power grid. The power grid for a modern integrated circuit may consist of several million electrical elements, which makes simulation computationally (time-resource) intensive.

The power grid is traditionally described as a large-scale linear system. Simulation of power grids usually consists of both DC and transient analysis. DC analysis is used to estimate the IR voltage drop by solving the system once, whereas transient analysis needs simulate the system several times to verify the Ldi/dt voltage drop.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Although several techniques have been developed in literature to simulate power grids, no fully parallelized simulator has been presented. In this paper, based on techniques of divide-and-conquer, we present a parallel domain decomposition [12, 13] approach for power grid analysis. The proposed approach allows solving several subcircuits in parallel to obtain the solution of the original circuits. Domain decomposition leads to parallelism through assigning subproblems to multiple processors. Parallel computing has received revived attention because of the multi-core architecture in mainstream computers. Our approach has improved simulation performances in comparison to the original circuit simulator. Simulation results show that by the proposed DD framework, existing linear circuit simulators can be extended to handle otherwise intractable systems, e.g., a circuit with 10M nodes.

The rest of the paper is organized as follows. Section 2 presents a background on power grid analysis. Section 3 describes the proposed approach using domain decomposition. Section 4 describes a domain decomposition preconditioner. Section 5 includes simulation results and discussions. Section 6 gives a conclusion.

2. BACKGROUND

The power grid can be described using the Modified Nodal Analysis (MNA) [10] as

$$Gx(t) + C\dot{x}(t) = u(t), \quad (1)$$

where x is a n dimensional real vector of node voltages and inductor currents, G is a $n \times n$ conductance matrix, C denotes the capacitance and inductance terms, and $u(t)$ includes the loads and voltage sources. n can be of the order of millions for reasonable size power grids.

By applying the backward-Euler method to the system of equations in (1), we obtain

$$(G + C/h)x(t+h) = u(t+h) + Cx(t)/h, \quad (2)$$

where h is a fixed time step for the transient analysis. The system of equations (2) can be rewritten as

$$Ax(t+h) = B, \quad (3)$$

where $A = G + C/h$ and $B = u(t+h) + Cx(t)/h$. The matrix A usually lacks the sparse symmetric positive definite structure when inductance effects are considered. As a result, the Cholesky decomposition is not applicable in our simulation.

Two kinds of approaches widely used to solve Eqn. (3) are direct solvers and iterative solvers. Both have been implemented as solvers of subdomains in our DD framework. Our results will demonstrate that the choice of solvers has profound impacts on the

simulation performance in terms of CPU time and memory requirements. Here, we give a brief introduction to relative approaches.

Direct solvers perform a factorization such as Gaussian elimination on the matrix A to get lower and upper triangular matrices L and U . The system is then solved using forward and backward substitutions on the vector B . After factorization, direct solvers are usually much faster than iterative solvers. Whereas factorization is significantly more expensive than the substitution pass, direct factorization of A by LU decomposition may be still justified in transient analysis. This is because the factorization can be reused in each time step, and its cost is negligible when amortized over hundreds of time steps in transient analysis. The primary drawbacks of direct methods are memory requirement and fill-in. Direct factorization of a large matrix A is computationally (CPU time and memory) intractable as the size is growing.

Iterative methods [12] use simple matrix operations to avoid the costly direct factorization. Two of the most common computational kernels are matrix-vector multiplication or transpose matrix-vector multiplication. Iterative solvers can be very fast when solving a system a few times. However, as the number of simulation steps increase, they are inefficient due to the similar workload at each time step. The cumulative costs over several time steps during transient analysis may exceed the cost of direct factorization.

A good preconditioner is necessary for most iterative solvers. Preconditioning is a kind of modification of the original problem that accelerates the iterative methods. For example, in solving a linear system $Ax = B$, an explicit or implicit operator P^{-1} such that $P \approx A$ is sought to solve $P^{-1}Ax = P^{-1}B$ efficiently. The preconditioners used in this paper are incomplete factorization, algebraic multi-grid, and additive Schwarz. For a sparse matrix A , the incomplete factorization computes a sparse lower triangular matrix L and a sparse upper triangular matrix U . The purpose is to make the residual $A - LU$ satisfy the constraints. The basic idea of multi-grid preconditioning is to approximate the original system by interpolations of a smaller system on coarse grids. The additive Schwarz preconditioner uses domain decomposition as a preconditioner, and will be elaborated in this paper.

2.1 Previous solutions

Several simulation techniques have been developed for power grid simulation and analysis in literature. Sparsity and the grid structure in the power distribution network are usually exploited to reduce computational complexity [1, 6, 14, 18]. A preconditioned conjugate gradient iterative method, using incomplete Cholesky factorization as the preconditioner, was described in [1]. Although this preconditioner-based iterative method reduces the computational complexity of DC analysis of power grids to $O(n^2)$, it is not efficient for transient analysis since it is not possible leverage previous simulation runs. A multi-grid approach that also exploits the grid structure by mapping the original system to a coarsened grid, solving the coarsened grid, and remapping back to the original grid was described in [6]. The solution of the original system through remapping is obtained through an interpolation procedure. However, in the absence of error bounds, this method may not always be accurate. Moreover, the effort to keep track of the geometrical information of the power grid is expensive, further limiting its applications. Algebraic multi-grid methods were proposed in [14] and [18] to handle general network topologies. Algebraic multi-grid methods can be thought of as iterative solvers that use the multi-grid operator as a preconditioner. In such methods, the computational cost in each time step of the transient analysis is comparable to that for DC analysis, making it unsuitable for efficient transient analysis. Other approaches to power grid analysis include

random walks, model order reduction, and hierarchical analysis. Statistical techniques based on random walks [8, 11] are very fast but suffer from accuracy loss and convergence issues. Model order reduction methods are inefficient for power grid simulation due to (i) a large number of external terminals and (ii) the loss of sparsity in the reduced model [4, 16]. Hierarchical techniques are applicable if the power grid is not flattened, and macro-models for local grids can be built to speed up simulation at the global level [7, 17].

In this paper, we propose a divide-and-conquer approach based on DD. It complements traditional factorization-based methods as well as preconditioner-based iterative methods, and provide improvements in both runtime and problem tractability.

3. DOMAIN DECOMPOSITION

Domain decomposition methods refer to techniques of divide-and-conquer that have been primarily developed for solving partial differential equations [12] [13]. They are based on the general concepts of graph partitioning. We present two domain decomposition techniques, Schur complement and additive Schwarz preconditioner, with different involved domain partitions.

3.1 Non-overlapping domain decomposition

Suppose the power grid described by (3) has been partitioned into m subdomains Ω_i , $i = 1, 2, \dots, m$ by reordering the variables. The nodes in the original system is classified into (i) interior nodes of subdomains and (ii) interface nodes. Note that interface nodes are not contained in any of the subdomains. Based upon this, for a general partitioning of the original system into m subdomains, (3) has the following structure:

$$\begin{pmatrix} A_1 & & & E_1 \\ & A_2 & & E_2 \\ & & \ddots & \vdots \\ & & & A_m & E_m \\ F_1 & F_2 & \cdots & F_m & A_\Gamma \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ y \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \\ g \end{pmatrix} \quad (4)$$

In this system, the matrices A_1, A_2, \dots, A_m correspond to the m subdomains, A_Γ corresponds to the interface nodes, and E_1, E_2, \dots, E_m and F_1, F_2, \dots, F_m are matrices that capture connectivity information between the interface and the corresponding subdomain. Each x_i represents the sub-vector of state variables that are interior to subdomain Ω_i , and y represents the sub-vector of all interface variables. The matrices f_1, f_2, \dots, f_m correspond to the loads and voltage sources contained within the corresponding subdomain Ω_i , and the matrix g corresponds to the loads and voltage sources at the interface nodes. Figure 1 shows an example of the partition.

THEOREM 3.1. *If there is no direct coupling capacitance and no direct mutual inductance between subdomains, the systems described by equations (3) and (4) are equivalent.*

PROOF. When DD is used to partition the original system, the nodes are classified into either interior nodes or interface nodes. Cross terms between x_i and x_j ($i \neq j$) are encountered during MNA only if coupling capacitances or mutual inductances exist in the system. Since such effects can be negligible at the interface nodes (see note below), the cross terms are zero and the two systems are equivalent. \square

Note that the assumption of Theorem 3.1 is generally true because mutual inductance effects in the power grid are still negligible [15]. Furthermore, flip-chip technology uses C4 bumps that provide a more direct power delivery path in modern designs, resulting in power grid shells [2] and localized responses at frequencies greater than 250MHz [9]. These properties can be exploited to

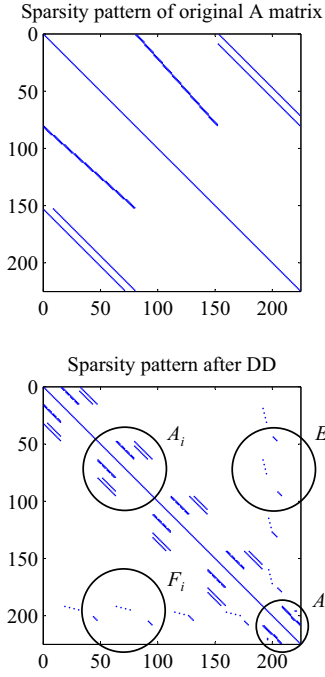


Figure 1: This figure illustrates circuit partitioning with four domains and a single interface. The sparse structure of the original matrix A (top-half of the figure) was preserved in each of the subdomains A_i (bottom-half of the figure). The matrices E_i , F_i , and A_Γ are also circled for illustration in the bottom-half of the figure.

select interface nodes that naturally isolate the original system into subdomains.

3.2 Schur complement

To simplify notation, (4) can be rewritten as

$$A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \text{ where } A = \begin{pmatrix} A_D & E \\ F & A_\Gamma \end{pmatrix}. \quad (5)$$

Consider the linear system written in (5), in which A_D is assumed to be non-singular. From the first equation, the subdomain variables x can be expressed as

$$x = A_D^{-1}(f - Ey). \quad (6)$$

Substitution of this form into the second equation yields the following system of equations:

$$(A_\Gamma - FA_D^{-1}E)y = g - FA_D^{-1}f. \quad (7)$$

The matrix

$$A_\Gamma - FA_D^{-1}E \quad (8)$$

is called the *Schur complement* matrix associated with the interface variables y . Solving equation (6) constitutes the computational core of the DD technique. A consolidated three-step procedure to solve the original system is as follows. First form the Schur complement matrix $A_\Gamma - FA_D^{-1}E$ and the right hand side of (7). Next, solve equation (7) for the interface variables y . Finally, by substituting y in equation (6), obtain a solution for the subdomain variables x .

A high-level DD algorithm DOMAIN-DECOMPOSITION-SOLVER that takes as inputs A_D , A_Γ , E , F , f and returns the solution for the interface variables y and the subdomain variables x is as follows.

1. Solve $A_D P = E$ for P , i.e., obtain $P = A_D^{-1}E$
2. Form the Schur complement matrix $S = A_\Gamma - FP$
3. Solve $A_D q = f$ for q , i.e., obtain $q = A_D^{-1}f$
4. Calculate $g' = g - Fq$
5. Solve the equation $Sy = g'$ for interface variables y
6. Solve $x = q - Py$ for the subdomain variables x

There are three opportunities for parallelization in the above algorithm, all of which exploit the block diagonal structure of matrix A_D . They are in (i) step 1 where the equation $A_D P = E$ is solved, (ii) step 3 where the equation $A_D q = f$ is solved, and (iii) step 6 where the equation $x = q - Py$ is solved.

3.3 An example of two subdomains

An example based on two subdomains to illustrate the above algorithm is as follows. With two subdomains, equation (3) has the following form:

$$\begin{pmatrix} A_1 & 0 & E_1 \\ 0 & A_2 & E_2 \\ F_1 & F_2 & A_\Gamma \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ g \end{pmatrix} \quad (9)$$

where x_1 , x_2 , and y are the vectors of node voltage and inductor current in subdomain 1, subdomain 2, and the interface respectively. For this system (9), the Schur complement matrix is

$$S = A_\Gamma - F_1 A_1^{-1} E_1 - F_2 A_2^{-1} E_2. \quad (10)$$

The interface variable y can be obtained by solving the equation

$$Sy = g - F_1 A_1^{-1} b_1 - F_2 A_2^{-1} b_2. \quad (11)$$

Finally, x_1 and x_2 can be obtained by

$$x_1 = A_1^{-1} b_1 - A_1^{-1} E_1 y, \quad (12)$$

$$x_2 = A_2^{-1} b_2 - A_2^{-1} E_2 y. \quad (13)$$

Suppose that the inverse operator A_i^{-1} is implemented by LU factorization. For the example above, the LU factorization is performed for A_1 and A_2 . If A_i is half the size of the original matrix A , the total factorization cost for A_1 and A_2 is less than the factorization cost of A . Note that $A_1^{-1} E_1$ and $A_2^{-1} E_2$ are computed by solving the equations $A_i P_i = E_i$ for P_i , and that the E_i are sparse matrices. Once the Schur complement matrix S given by

$$S = A_\Gamma - F_1 P_1 - F_2 P_2 \quad (14)$$

is obtained, the equations $A_i q_i = b_i$ are solved for q_i . The q_i are easy to compute, since the LU factorizations for the A_i can be reused. The interface variables can be computed by solving $Sy = g - F_1 q_1 - F_2 q_2$. Here, S is a small matrix in comparison to A . Finally, the subdomain variables x_i are obtained by solving the equation

$$x_i = q_i - P_i y. \quad (15)$$

3.4 Complexity analysis

Without loss of generality, assume that a circuit with n nodes is partitioned into m equal sub-circuits. Then, each subdomain approximately has n/m nodes. Suppose that solving a linear system by direct or iterative methods has a complexity of $O(n^p)$, where n is the size of the linear system and $p \geq 1$. Using the proposed DD technique, the computational cost of solving a single sub-system is $O((n/m)^p)$. Multiplying by m , the complexity of the solution for

the entire system is $O((n^p)/(m^{p-1}))$. Note that this only occurs in the theoretically ideal case. For most practical cases, it is essential to factor in auxiliary computations in DD algorithms, such as addition, multiplication, the formation of the Schur complement matrix (which is much smaller than the subdomain matrices), and the cost for solving the interface. Based on this, it is reasonable to expect that a parallel version of the DD algorithm will achieve speed-up over solving the original system for large n .

4. NON-OVERLAPPING DOMAIN DECOMPOSITION

The above Schur complement approach requires a clear partition of original circuit. Another approach using an overlapping partition to build a preconditioner is described in this section.

4.1 Additive Schwarz preconditioner

In the additive Schwarz procedure, the original matrix A is partitioned into overlapping subdomains, instead of non-overlapping subdomains in the Schur complement approach. The overlapping regions from different subdomains play a role in data communication.

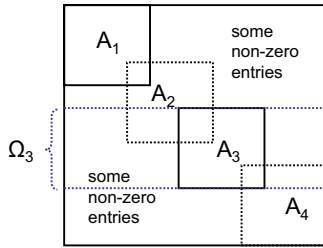


Figure 2: Overlapping matrix partition

Considering a linear system $Ax = b$, Fig. 2 is an example of its partition. A is partitioned into four blocks from A_1 to A_4 , and two regions with some non-zero entries. Each A_i is defined by $A_i = R_i A P_i$, where P_i is the operator that maps from subdomain Ω_i to Ω and R_i is the restriction operator which restricts the global vector to the vector on Ω_i . Subdomain Ω_i has the same rows as block A_i but the same columns as the whole matrix A .

The additive Schwarz preconditioner (P_{AS}^{-1}) of the linear system is defined by

$$P_{AS}^{-1} = \sum_{i=1}^m P_i A_i^{-1} R_i. \quad (16)$$

The term additive Schwarz simply refers to the fact that the components of the preconditioners are added together. Both direct and iterative methods can be used when applying A_i^{-1} to a vector.

This additive Schwarz preconditioner is able to accelerate the convergence of iterative methods by a block-like structured preconditioner. For instance, we compare DC simulations of a 40K power grid using three preconditioners, no preconditioner, a three-step Jacobi preconditioner, and a four-subdomain additive Schwarz preconditioner, in a GMRES algorithm. Here, we use a same stopping criteria $\|Ax - b\|_2 \leq 10^{-6} \|b\|_2$. GMRES converges after 4631 iterations (28.94 secs) without any preconditioner, 1550 iterations (14.82 secs) with the Jacobi preconditioner, and only 12 iterations (0.29 secs) with the additive Schwarz preconditioner.

4.2 Algebraic partition

The partition in the additive Schwarz procedure can be implemented on the matrix directly rather than on the power grid physically. Here, we assume that the rows of the sparse matrix A are distributed on a parallel machine. In the beginning, each row can only be assigned to one processor. A_i will be easily extracted on each processor from its own part in the matrix A , by dropping all off-processor columns. Once A_i is known, matrixes R_i and P_i are implicit. In this non-overlapping case, there will be no communication among processors.

Next, by enlarging the set of rows on each processor, we have a partition with wider overlapping. It means that any processor can share common rows with any other processors. Such overlapping requires more data communication among processors. Information recording the allocation of rows in i th processor is the matrix P_i . we use $R_i = P_i^T$ to extract block A_i such that $A_i = R_i A P_i$. Fig. 3 is an example to show the matrices P_1 and R_1 in extracting block A_1 from matrix A . We associate one subdomain with one processor to achieve parallelism.

$$\begin{array}{c}
 A_1 \\
 \begin{array}{|cccc|cc}
 \hline
 1 & 2 & 0 & 0 & 2 & 0 \\
 3 & 4 & 5 & 0 & 0 & 0 \\
 6 & 0 & 7 & 4 & 3 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 \\
 \hline
 0 & 4 & 3 & 0 & 5 & 0 \\
 0 & 0 & 0 & 0 & 2 & 6 \\
 \hline
 \end{array} \\
 A_2
 \end{array}
 \quad
 \begin{array}{c}
 P_1 \\
 \begin{array}{|ccc}
 \hline
 1 & 0 & 0 \\
 0 & 1 & 0 \\
 0 & 0 & 1 \\
 0 & 0 & 0 \\
 0 & 0 & 0 \\
 0 & 0 & 0 \\
 \hline
 \end{array} \\
 R_1 = P_1^T
 \end{array}$$

Figure 3: The matrix A is divided into two subdomains, subdomain Ω_1 from row1 to row 4 and subdomain Ω_2 from row3 to row6.

4.3 Parallelism strategy

The construction of the preconditioner can be well parallelized by calculating the LU factorization of A_i on each processor. For the solution of the preconditioned linear system

$$P_{AS}^{-1} Ax = P_{AS}^{-1} b. \quad (17)$$

the main workload in every iteration of any iterative method (like GMRES) is to evaluate $z = P_{AS}^{-1} Ax$. The evaluation contains two steps (1) $y = Ax$ and (2) $z = P_{AS}^{-1} y$. In the first step $y = Ax$, since A is already distributed along rows, each processor computes only its local part in Ax . The global vector y is formed by collecting all local vectors from all processors and then is broadcasted back to all processors. In the second step $z = P_{AS}^{-1} y$, each processor already knows the global vector y . Note that $P_{AS}^{-1} = \sum_{i=1}^m P_i A_i^{-1} R_i$. $z = P_{AS}^{-1} y$ is available by computing $z_i = P_i A_i^{-1} R_i y$ on each processor and adding all z_i 's together.

5. NUMERICAL EXPERIMENTS

The proposed domain decomposition methods have been implemented and integrated into a linear simulator written in C++. Mesh networks were used to model the upper-two global power grids, which consist of RLC wires, voltage sources, current sources, and decoupling capacitors. Fig. 4 shows a part of a grid model used in our experiments.

5.1 Simulation environment

A cluster machine with 4 nodes. Each node has

- CPU: AMD Opteron 146 (2.0 GHz)
- Memory: 4.0 GB

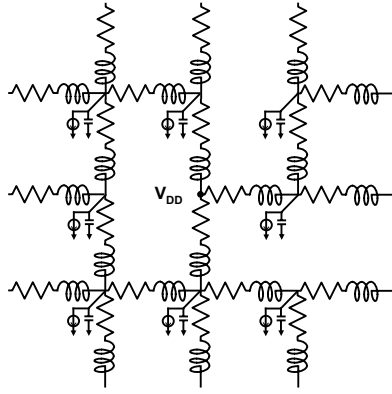


Figure 4: This figure illustrates a small portion of the power grids used for the simulations. The matrix A lacks the sparse symmetric positive definite structure due to the inductance. Hence, the conjugate gradient algorithm is not used in the simulations.

- OS: Red Hat Enterprise Linux 4.0, x86_64 (2.6.9-11 kernel)
- Compilers: GCC (g77, gcc and g++) 3.4.3
- Message Passing: MPICH 1.2.6
- Packages: Trilinos 6.0.15 (compiled without MPI), SuperLU 3.0

5.2 Implementation

The simulation results for 4 circuits, A with 40K nodes, B with 1M nodes, C with 4M nodes, and D with 10M nodes are presented here. Circuits A, B, and C were partitioned into 4 domains. Circuit D was partitioned into 8 domains.

Tables 1 and 2 report the average CPU time in seconds for DC and 20-step transient simulations of the four circuits respectively. Note that “—” means that the algorithm either ran out of memory or that it could not finish in a reasonable time allowed for completion. “*” means the results are not available because we did not have the necessary 8 processors. The different power grid simulation frameworks that were implemented (or used) are summarized below.

- SPICE: a general circuit simulator
- LU: computing LU factorization, solving the linear system k times by forward and backward substitutions;
- IF: generating incomplete LU factorization preconditioner (IF) with drop tolerance 0.001, solving the linear system k times by GMRES method with IF preconditioner. For a symmetric positive definite structure, incomplete Cholesky factorization may perform better, but our system does not have such structure;
- MG: generating five-level multi-grid preconditioner (MG) with direct solver on coarse grid and Gauss-Seidel prior/post smoothers, solving the linear system k times by GMRES method with MG preconditioner;
- DD+LU: DD method solving the linear system k times with direct solver (LU) for subdomain problems;
- DD+IF: DD method solving the linear system k times with iterative solver (IF preconditioner) for subdomain problems;
- DD+MG: DD method solving the linear system k times with iterative solver (MG preconditioner) for subdomain problems;

- AS: iterative GMRES solver with additive Schwarz preconditioner. LU factorization is used to solve the subdomain problems.

Note that the Trilinos [5] package we used for computation has already been well optimized for large-scale sparse computing. The direct solver that was used, SuperLU [3], for LU factorization is also one of the best available packages. Our DD-based approaches have shown that existing algorithms can be extended to large-scale power grid analysis.

Table 1: Runtimes for DC simulation (secs)

Circuit	A	B	C	D
No. nodes	40 K	1 M	4 M	10 M
No. subdomains	4	4	4	8
SPICE	250.00	—	—	—
LU	0.81	124.85	—	—
IF	1.29	310.72	—	—
MG	0.56	20.44	—	—
DD+LU(serial)	0.44	56.30	—	—
DD+IF(serial)	46.41	1338.06	—	—
DD+MG(serial)	3.16	143.69	—	—
DD+LU(MPI)	0.16	14.32	157.20	449.32
DD+IF(MPI)	14.45	402.07	4340.94	—
DD+MG(MPI)	1.23	39.82	178.43	653.17
AS(MPI)	0.29	27.11	234.73	*

Table 2: Runtimes for transient simulation with 20 time steps (secs)

Circuit	A	B	C	D
No. nodes	40 K	1 M	4 M	10 M
No. subdomains	4	4	4	8
SPICE	563.48	—	—	—
LU	3.67	230.22	—	—
IF	17.53	3061.75	—	—
MG	8.45	339.89	—	—
DD+LU(MPI)	1.14	18.96	183.80	517.21
DD+IF(MPI)	31.56	873.84	13297.53	—
DD+MG(MPI)	2.58	85.69	376.11	1377.16
AS(MPI)	3.12	308.74	1543.72	*

5.3 DC simulations

The computational cost of DC simulation consists of pre-computing and one step substitution. The pre-computing means LU factorization (for LU), generating preconditioner (for MG, IF, and AS), or forming Schur complement matrix and its factorization (for DD+LU, DD+MG and DD+IF). For single processor machines, if sufficient memory is available, the iterative solver with MG preconditioner is faster than all the methods except DD+LU(MPI). This is because the computational cost of forming a dense Schur complement matrix S defined by Eqn. 8 may surpass the gain from solving subdomains. However, the memory requirement of MG limits its applying to large-scale problems directly. Note that MG is a GMRES

Table 3: Runtimes analysis with various interface nodes for DD+LU, 4M size (secs)

No. interface nodes	40	100	200	400	800	1000	2000	4000
$LU(A_i)$	123.40	26.10	127	134.57	131.69	129.73	131.48	133.78
$F_i A_i^{-1} E_i$	5.73	14.20	29.15	61.74	123.53	147.64	307.89	651.40
$LU(S)$	0.01	0.01	0.01	0.01	0.10	0.19	1.45	11.01
One substitution	1.25	1.47	1.53	1.78	1.89	1.96	2.13	2.36
Total	130.39	141.78	157.69	198.10	257.21	279.52	442.95	798.54

iterative solver with a MG preconditioner. Total memory requirement contains the storages of multigrid preconditioner, GMRES, and a sparse matrix A . For instance, solving a 4M linear system, a GMRES algorithm with restarting every 60 inner iterations needs at least $4\text{MB} \times 60 \times 8 \approx 1.92\text{GB}$ memory. The storage for the sparse matrix A is around 1GB.

Our solution to overcome this difficulty is parallel domain decomposition. Without parallelism, a single processor machine has to take the memory requirements from all subdomains. This may exceed the maximum available memory of the processor. Consequently, we still cannot solve problems like 4M or larger in a single processor machine. Parallel domain decomposition distributes memory requirement to each processor. Each processor only associates with one subdomain. This divide-and-conquer technique reduces the computational cost for each processor. Here, DD+LU, DD+MG, or AS are all good candidates on cluster machines.

5.4 Transient simulations

As shown in Table 2, DD+LU is the best algorithm for transient simulation. The computational cost of transient simulation consists of pre-computing and 20 step substitutions for LU-based approaches. The cost is roughly 20 step iterations for iterative methods after being deducted the cost of the reusable preconditioner. Under the domain decomposition framework, LU factorization can be used both in the Schur complement and in solving subdomains. Furthermore, solving subdomains in DD+LU has been simplified into back substitutions after LU factorization. Hence, DD+LU achieves the best performance in the transient simulation.

5.5 Effect of the size of interface

As the number of interface nodes increases, DD+LU becomes less efficient. This is because that it spends more time on forming the Schur complement. We use a 4M grid as an example and profile the cost of each computational step in DD+LU. The results are listed in Table 3. We change the interface nodes from 40 to a maximum of 4000 in the first row. The second row gives the time for LU factorization of each subdomain; the third row is the time to form the Schur complement matrix S ; the fourth row indicates the cost of one substitution; the last row reports the total cost for the DC simulation. It is evident from the table that forming the Schur complement matrix S is the dominance cost factor. This cost linearly scales with the number of interface nodes. Combining Tables 1 and 3, we find that AS will outperform DD+LU for the DC simulation when interface nodes are greater than 800. AS is an iterative solver and does not rely on the size of interface nodes. So it is suitable for the DC simulation of densely connected grids.

6. CONCLUSION

This paper described an efficient parallel domain decomposition framework for large-scale power grid simulation. Simulation results indicate that this approach can be used to extend existing lin-

ear circuit simulators to handle otherwise intractable systems. The numerical results successfully demonstrate our parallel domain decomposition approach.

7. REFERENCES

- [1] T. Chen and C. C. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," *Proc. Design Automation Conference*, pp. 559–562, 2001.
- [2] E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells," *Proc. Intl. Conference on Computer-aided Design*, pp. 485–488, 2004.
- [3] Sherry Li, Jim Demmel, John Gilbert, "SuperLU user's guide", <http://crd.lbl.gov/xiaoye/SuperLU/>.
- [4] P. Feldmann and F. Liu, "Sparse and efficient reduced order modeling of linear subcircuits with large number of terminals," *Proc. Intl. Conference on Computer-aided Design*, pp. 88–92, 2004.
- [5] M. A. Heroux and J. M. Willenbring, "Trilinos users guide", Sandia National Laboratories, SAND2003-2952, 2003, <http://software.sandia.gov/trilinos/>.
- [6] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. Computer-aided Design*, vol. 21, pp. 1148–1160, Oct. 2002.
- [7] Y.-M. Lee, Y. Cao, T. H. Chen, J. Meiling, and C.C. Chen, "HiPRIME: Hierarchical and passivity preserved interconnect macromodeling engine for RLKC power delivery," *IEEE Trans. Computer-aided Design*, vol. 24, pp. 797–806, Jun. 2005.
- [8] P. Li, "Power grid simulation via efficient sampling-based sensitivity analysis and hierarchical symbolic relaxation," *Proc. Design Automation Conference*, pp. 664–669, 2005.
- [9] S. Pant and E. Chiprout, "Power grid physics and implications for CAD," *Proc. Design Automation Conference*, pp. 119–204, 2006.
- [10] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic and system simulation methods*, McGraw Hill, New York, 1994.
- [11] H. Qian, S. R. Nassif, and S. S. Sapatnekar, "Power grid analysis using random walks," *IEEE Trans. Computer-aided Design*, vol. 24, pp. 1204–1224, Aug. 2005.
- [12] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
- [13] B. Smith, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 2004.
- [14] H. Su, E. Acar, and S. R. Nassif, "Power grid reduction based on algebraic multigrid principles," *Proc. of Design Automation Conference*, pp. 109–112, 2003.

- [15] H. Su, K. H. Gala, and S. S. Sapatnekar, "Analysis and optimization of structured power/ground networks," *IEEE Trans. Computer-aided Design*, vol. 22, pp. 1533–1544, Nov. 2003.
- [16] H. Yu, Y. Shi, and L. He, "Fast analysis of structured power grid by triangularization based structure preserving model order reduction," *Proc. Design Automation Conference*, pp. 205–210, 2006.
- [17] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Trans. Computer-aided Design*, vol. 21, pp. 159–168, Feb. 2002.
- [18] Z. Zhu, B. Yao, and C.-K. Cheng, "Power network analysis using an adaptive algebraic multigrid approach", *Proc. Design Automation Conference*, pp. 105–108, 2003.