PowerRush* : Efficient Transient Simulation for Power Grid Analysis†

[Invited Special Session Paper]‡

Jianlei Yang Zuowei Li Yici Cai Qiang Zhou
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University, 100084, Beijing, China
yjl09@mails.tsinghua.edu.cn lizuoweirain@gmail.com
caiyc@mail.tsinghua.edu.cn zhouqiang@tsinghua.edu.cn

ABSTRACT

Transient analysis is the most practical and effective approach for power grid validation, but which is very challengeable for large scale VLSI chips because it is really time consuming and requires large memory resources. In this paper we proposed a parallel transient simulation approach for efficient power grid analysis. Firstly we adopt symmetric formulation for NA equation of RLC power grid to reduce memory usage. Meanwhile, fast Cholesky factorization solver can be used to improve simulation efficiency. Secondly, we perform partition-based parallel transient simulation for naturally independent subnets without accuracy lost. Thirdly, we propose a composite simulation flow for efficient and practical transient analysis for industrial power grid. Finally, several industrial power grid benchmarks are evaluated on our approaches for high accurate transient simulation with extremely low memory consumption.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Simulation*

General Terms

Algorithms, Design, Performance, Verification

Keywords

PowerRush, Power Grid, Circuit Simulation, Transient Analysis

*__PowerRush__ is denoted as __PowerRush__™ for TradeMark declaration.

†This work is supported in part by the National Natural Science Foundation of China (NSFC) No.60976035 and No.61274031.

‡This paper is invited by Special Session of *2012 TAU Power Grid Simulation Contest* in ICCAD 2012.

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

1. INTRODUCTION

The performance and reliability of power distribution network is degraded due to the higher device densities and faster switching frequencies which cause large switching currents to flow on it [1]. Aiming to improve the robustness of power grid network, designers usually need to perform both static analysis (DC) and transient analysis. However, it is really challengeable to analyze power grid with tens or even hundreds of million nodes which has became the most computation consuming tasks in design verification flow. Many research efforts have been made to perform efficient simulation in terms of accuracy, runtime as well as memory usage.

Static analysis is required to obtain the static IR-drop or steady state responses corresponding to the IR-drop caused by the average current flowing through a power supply network. Due to the symmetric positive and definite (SPD) conductance matrices, static analysis can be performed by many efficient methods which includes Krylov-subspace method [2], Hierarchical and Macro-modeling methods [3], Random Walk [4], Multigrid methods [5][6], and etc.

Due to the growth in power consumption and switching speeds of modern high performance chips, the Ldi/dt effects are becoming a growing concern so that transient analysis is required for determining the dynamic IR-drop and Ldi/dt noise. But the MNA equation of RLC power grid is not still SPD because addition current variables are introduced by inductors. Many efficient solvers for SPD systems were not still available [7]. General solution techniques such as LU factorization can be used for the non-SPD system, however, the large size of the network makes such general solution infeasible. Partition based macro-modeling approach is proposed in [7] to reformulate the original non-SPD system as a SPD system which can be solved by Cholesky factorization. But it requires too much more additional computation to practical use. Aiming to overcome this difficulty a new approach is proposed to reformulate the MNA equation as a symmetric NA equation by eliminating the additional current variables [2]. In this paper, PowerRush adopts the above reformulation to be solved by Cholesky factorization related methods which can significantly improves the factorization efficiency and reduce the memory consumption since the Cholesky factorization takes only half of multiplications and memory references than LU factorization.

Another challengeable topic in transient simulation is that it requires performing each simulation on each time point so

that it can be very time consuming for a long simulation period. Some parallel approaches have been proposed to speedup transient simulation, some of which mainly focus on performing fine-grained parallelization to accelerate several kinds of linear solvers [8][9]. But all of them require fine-grained parallel programming, so it needs high implementation and debugging effort which may limit its practical use for large scale power grid analysis. Another kind of approaches is focused on partitioning power grid to subgrids and then simulating them in parallel which can absorb some heuristic information, such as pad location in Flip-Chip package based on local effect, to partition power grid by the weakest coupling direction [10][11]. But all of them require to smooth errors existed on boundary nodes and thus its performance can be affected by the original power grid topologies. In our simulator, naturally separated subnets based coarse-grained application level parallel approach is adopted to analyze each subnet individually without any accuracy lost because there is no electrical coupling between each separated subnet.

Meanwhile, there are already many research contributions to linear solvers for power grid analysis but we believe that detailed implementation on transient simulation flow is also important for industrial practical use. In this paper, PowerRush is developed as a completely practical simulation flow and firstly presented in research area. The authors hope that these works would be helpful for power grid simulation community.

In this paper, we propose an efficient parallel transient simulation approach. The contributions of our work are: **(i)** Symmetric formulation of NA equation for RLC power grid is adopted to reduce memory usage and improve matrix factorization efficiency; **(ii)** Naturally separated nets are identified to solve independently without accuracy lost for parallel simulation; **(iii)** Composite simulation flow are proposed for efficient and practical transient analysis.

The reminder of this paper is organized as follows. Section 2 gives a brief background of transient power grid analysis. Following that, detailed transient simulation approach for RLC power grid is presented in Section 3. Experimental results are provided and numerically illustrated in Section 4. Concluding remarks are given in Section 5.

2. TRANSIENT SIMULATION BACKGROUND

2.1 Power Grid with RLC Model

For modern power grid designs, RLC model is required for transient simulation but only package inductance is considered because on-chip inductance effect is relative too small than off-chip inductance effect [7]. The typical equivalent circuit of detailed RLC model from IBM power grid [12] is shown in Figure 1. On-chip power/ground wires are modeled in lumped resistance and on-chip inductance is traditionally ignored, because it is smaller than package inductance and analyzing power network with on-chip inductance requires huge computational cost or even introducing ill-conditioned problems. Chip cells are simulated and modeled as equivalent circuits with current source, resistance and capacitance. Power IOs, which supply current from package to die, are often modeled in series of an inductance and a resistance.

Firstly we consider the pad model and current loading model by taking *ibmpg1t.spice* as an example [12]. As shown in Figure 2(a), a pad model includes a series of an ideal voltage source Vdd, a lumped self-inductance L and a lumped resistance r. For convenience, we transform it as a Norton equivalent model which is shown in Figure 2(b). The pad resistance r is divided into two smaller resistances whose resistance value is half of r and their conductance is twice of it. Then pad inductance is located between the two divided resistances. So the serial of ideal voltage source Vdd and its neighbored divided resistance $r/2$ can be transformed as a current source $2g$Vdd and a conductance $2g$ in parallel. It's easy to know that the response of transformed circuit is same as the original circuit. Thus, all of voltage sources have been transformed as current sources, which can be directly stamped into MNA equations.

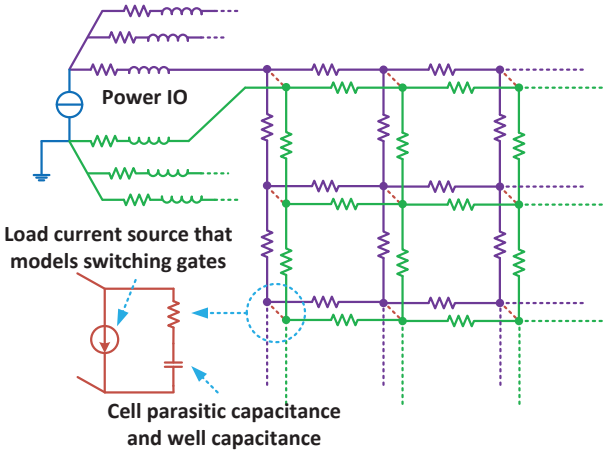
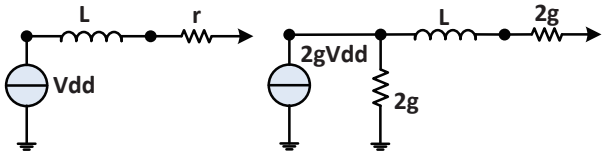


Figure 1: Power Grid with RLC Model.



(a) In Serial (b) Norton Equivalent Circuit

Figure 2: Pad Model in IBM Power Grid.

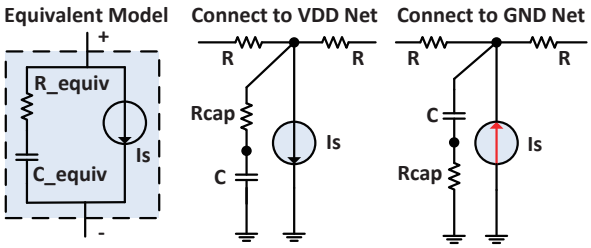


Figure 3: Current Source Loading Model in IBM Power Grid.

And for current source loading, it has been simulated and modeled as an equivalent circuit which is shown in Figure 3. A serial of a resistance R_{equiv} and a capacitance C_{equiv} is connected with a current source I_s in parallel as a companion model. Current source loading is connected from VDD net node to GND net node but there is a little different between them. As shown in Figure 3, the sequence of resistance

and capacitance is different between for VDD net and for GND net. But it's also easy to figure out that this will not influence their circuit responses as long as the current source can be used properly. That is to say, we just only need to distinguish their current directions as shown in Figure 3.

2.2 Transient Simulation Equation

Transient simulation involves computing waveforms of power grid as a function of time. For power grid with RLC model, transient simulation can be performed by formulating it as a system of differential equations. The practical approach for solving differential equations is to repeatedly discretize them and solve the resulting differential algebraic equations (DAEs) [13]. The challenge in transient simulation is to move time forward by integrating these terms to get sets of equations at each time step that we have to solve.

By using MNA method which includes KCL and KVL laws, and considering that only the branch currents running through inductors have to be kept in the equations, a n-nodes power grid with RLC model can be formulated as following [2]:

$$\begin{pmatrix} G & A_l^T \\ -A_l^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_n \\ \mathbf{i}_l \end{pmatrix} + \begin{pmatrix} C & 0 \\ 0 & \mathcal{L} \end{pmatrix} \begin{pmatrix} \frac{d\mathbf{v}_n}{dt} \\ \frac{d\mathbf{i}_l}{dt} \end{pmatrix} = \begin{pmatrix} A_i^T \mathbf{i}_i \\ 0 \end{pmatrix} \tag{1}$$

where $\mathbf{v}_n$ is the unknown vector of node voltages, $\mathbf{i}_l$ is the vector of branch currents running through inductors, $\mathbf{i}_i$ is the vector of current source loadings, and $G = A_g^T \mathcal{G} A_g$, $C = A_c^T \mathcal{C} A_c$, $L = A_l^T \mathcal{L} A_l$, where $\mathcal{G}$ is diagonal matrices whose diagonal elements are conductance value for each resistor, $\mathcal{C}$ is diagonal matrices whose diagonal elements are capacitance value for each capacitor, $\mathcal{L}$ is diagonal matrices whose diagonal elements are inductance value for each inductor, A_i, A_g, A_c, A_l are adjacent matrices to present the connectivity of the power grid circuit, and obviously each row of the adjacent matrix only contains two non-zero elements (only one if the branch connects to the ground). The subscripts i, g, c and l stand for branches which contain independent current sources, resistors, capacitors, and inductors, respectively.

3. TRANSIENT SIMULATION APPROACH

3.1 Symmetric Representation

As demonstrated in equation (1), although MNA provides a good solution for general circuits, the introduction of additional current variables makes the system matrix non-positive definite, which is crucially necessary for the efficiency and fast convergence of both iterative and direct methods since the Cholesky factorization takes only half of multiplications and memory references than the LU factorization. However, it was believed that it is infeasible to use the NA method for general RLC circuits due to the need of additional current variables. And by eliminating the current variables, we can obtain an NA formulation for RLC circuits, which is a SPD system. Hence the Cholesky factorization and the conjugate gradient method can be still used for efficient simulation [2].

Noticed that the above transient simulation equations include dynamic components, usually we transform them as differential algebraic equations (DAEs) by using a finite difference approximation. The most common used methods for solving DAEs in circuit simulation are Backward Euler differential approximation method and Trapezoidal differential approximation method (TR). Due to the difference of efficiency and accuracy between them, both of their formulations are discussed below.

3.1.1 Backward Euler Method

By using backward Euler method, the circuit equation (1) can be reformulated at time point $(t+h)$ with differential step size h as following differential algebraic equation:

$$\begin{pmatrix} G & A_l^T \\ -A_l & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}_n(t+h) \\ \mathbf{i}_l(t+h) \end{pmatrix} + \begin{pmatrix} C & 0 \\ 0 & \mathcal{L} \end{pmatrix} \begin{pmatrix} \frac{\mathbf{v}_n(t+h) - \mathbf{v}_n(t)}{h} \\ \frac{\mathbf{i}_l(t+h) - \mathbf{i}_l(t)}{h} \end{pmatrix} = \begin{pmatrix} A_i^T \mathbf{i}_i(t+h) \\ 0 \end{pmatrix}$$

by moving the voltage variable $\mathbf{v}_n(t)$ and current variable $\mathbf{i}_l(t)$ at time point t to the right hand side, we can obtain

$$\begin{pmatrix} G + \frac{C}{h} & A_l^T \\ -A_l & \frac{\mathcal{L}}{h} \end{pmatrix} \begin{pmatrix} \mathbf{v}_n(t+h) \\ \mathbf{i}_l(t+h) \end{pmatrix} = \begin{pmatrix} \frac{C}{h}\mathbf{v}_n(t) + A_i^T \mathbf{i}_i(t+h) \\ \frac{\mathcal{L}}{h}\mathbf{i}_l(t) \end{pmatrix}$$

by eliminating current variables $\mathbf{i}_l(t+h)$ we can obtain

$$\left(G + \frac{C}{h} + A_l^T \frac{h}{\mathcal{L}} A_l\right) \mathbf{v}_n(t+h) = \frac{C}{h}\mathbf{v}_n(t) + A_i^T \mathbf{i}_i(t+h) - A_l^T \mathbf{i}_l(t)$$

and remembering that $L = A_l^T \mathcal{L} A_l$ we can obtain that

$$\left(G + \frac{C}{h} + \frac{h}{L}\right) \mathbf{v}_n(t+h) = \frac{C}{h}\mathbf{v}_n(t) + A_i^T \mathbf{i}_i(t+h) - A_l^T \mathbf{i}_l(t)$$

3.1.2 Trapezoidal Method

For Trapezoidal case, the equation (1) also can be reformulated as differential algebraic equation [2] and finally we can obtain

$$\left(G + \frac{2}{h}C + \frac{h}{2L}\right) \mathbf{v}_n(t+h) = \left(-G + \frac{2}{h}C - \frac{h}{2L}\right) \mathbf{v}_n(t) + A_i^T \left(\mathbf{i}_i(t+h) + \mathbf{i}_i(t)\right) - 2A_l^T \mathbf{i}_l(t) \tag{2}$$

The above derived linear equation has been proved to be a SPD system and also a diagonally dominant system [2]. After obtained the voltage vector $\mathbf{v}_n(t)$ we can compute the solution $\mathbf{v}_n(t+h)$ of next time point as long as the right hand side has been constructed. We give a more detailed discussion about the three terms of right hand side in equation (2).

The first one is $\left(-G + \frac{2}{h}C - \frac{h}{2L}\right)\mathbf{v}_n(t)$ which can be obtained from the voltage solution of previous time point, but there is no need to construct $\left(-G + \frac{2}{h}C - \frac{h}{2L}\right)$ explicitly. Because this expression can be reformulated as following

$$\left(-G + \frac{2}{h}C - \frac{h}{2L}\right) \mathbf{v}_n(t) = -\left(G + \frac{2}{h}C + \frac{h}{2L}\right) \mathbf{v}_n(t) + \frac{4}{h}C\mathbf{v}_n(t)$$

Noticed that the expression $\left(G + \frac{2}{h}C + \frac{h}{2L}\right)\mathbf{v}_n(t)$ just is the right hand side of the previous time point, we can store it for reuse.

The second term $A_i^T\left(\mathbf{i}_i(t+h) + \mathbf{i}_i(t)\right)$ is the sum of current sources for time point $(t+h)$ and t which is easy to compute. The third term $2A_l^T\mathbf{i}_l(t)$ is the current flowing through the pad branch on previous time point t, which can be directly obtained according to the Norton equivalent circuit as shown in Figure 2(b).

And for Backward Euler method, the meanings of the three items on right hand side are same as Trapezoidal case. Thus, we have obtained the NA formulation for RLC power grid which is a SPD system. That's to say, we have reformulated the non-symmetric MNA equation as a symmetric NA equation which can be solved by Cholesky factorization and many other efficient symmetric sparse solvers, such as AMGPCG [6] which will be discussed in Section 3.2.

3.1.3 Accuracy Analysis

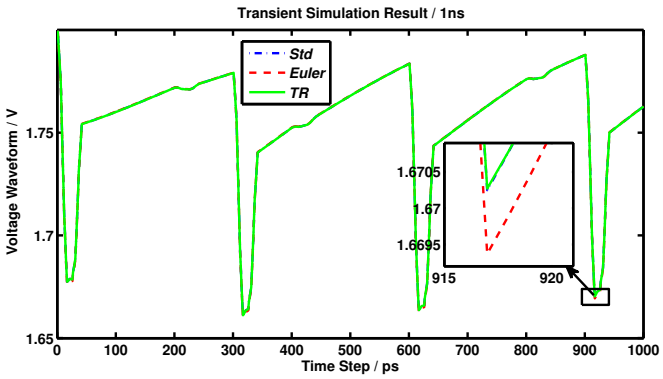
The Euler method is a first-order method, which means that the local error (error per step) is proportional to the square of the step size $O\left(h^2\right)$, and the global error (error at a given time) is proportional to the step size. It also suffers from stability problems. For these reasons, Euler method is not often used in practice but just serves as the basis to construct more complicated methods.

The Trapezoidal method is an implicit second-order method, which is probably the most often used solution method in circuit simulation. TR can be viewed as a linear multistep method and an implicit method with $O\left(h^3\right)$ error and excellent stability which has better accuracy than Euler method.

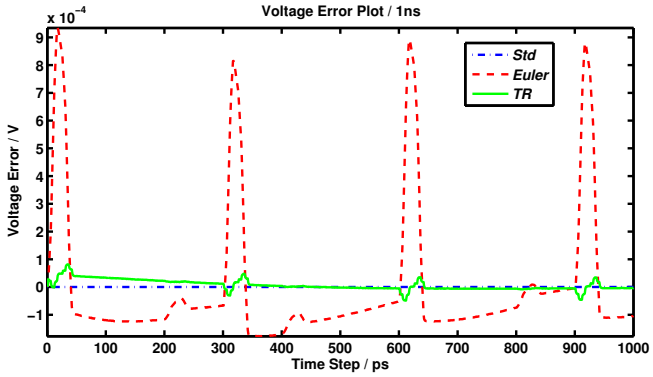
Table 1: Accuracy Comparison of Euler Method and TR Method. E_{max} is max error and E_{avg} is average error compared with standard solution.

Grid	Euler Method		TR Method		Improve	
	E_{max}	E_{avg}	E_{max}	E_{avg}	E_{max}	E_{avg}
ibmpg1t	1E-03	1E-04	8E-05	7E-06	12X	20X
ibmpg2t	8E-04	1E-04	8E-05	9E-06	10X	16X
ibmpg3t	8E-04	1E-04	7E-05	5E-06	12X	23X
ibmpg4t	2E-03	1E-04	1E-04	9E-06	16X	18X
ibmpg5t	5E-04	9E-05	3E-05	3E-06	16X	23X
ibmpg6t	6E-04	1E-04	5E-05	5E-06	10X	21X

We can use the Euler method to get a fairly good estimation for the solution, which can be adopted as the initial guess of differential equations. The voltage waveform of node *n1_18333_5432* in *ibmpg1.spice* is shown in Figure 4, where the step size $h = 1ps$ and total simulation period $T = 1ns$. For clarity, the detailed accuracy comparison between Euler method and TR method is shown in small window of Figure 4(a) and voltage error is plotted in Figure 4(b). The accuracy comparison between Euler method and TR method is numerically illustrated in Table 1, from which we can see that TR method can improve the solution accuracy more than 10X compared with Euler method. This result just exactly validates the theoretical analysis that TR method has one higher order accuracy than Euler method. It is worth notice that Euler method is adopted in PowerRush of contest submitted version which cannot meet the accuracy requirements, but finally TR method is properly introduced into PowerRush to improve simulation accuracy.



(a) Voltage Waveform



(b) Voltage Error Plot

Figure 4: Simulation Result of Node *n1_18333_5432* in *ibmpg1t.spice*. Std means standard solution of golden simulator.

3.2 Transient Simulation Flow

The flow chart shown in Figure 5 is useful to visualize the overall transient simulation flow inside our transient power grid simulator PowerRush. The SPICE netlist is parsed into build-in data structure and then the circuit builder identifies the existed separated subnets to simulate them in parallel. Then the DC operation point is computed for static analysis and as preparation for transient simulation. On transient simulation stage, the simulator repeatedly applies time discretization for circuits, builds NA equation and RHS, and then passes them into solver to obtain solutions.

Once the power grid has been solved at time point $(t+h)$, we then need to compute voltage solution for every capacitor and current for every inductor. Since we hope to avoid constructing each term of RHS in equation (2) explicitly, these are required at the next time point, in order to update the model parameter values as shown in their companion models. Obviously the voltage value of every capacitor can be directly obtained, as simply the difference of its two node voltages. And current flowing through every inductor, we can compute it by using its Norton equivalent circuit as shown in Figure 2(b). This requires knowledge of pad currents $\mathbf{i}_l$, which should have been stored at the previous time point, and therefore at all previous time points, going back to the beginning of simulation time $t = 0$, where $\mathbf{i}_l\,|_{t=0}$ is

required. This generates a requirement that, as part of the DC analysis is run at $t = 0$, so we should find and store the current for every inductor. As we have known that, DC analysis is adopted as an initialization for transient analysis. Remembering that the immediate method of performing DC analysis involves stripping the dynamic components of RLC circuit, by disabling all $\mathcal{L}$ and $\mathcal{C}$ components. Disabling dynamic components means that every capacitor is replaced by an open circuit which will lead to no connection between the two neighbored nodes, and so the voltage across the open circuit can be easily found from the DC analysis results. It also means that every inductor is replaced by a short circuit which will lead the two neighbored nodes to be merged as an equivalent node, typically represented by a $0V$ voltage source, whose current is also easily available from the NA solution vector. Thus, DC analysis can provide the initial state of all dynamic components: currents flowing through inductors and voltages across capacitors.

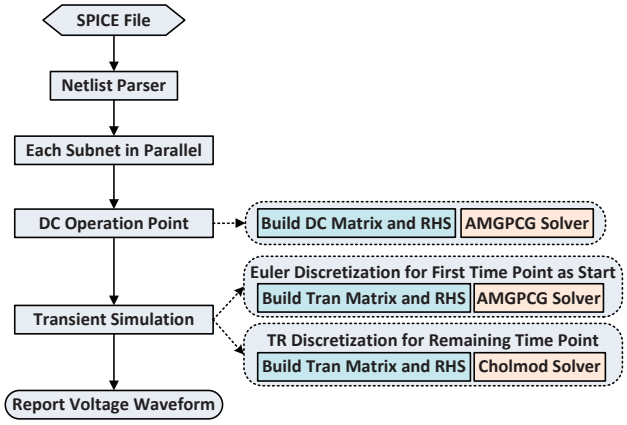


Figure 5: Overall Transient Simulation Flow.

As demonstrated in Section 3.1.3, TR discretization method is more accurate than Euler method, and the accuracy of Euler method is not enough for practical power grid simulation. In our transient simulator, TR discretization method is mainly used and combined with Euler method for the first time point as a startup. The detailed description of simulation steps is shown in below:

(1) Netlist parser for power grid with SPICE format;
(2) Circuit builder identifies separated subnets, then simulate each subnet in parallel;
(3) Static analysis at $t = 0$ to obtain the DC operation point:
  - Build DC matrix and RHS;
  - AMGPCG solver for voltage solutions;
(4) Transient simulation at first time point by using Euler method:
  - Build Tran matrix and RHS;
  - AMGPCG solver for voltage solutions;
(5) Transient simulation for remaining time points by using TR method:
  - Rebuild Tran matrix and RHS;
  - Cholmod solver for voltage solutions at each time point.

Noticed that in simulation procedure, there are three matrices to build because of considering different circuit models or using different discretization methods. As we have known that [6], AMGPCG is faster than Cholmod when performing simulation just once because that Cholmod solver needs much more runtime for Cholesky factorization but AMGPCG has a very efficient Multigrid setup step and a very robust convergence. So AMGPCG solver is called when performing DC analysis and the transient simulation at first time point as startup, while Cholmod solver is called when performing the remaining transient simulation steps.

Since Euler discretization method just needs the circuit status of previous one time point, while TR method needs the circuit status of previous two time points, both Euler discretization method and TR discretization method are used in our simulator for practical transient simulation. As shown in Figure 6, the first 4 simulation steps are performed as following:

(1) DC analysis is performed at the simulation starting time point $t0$;
(2) The first time point of transient simulation is performed at time point $t1$ by using Euler method according to the solutions of previous time point $t0$;
(3) The second time point of transient simulation is performed at time point $t2$ by using TR method according to the solutions of previous time point $t0$ and $t1$;
(4) The third time point of transient simulation is performed at time point $t3$ by using TR method according to the solutions of previous time point $t1$ and $t2$;

Later, the TR method is used for simulation at remaining time points until it arrives at the time point for ending.

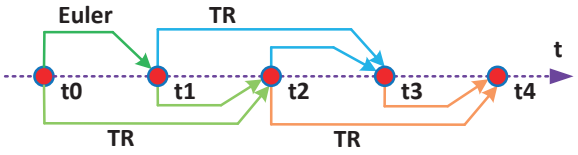


Figure 6: Data Dependency in Transient Simulation.

3.3 Nets Level Parallelization (Coarse-Grained)

The existed circuit/topology level parallelization can absorb heuristic information to partition power grid to accelerate transient simulation. But all of them require to smooth boundary errors and thus its performance can be affected by practical power grid. Also for the existed fine-grained parallel simulation approaches, it is possible to parallelize the key steps in an existing sparse linear solver. However, the solver/algorithm level parallelization requires fine-grained parallel programming, hence high implementation and debugging effort which is infeasible for practical use.

In our simulator, we adopt a coarse-grained application level parallel approach based on power grid partitioning. Taking IBM power grid as an example, usually there exists only single GND net while several VDD nets exist without any electrical connection. That is to say, there is no electrical coupling between each separated subnet so that they can be analyzed individually without any accuracy lost. As shown in Figure 7, usually GND net only includes subnet #0 while VDD net includes three subnets: subnet #1, subnet #2 and subnet #3. All of those naturally separated subnets are identified by our circuit builder and then pass them to multi-threads for parallel solving. Considering that if each of them is solved by each thread it will require much more memory resources because the peak memory is the sum

of each memory required by each thread. So we may take a multi-threads scheduling strategy to optimize the peak memory usage. Noticed that GND net is usually only includes one single net, and its total number of nodes is similar to the sum of all VDD subnets, so we put the GND net into a thread and put all VDD subnets into another thread. If the size of subnets is not as the above case, we can firstly reorder them to obtain an optimized scheduling result according to their size and the total number of threads.

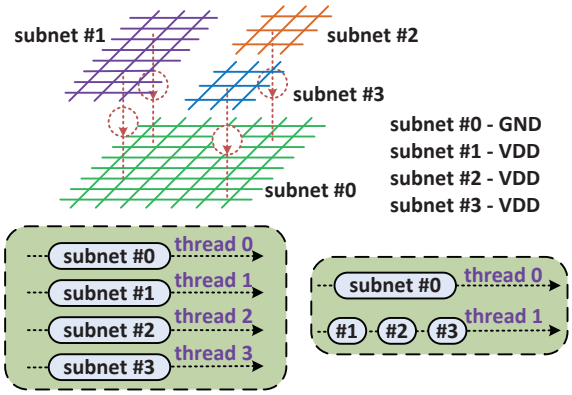


Figure 7: Multi-Threads Scheduling of Subnets.

4. EXPERIMENTAL RESULTS

All algorithms in PowerRush are implemented by C/C++ language. The simulation platform is a 64-bit RedHat Enterprise Linux Advanced Server with 2 Quad-Core Intel Xeon E5506 CPU@2.13GHz and 24GB RAM. The AMGPCG solver is same as [6]. Cholmod [14] and KLU [15] are from SuiteSparse [16]. The system built-in BLAS library is adopted in PowerRush simulator, while GotoBLAS is not used because we have found that GotoBLAS is slower than built-in BLAS because GotoBLAS probably cannot be properly optimized for our used hardware platforms. For Cholmod solver, two reordering methods of AMD and METIS are provided for selecting to be optimal according to Cholmod internal analyzing. Meanwhile, the conductance matrix of power grid is too sparse to allow efficient exploitation of supernodes so that supernodal technique is disabled for Cholmod solver. And for KLU solver, AMD reordering method is enabled but BTF is disabled because BTF pre-ordering can dramatically increase the fill-in in the LU factors when KLU is applied to power grid cases. The runtime and peak memory usage in transient simulation is measured by memtime [17]. We have to emphasize that, the peak memory usage given in this paper is the memory used for total simulator which includes SPICE parser, circuit builder and linear solver, not only for linear solver.

Several industrial power grid benchmarks which are drawn from real designs [12] are evaluated to validate the promising performance of proposed transient simulator. The simulation step size $h = 1ps$ and total simulation period $T = 1ns$, that is to say, transient simulation is performed for 1000 time points. The results of total runtime and memory usage for six IBM PG benchmarks are listed in Table 2. For all cases, transient simulation with Cholmod is faster than with KLU and uses less memory than with KLU solver because Cholmod just needs to compute the L factor while KLU needs to compute factor L and U. Also we perform net level parallelization with sing-thread BLAS for simulation with Cholmod solver. As shown in Table 2 for simulation with Cholmod solver, multi-threads simulation with net level parallelization can achieve $1.15X$∼$1.79X$ speedups. Of course the memory usage of simulation with multi-threads is more than simulation with single-thread but which is not relative significant.

Aiming to present the advantages of symmetric reformulation for transient simulation equations, especially we compare the memory consumption between Cholmod and KLU solver which are both from SuiteSparse package. Both of them are called by single-thread with Linux built-in BLAS library and without net level parallelization. For clarity, only GND net to be presented and compared because GND net usually is the largest net for each benchmark. As shown in Table 3, the peak memory of Cholmod solver just is about 29%∼68% of peak memory of KLU solver. All of these should benefit from the symmetric reformulation of transient simulation equation as demonstrated in Section 3.1.

Table 3: Memory Usage Comparison between Transient Simulation with Cholmod and with KLU. N_{GND} is node number of GND net. M_{KLU} is peak memory usage of KLU. $M_{Cholmod}$ is peak memory usage of Cholmod. The peak memory value (KB) is fetched by internal function of KLU and Cholmod solver, which only includes memory used of pure solver.

Grid	N_{GND}	M_{KLU}	$M_{Cholmod}$	$\frac{M_{Cholmod}}{M_{KLU}}$
ibmpg1t	13977	649.46	443.71	68%
ibmpg2t	83901	9211.77	4164.50	45%
ibmpg3t	530008	117802.00	38121.40	32%
ibmpg4t	604300	156724.00	45352.30	29%
ibmpg5t	528336	48463.00	24786.90	51%
ibmpg6t	746403	53508.80	31349.90	59%

The second annual Power Grid Simulation Contest has been successfully organized in 2012 TAU workshop by IBM Austin Research Lab to focus on transient analysis and parallel implementation. PowerRush is also evaluated with other simulators [18][19] on the same HW/SW platforms by a set of real life industrial benchmarks in this contest [20]. Comparing with the other teams, PowerRush obtained an extremely low memory consumption with relative fast simulation speed.

5. CONCLUSIONS

We have presented the detailed implementations of PowerRush for efficient transient simulation. A composite simulation flow is proposed for practical transient simulation of industrial power grid. Symmetric formulation for NA equation of RLC model is adopted in PowerRush to reduce the memory usage and improve matrix factorization efficiency. Naturally independent subnets in power grid are analyzed by multi-threads implementation for parallel simulation without accuracy lost. Also by taking the advantage of effective SPICE parser and robust circuit builder, PowerRush has shown to be an efficient and robust transient simulator for real power grid analysis. We only developed the coarse-grained parallel simulation, since its parallel efficiency really

Table 2: Transient Simulation Results of IBM Power Grid Benchmarks. The runtime (second) and peak memory value (KB) are measured by memtime [17]. T_{pre} is the runtime of preparation which includes SPICE parser and circuit builder procedures. The remaining runtime includes stamping matrix, building RHS, solving each linear equation and other related issues. T^{tot}_{KLU} is the total runtime of simulation with KLU solver. M^{tot}_{KLU} is the total peak memory usage of simulation with KLU solver. $T^{tot}_{Cholmod}$ is the total runtime of simulation with Cholmod solver. $M^{tot}_{Cholmod}$ is the total peak memory usage of simulation with Cholmod solver. Cholmod-MT means simulation with Cholmod solver and net level parallelization with multi-threads implementation. T^{tot}_{CMT} is the total runtime of multi-threads simulation with Cholmod solver. M^{tot}_{CMT} is the total peak memory usage of multi-threads simulation with Cholmod solver.

			KLU		Cholmod		Cholmod-MT			
Grid	Size	T_{pre}	T^{tot}_{KLU}	M^{tot}_{KLU}	$T^{tot}_{Cholmod}$	$M^{tot}_{Cholmod}$	T^{tot}_{CMT}	M^{tot}_{CMT}	$\frac{T^{tot}_{Cholmod}}{T^{tot}_{CMT}}$	$\frac{M^{tot}_{Cholmod}}{M^{tot}_{CMT}}$
ibmpg1t	30638	0.19	3.53	22612	3.36	21496	2.52	22600	1.33	1.05
ibmpg2t	127238	0.78	52.73	144316	36.93	115460	23.84	141508	1.55	1.23
ibmpg3t	851584	5.45	600.91	1290600	320.35	764732	213.11	1097936	1.50	1.44
ibmpg4t	953583	6.03	925.87	1618732	405.45	919952	226.44	1291120	1.79	1.40
ibmpg5t	1079310	7.16	308.58	938388	239.53	797112	157.66	926360	1.52	1.16
ibmpg6t	1670494	12.31	416.38	1322868	388.56	1181844	259.97	1557920	1.49	1.32

depends on the number of independent subnets of original power grid. Due to the rapidly increasing power grid size, efficient transient simulation is becoming more and more challengeable for both academic research and industrial applications. In the future, we plan to explore more practical and general parallel strategies in PowerRush simulator.

6. ACKNOWLEDGMENTS

The authors would like to thank Zhuo Li from IBM Austin Research Lab, for his careful organizing effort on 2012 TAU Transient Power Grid Contest.

7. REFERENCES

[1] R. Berridge, R. M. Averill III, A. E. Barish, and etl. IBM POWER6 microprocessor physical design and design methodology. *IBM Journal of Research and Development*, 51(6):685–714, November 2007.

[2] Tsung Hao Chen and Charlie Chung-Ping Chen. Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods. In *Proc. IEEE/ACM DAC*, pages 559–562, 2001.

[3] Min Zhao, Rajendran V. Panda, Sachin S. Sapatnekar, and etl. Hierarchical analysis of power distribution networks. In *Proc. IEEE/ACM DAC*, pages 150–155, 2000.

[4] Haifeng Qian, Sani R. Nassif, and Sachin S. Sapatnekar. Power grid analysis using random walks. *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, 24(8):1204–1224, 2005.

[5] Joseph N. Kozhaya, Sani R. Nassif, and Farid N. Najm. A multigrid-like technique for power grid analysis. *IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst.*, 21(10):1148–1160, 2002.

[6] Jianlei Yang, Zuowei Li, Yici Cai, and Qiang Zhou. PowerRush: A linear simulator for power grid. In *Proc. IEEE/ACM ICCAD*, pages 482–487, 2011.

[7] Rajat Chaudhry, Rajendran Panda, Tim Edwards, and David Blaauw. Design and analysis of power distribution networks with accurate rlc models. In *Proc. VLSI Design*, pages 151–155, 2000.

[8] U. Wever and Q. Zheng. Parallel transient analysis for circuit simulation. In *Proc. 29th Hawaii International Conference on System Sciences*, pages 442–447, 1996.

[9] He Peng and Chung-Kuan Cheng. Parallel transistor level full-chip circuit simulation. In *Proc. DATE*, pages 304–307, 2009.

[10] Ramachandra Achar, Miche S. Nakhla, Harjot S. Dhindsa, and etl. Parallel and scalable transient simulator for power grids via waveform relaxation (PTS-PWR). *IEEE Trans. Very Large Scale Integr. Syst.*, 19(2):319–332, February 2011.

[11] Chun-Jen Wei, Howard Chen, and Sao-Jie Chen. Design and implementation of block-based partitioning for parallel flip-chip power-grid analysis. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 31(3):370–379, March 2012.

[12] S. R. Nassif. IBM power grid benchmarks, [online]. `http://dropzone.tamu.edu/~pli/PGBench`.

[13] Farid N. Najm. *Circuit Simulation*. John Wiley & Sons, 2010.

[14] Yanqing Chen, Timothy A. Davis, Willam W. Hager, and etl. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):22–35, October 2008.

[15] Timothy A. Davis and Sivasankaran Rajamanickam. Algorithm 907: KLU, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software*, 37(3):36–52, September 2010.

[16] Timothy A. Davis. Suitesparse 3.7.2, [online]. http://www.cise.ufl.edu/research/sparse/SuiteSparse.

[17] memtime. `http://tiger.cs.tsinghua.edu.cn/Students/yangjl/memtime`.

[18] Ting Yu and Martin D. F. Wong. PGT_SOLVER: An efficient solver for power grid transient analysis.

[19] Xuanxing Xiong and Jia Wang. Parallel forward and back substitution for efficient power grid simulation.

[20] Zhuo Li, Raju Balasubramanian, Frank Liu, and Sani Nassif. 2012 TAU power grid simulation contest: Benchmark suite and results.