High Accurate Pattern Based Precondition Method for Extremely Large Power/Ground Grid Analysis*

Jin Shi[1], Yici Cai[1], Sheldon X.- D. Tan[2], Xianlong Hong[1]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China, 100084
Tel: +86-10-62785564
e-mail: shi-j03@mails.tsinghua.edu.cn
caiyc@mail.tsinghua.edu.cn
hxl-dcs@mail.tsinghua.edu.cn

[2]Department of Electrical Engineering, University of California at Riverside, CA 92521, USA
e-mail: stan@ee.ucr.edu

ABSTRACT

In this paper, we propose more accurate power/ground network circuit model, which consider both via and ground bounce effects to improve the performance estimation accuracy of on-chip power distribution networks. On top of this, a new precondition iterative method, which exploits geometry characters of power/ground networks, is developed to reduce memory usage and speed up the simulation. Experimental results show that the proposed method is about 5X faster than the *incomplete LU decomposition* (ILU) based preconditioned conjugate gradient iterative method and about half memory usage for simulating multi-layers large scale power/ground networks.

Categories and Subject Descriptors

B.7.2 [**INTEGRATED CIRCUITS**]: Design Aids;
B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids;
G.1.3 [**NUMERICAL ANALYSIS**]: Numerical Linear Algebra.

General Terms

Algorithm, Design, Performance

Keywords

Pattern, Precondition, Iterative method, PCG

1. INTRODUCTION

Report from ITRS reveals that the supply voltage will continuously shrink, and fall below 0.5 volt in the next few years [1]. On the other hand, the current of a chip will rise dramatically due to the increasing transistors and frequency. Both of the two factors challenge the power design of VLSI in several ways. Lower supply voltage not only leads to smaller noise margin, but also requires higher simulation accuracy to capture the power integrity problems. In the past, power distribution analysis is always a tradeoff between the simulation accuracy and the time constraints. Therefore, how to archive higher speed and how to simulate larger area was the main focus of the past several years.

Many papers have been published to discuss new algorithms to handle this problem, such as multi-grid method [2], the random walk method [3], the alternate direction implicit (ADI) method [4], the projection based passive model reduction method [5], the hierarchical based method [6], and the geometry based partition method [7]. Generally, existing methods work on very simplified power/ground wire models, which may leads to significant errors as supply voltage scale down. For instance, existing P/G simulation model does not consider via resistance, and most simulators ignore the ground bounce effect. According to our experiment results, the two effects can cause as much as 60% error to the simulation result. Further, as the process scales down and the number of metal layer increases, this ratio will increases. So we need to consider all those effects for better accuracy.

One efficient way to solve large scale linear systems is by means of iterative solution. [8] presented a preconditioned conjugate gradient (PCG) iteration method to perform P/G distribution analysis, but due to its simple precondition structure, it still consumes too much memory to handle huge problems.

As we add more details of physical effects and structures in the power/ground networks, we end up with more complicated circuits to solve. So very scalable and efficient dedicated linear solvers for power/ground networks are required.

In this paper, we propose to use more accurate power/ground models and improve the iteration method by using a better preconditioner. Our contributions are: (1) We explicitly model the vias resistance. (2) We analyze geometry character of P/G grid, observe that any three-dimension (3-D) P/G grid consists of many identical one-dimension (1-D) structure, each of which can be named a pattern. (3) We propose a new preconditioner to accelerate classical iterative process by using pattern structures of the grid to reduce memory usage. Also, we discuss the performance of this algorithm in parallel environment.

ISPD'06, April 9–12, 2006, San Jose, CA, USA.

* This work is supported by National Natural Science Foundation of China (NSFC) 60476014, National Hi-tech R&D Program of China 2005AA1Z1230, and Foundation of Intel Corporation

Our paper is organized as follows: Section 2 introduces the via effect and the ground bounce effect; Section 3 introduces new P/G grid model and patterns in multi-layer P/G grids; Section 4 introduces our method to generate the simulation matrix; in section 5, we construct a new preconditioner based on the pattern structure to accelerate the iteration process; Section 6 summarizes the new algorithm and gives out experimental results; section 7 mainly focuses on future work while section 8 concludes the whole paper.

2. Via Resistance and Ground Bounce

We first look at the accuracy loose due the ignorance of via resistance in today's technologies. In order to answer this question, we obtain some CPU benchmarks from our industrial partner, and then we extract the metal resistance using a commercial boundary element method (BEM) based tool. The results are lists below.

Table 1 Typical resistance in a CPU benchmark

(M1 is the bottom layer while M4 is the top layer)

Process	tsmc 130nm		
Layer Number	Typical Metal Resistance: Ohm	Typical Via Resistance: Ohm	Node Number
M1	8.95	0.68	3436
M2	0.65	0.68	96
M3	0.2	0.42	102
M4	3		14

In table 1, typical metal resistance stands for the algebraic average value of all the extracted resistors in a certain layer. Node number stands for the number of terminal nodes of all the extracted resistors in a certain layer. From the table, we can see that, via resistance is about 1/10 of the typical resistance of M1, and since there are no long wires run in M2 and M3, the typical resistance in the two layers is low. However, because the current in the bottom layer converge through the vias, the voltage droop of the vias can be amplified. Considering the node ratio of M1 to M2 is about 35, which means that 35 piece of current will flow through one via from M1 to M2, the voltage droop on a via (M1-M2) can equal to the droop on 3.5 piece of medium length power wire in M1. This droop is obvious, and if the current does not distribute evenly, the droop on some vias will become larger. Therefore, in order to get enough accuracy, we can not ignore via resistance when modeling the P/G grid. Another consideration is the ground bounce effect. If the power net and ground net are ideal symmetric and the pins of each cell are symmetric and aligned, the droop on power net will be the bounce on ground net. Unluckily, practical P/G network does not have this character especially when package effect is considered. So, in this paper, both the power net and ground net as well as the vias are modeled to guarantee the simulation accuracy.

3. New Grid Model and Patterns

Fig. 1 shows a typical two-layer P/G grid and its common model. In this model, wire between two neighbor vias is considered as a resistor while vias between two adjacent layers are neglected. As a result, the original 3-D grid structure degenerates to a 2-D mesh structure.

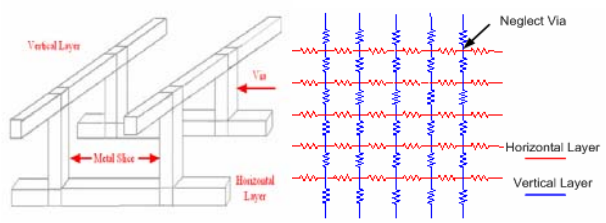

Figure 1: Two-layer P/G grid and its resistive model

For a P/G network with more than two layers, additional layers can be considered as denser mesh structure and be embedded into the original sparse mesh. The embedding process breaks long wires running in upper layer into many short pieces, which introduces many virtual nodes. Because no via is considered, two virtual nodes in adjacent layers are connected directly. This connection introduces errors because in actual 3-D structure, only terminal points of vias are considered nodes. This kind of nodes only connect two neighbor nodes in the same layer while in a mesh structure, each virtual node connects to four neighbor nodes in two different directions. In section 3, we will see that the reduction of connectivity could benefit the simulation matrix greatly.

Usually, metal wires in the same layer have the same parameters, such as length, width, pitch and thickness, so do the vias between two adjacent layers. This means that when we construct our simulation matrix, if we extract the P/G network wire by wire and layer by layer, the matrix elements we obtained will appear periodically, see section 3 for more detail. This is similar to building a wall with bricks. Therefore, we can call a group of repeating elements a pattern of the P/G grid.

Fig. 2 shows a typical pattern in a horizontal layer. Here resistors between node 1.1 and 1.5 have almost the same value and vias between layer 1 and layer 2 also have the same value. Because each layer is consisted of many identical patterns, we only need to story one typical pattern of each layer in memory rather than the complete grid structure. This kind of compaction is especially useful when the original P/G network is very regular and has many metal layers, which usually appears in complicated high performance CPU design. In theory, this kind of compaction can reduce the memory usage from $O(N)$ to $O(\sqrt{N})$. The proof is given below.

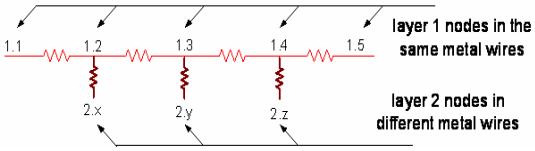

Figure 2: The pattern in a horizontal layer

Suppose there are k_i nodes in a metal wire of layer i, and p_i wires in layer i, then the total nodes in layer i will be $k_i \cdot p_i$. If we have t layers in total (from 1 to t), then we can get equation (1) because the number of nodes in each metal wire of layer i equals to the number of endpoint of vias which connect to both upper and lower layers running in the same direction.

$$k_i = p_{i-1} + p_{i+1} \quad 1 < i < t \quad (1)$$

Then the total number of nodes in the P/G grid, N , can be got using (2),

$$N = k_1 \cdot p_1 + \sum_{2}^{t-1} p_i \cdot (p_{i-1} + p_{i+1}) + p_t \cdot p_{t-1} \quad (2)$$

On the other hand, the total number of nodes after pattern based compaction, N_p can be computed by (3).

$$N_p = k_1 + \sum_{2}^{t} (p_{i-1} + p_{i+1}) + p_{t-1} \quad (3)$$

Because p_i / p_{i+1} is close to a constant in actual P/G grid, comparing (2) and (3), we can see that the memory cost of the compacted system is about $\sqrt{N}$. Reference [7] has discussed the voltage distribution within a local area in x-y plane. Comparing with it, the periodical appearance of elements in the same layer can be treated as locality in z direction. Therefore, when we simulate an extreme large P/G grid, we can first partition the whole grid in the z direction and then use locality in x-y plane to reduce the memory cost and gain acceleration.

4. Pattern Based Grid Extraction

A report about the extraction result of an industrial P/G network reveals that in a certain layer, more than 70% extracted resistors share the same value. This indicates us that a pattern based grid extraction process will save a lot of memory usage. By using the nodal analysis (NA) method, each resistor in the P/G grid model will stamp 4 elements in the simulation matrix. The stamp of a conductor is shown in Fig. 3.

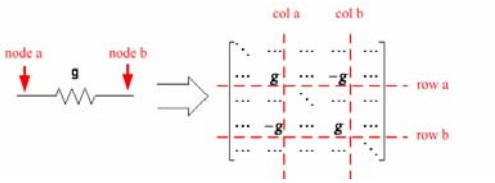

Figure 3: The stamp of resistor in the matrix

Now, if we number nodes in the P/G grid layer by layer and pattern by pattern as shown in Fig. 2, then we can obtain a new kind of simulation matrix, as shown in Fig. 4. Some enlarged details of Fig.4 are shown in Fig. 5. From these figures, we can observe that the simulation matrix is symmetrical, and the use of 1-D structure after considering the vias makes the simulation matrix a block matrix naturally. Now, one layer of the grid can be mapped into three blocks in the simulation matrix, one is a diagonal block consisting of metal wire elements and the other two are side block consisting of via elements. If we pay attention to the diagonal block, we can find that it is obviously consisted of a certain number of patterns in the block. Because each node only connects to two neighbor nodes in a pattern, the pattern remains a tri-diagonal from in the simulation matrix, as shown in Fig. 5. As a result, we only need to store only one pattern instead of each of them to reduce memory usage.

Further, although the side block in the matrix looks like a dense sub matrix, actually, all the elements in the side block share the same value. This is because all the vias in a certain layer have the same value. Therefore only one element is enough for the whole side block. Later, we can see that these structural characters of the matrix can be used to construct special preconditioner, which can achieve high performance.

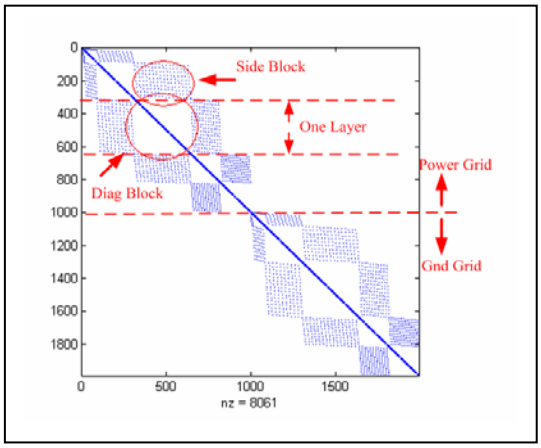

Figure 4: A typical matrix of a 7 layer regular grid when via and ground net are considered

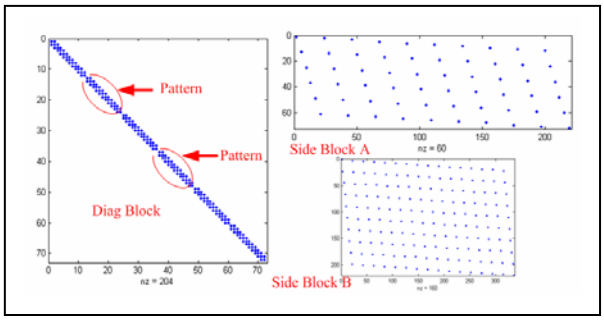

Figure 5: Details of sub blocks in Figure 7

For more irregular grids such as grips used in ASIC and DSP chips, the patterns are also obvious. We have examined some general cases, and find that the identical element ratio in the original extracted matrix can be as much as 62%.

5. Pattern Based Preconditioner

Iterative method is good at the error controlling, even some fast direct linear solvers have to be customized to hybrid ones to improve their robustness. However, the convergence speed of iterative method highly relies on the numerical character of the matrix and the performance of its preconditioner. However, we find that the numerical characters of matrices with and without vias are quite different. Table 2 compares the eigenvalue of two kinds of matrix. Actually, because vias are considered, new matrix is wider in band than the classical one (without considering vias), therefore, the new matrix has larger condition number than the classical one (larger eigenvalue ratio). The larger condition numbers usually means slow convergence rate in iteration process as the systems become stiff.

Table 2 Comparison on Matrix Eigenvalue

Matrix Size: 2k	Typical Max Eigenvalue	Typical Min Eigenvalue	Max/Min Ratio
New Matrix	1.0202e4	0.02123	4.80479e5
Classical Matrix	1.0226e3	0.04641	2.20317e4

Thus, in order to maintain high convergence rate, more effective preconditioner are desirable. Within the domain of P/G grid simulation, it is proven that the ideal preconditioner is a dense matrix, which means that the more memories we can afford, the faster we can reach. Already plagued by the huge problem size, we almost cannot afford any additional memory usage. Therefore, both the performance of the preconditioner and the memory usage needs to be considered.

Taking the most widely used preconditioner: incomplete LU decomposition (ILU) and incomplete Cholesky decomposition (ICD) [9] as examples, they will at least consume as much memory as the original matrix. On the other hand, it is obvious that the wider the band of the matrix is, the poorer the performance of ILU and ICD will be.

Comparing with the traditional preconditioners, our new pattern based method not only reduces memory usage but also accelerate the iteration convergence speed. The basic idea of our new preconditioner is to use pattern structure to approach the inverse matrix of the original one and do iteration from layer to layer to reduce the slowdown effect of side block caused by vias. Due to the tri-diagonal form of the pattern structure, the preconditioner can be obtained very easily. Because it utilizes the geometry information of the original matrix, it can help the iteration process to achieve much faster convergence than the traditional preconditioner.

Specifically, note that new matrix we can obtain is a symmetric block matrix (see Fig. 4), so we can use (4) to represent any triple-layer in a PG grid. It is not difficult to expand this structure from one triple to two and so on to describe the whole PG grid.

$$G=\begin{bmatrix} A & D & \\ D^T & B & E \\ & E^T & C \end{bmatrix} \quad (4)$$

Our algorithm begins with a block-gauss-elimination process which is carried on (4) to obtain an equivalent diagonal block form of *G*. The process actually is so called Schur decomposition. First it multiplies a matrix P_1 at the left hand side of G to get a new matrix T_1.

$$T_1 = P_1 \cdot G \quad (5)$$

Continuously, it multiplies matrix P_2 at the left hand side of T_1 to get matrix T_2.

$$T_2 = P_2 \cdot T_1 \quad (6)$$

Similarly, we can multiply matrix P_3 and P_4 at the left hand side of T_2 to get matrix T_4.

$$T_4 = P_4 \cdot P_3 \cdot T_2 \quad (7)$$

The construction of matrix P1 to P4 should make the matrix T4 a block diagonal matrix. Formula (8) demonstrates the detail of matrix P1 to P4 and T1 to T4.

$$P_1=\begin{bmatrix} I & 0 & 0 \\ -D^T\cdot A^{-1} & I & 0 \\ 0 & 0 & I \end{bmatrix} \quad T_1=\begin{bmatrix} A_1 & D & 0 \\ 0 & B_1 & E \\ 0 & E^T & C_1 \end{bmatrix} \quad \begin{matrix} A_1=A \\ B_1=B-D^T\cdot A^{-1}\cdot D \\ C_1=C \end{matrix}$$

$$P_2=\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & -E^T\cdot B_1^{-1} & I \end{bmatrix} \quad T_2=\begin{bmatrix} A_2 & D & 0 \\ 0 & B_2 & E \\ 0 & 0 & C_2 \end{bmatrix} \quad \begin{matrix} A_2=A_1 \\ B_2=B_1 \\ C_2=C_1-E^T\cdot B_2^{-1}\cdot E \end{matrix} \quad (8)$$

$$P_3=\begin{bmatrix} I & 0 & 0 \\ 0 & I & -E\cdot C_2^{-1} \\ 0 & 0 & I \end{bmatrix} \quad T_3=\begin{bmatrix} A_2 & D & 0 \\ 0 & B2 & 0 \\ 0 & 0 & C_2 \end{bmatrix}$$

$$P_4=\begin{bmatrix} I & -D\cdot B_2^{-1} & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \quad T_4=\begin{bmatrix} A_2 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & C_2 \end{bmatrix}$$

We know that $T_4 = P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot G$. Let matrix $Q = T_4^{-1} \cdot P_4 \cdot P_3 \cdot P_2 \cdot P_1$, then matrix Q can be treated as the ideal preconditioner for matrix G because $Q \cdot G = I$. Although we can not get the Q matrix directly, we can try to get a matrix-vector product $Q \cdot r$ approximately to perform excellent precondition. The process to get a vector as accurate as $Q \cdot r$ using the pattern structure is the main idea of our new preconditioner.

Usually, to get $Q \cdot r$ requires explicit use of matrix P_1 to P_4 and inexplicit use of matrix T_4. From (8) we know that in order to get T_4 and $P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$, A_1^{-1} , B_1^{-1} , C_2^{-1} should be got first. Because the matrix A and B and C are tri-diagonal matrix, if we can remain the tri-diagonal form in A_1, B_1 and C_1, we can benefit a lot. Formula (9) simplifies the way to get B_1 and C_2 in order to reduce computation complexity where $\wedge(x)$ operator means to obtain the main diagonal vector of x .

$$\begin{aligned} B_1 &= B-\wedge(D^T\cdot\wedge(A^{-1})\cdot D) \\ C_2 &= C_1-\wedge(E^T\cdot\wedge(B_2^{-1})\cdot E) \end{aligned} \quad (9)$$

This approximate B_1 and C_2 almost lose nothing because the original matrix is a positive definite one, which means the major big elements are distributed around the main diagonal. After this approximation, not only all the main diagonal blocks remain its tri-diagonal form, but also they remain the original pattern structures.

After we get T_4 using (9), let's see what we can benefit by using pattern structure in this process. One advantage to retaining the pattern structures is that the process of solving the main diagonal block can be divided into a loop in which only a small pattern structure should be solved. Further more, because the pattern structure remains in tri-diagonal form and repeated again and again in the main diagonal block, the LU decomposition of the pattern structure can be reused to improve the cache performance. Definitely, the process to update the diagonal part of original

matrix G to get T_4 only need to be computed once and can be reused in the later process.

Taking the way to solve equations $C_2 x = b$ as an example, because matrix C_2 is consisted of many pattern structures c_1 in tri-diagonal form as shown in (10), algorithm a_1 shown in Fig. 6 can be used to solve it.

$$C_2 = \begin{bmatrix} c_1 & & \\ & \ddots & \\ & & c_1 \end{bmatrix} \quad (10)$$

In this algorithm, the number pattern c_1 first was computed, then LU decomposition is performed on c_1. In the following loop, we only fetch small piece b_i from vector b and solve a pattern based equations $c_1 x_i = b_i$. Finally, all the x_i segments can be used to get the whole x vector. This pattern-based algorithm is quite different from the original direct LU solver. The main difference between them is that the decomposition only needs to be done once on a small tri-diagonal matrix and the result can be reused many times during the loop. Therefore, the efficiency of the solver is increased dramatically while the memory used for precondition is reduced to only one pattern size.

Another advantage of pattern structure is that the computation and memory cost of formula (9) is very low. Remember that all elements are identical in sub matrix, and each row of it only contains one element due to the connection topology of vias, thus

let $Y = \wedge(D^T \cdot \wedge(A^{-1}) \cdot D)$, we can get (11)

$$Y(i,i) = \sum_{d_{i,k} \neq 0} d_{i,k}^2 / a_{k,k} = d^2 / a_{k,k} \quad (11)$$

Until now, we can construct our new preconditioner $Q \cdot r$, where r is a residual vector, by two stage diagonal block solving processes.

In the first stage, we divide r into $r_1 \cdots r_3$ according to the block

$$X = P_1 \cdot r = \begin{bmatrix} I & 0 & 0 \\ -D^T \cdot A^{-1} & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 - D^T \cdot A^{-1} \cdot r_1 \\ r_3 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2' \\ r_3 \end{bmatrix} \quad (12)$$

$$P_2 \cdot P_1 \cdot r = P_2 \cdot X = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & -E^T \cdot B_1^{-1} & I \end{bmatrix} \begin{bmatrix} r_1 \\ r_2' \\ r_3 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2' \\ r_3 - E^T \cdot B_1^{-1} \cdot r_2' \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2' \\ r_3' \end{bmatrix}$$

structure of matrix G and try to get $P_2 \cdot P_1 \cdot r$. In this stage, the pattern solver mentioned in Fig. 6 is used to get $A^{-1} \cdot r_1$ and $B_1^{-1} \cdot r_2'$ described in formula (12), then r_2' and r_3' in (12) can be got by a direct matrix-vector multiplication operation followed by a vector subtraction operation.

Algorithm Name: a_1
Define: c_1: pattern of matrix C
N: row number of matrix C
k : row number of c_1
b : right hand size vector
Input: C, c_1, k, b
Output: solution vector x
1) get the number of patterns in C matrix , n=N/k
2) do LU decomposition on c_1 to get l_1 and u_1
3) for (i=0;i<n;i++)
4) {
5) read k elements from b to form piece b_i
6) solve $c_1 x_i = b_i$ using l_1 and u_1
7) Output x_i
8) }

Figure 6: The description of pattern solver algorithm

In the second stage, we revise the process sequence in (12) to start from the bottom vector r_3'. Similarly, we get $C_2^{-1} \cdot r_3'$ first and go on until we obtain result vector $P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$.

Fig. 7 describes the main part of pattern-based precondition algorithm. In this algorithm, step 1) to 7) belong to the top-down pass, and step 9) to 12) belong to the bottom-up pass. The top-down pass tries to move off the left side block in the original matrix while the bottom-up pass tries to move the right side block in order to get $P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$. Remember that we still need to multiply the result vector $P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$ by matrix T_4^{-1} to get the final preconditioned vector. The last step can be done using an inexplicit process trying to solve the equations $T_4 x = P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$. Because of the tri-diagonal form of matrix T_4 which is got using (9) in the very beginning step, the final preconditioned vector can be got easily once the bottom-up pass of vector $P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot r$ is finished.

The main advantage of this preconditioner is that it explicitly reduces the slowdown effect of side blocks to accelerate the convergence speed, and tries to precondition the matrix in a tri-diagonal format, which can extremely benefit from pattern structure of PG grid. We can observe that during the whole precondition process, patterns play a very important role to reduce memory usage and raise the cache performance in the loops of the algorithm. Due to the reusing of the decomposition of the pattern, convergence efficiency of the preconditioner is improved.

6. Experiment Result

We implement a new algorithm based on PCG method,

```
Algorithm Name: pattern-based preconditioner
Define: G: simulation matrix
        r : residual vector
        N: number of layers
        Di: diagonal block belonging to layer i
        Ui: upper block belonging to layer i
        Li: lower block to layer i
Input: G, r
Output: preconditioned vector Pr
1)   for(i=2;i<N;i++)
2)   {
3)      get piece ri belongs to layer i from r
4)      use algorithm a_1 take Di-1 and ri-1 as
           input parameter to get t1
5)      t2 = Li-1 * t1
6)      ri = ri - t2
7)   }
8)   use algorithm a_1 take DN and rN as input
        to get PrN
9)   for(i=N-1;i>=1;i--)
10)  {
        t1 = Ui * PrN
11)     ri = ri - t1
12)     use algorithm a_1 take Di and ri as
           input parameter to get Pri
13)     Output vector Pri
14)  }
```

Figure 7: Pattern-based preconditioner algorithm

using C++ language. The new method uses pattern based preconditioner. Also, the matrix-vector dot operation in traditional PCG method has been optimized using the pattern structure. Both the new algorithm and PCG method using *zero fill in incomplete LU* (ILU(0)) decomposition as its preconditioner are tested on a 2.8Ghz Intel Xeon Linux workstation with 2GB memory. Fig. 8 demonstrates their run time cost and memory usage. From this figure, we can see that the new method can gain about 5X speed acceleration while only consumes about half memory than the traditional ILU(0) based PCG method.

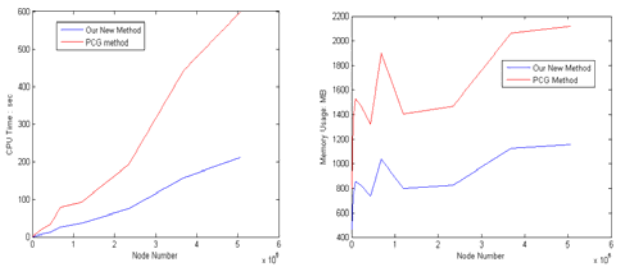

Figure 8: Run Time and Memory Usage comparison between PCG and our new method

7. Conclusion

As supply voltage continues to shrink, P/G simulation will become less accurate due to lack of accurate modeling. This trend will make us to consider more accurate model and reevaluate the iteration based solving technique. In this paper, we presented a high accurate and efficient simulation method to analysis large P/G girds. This new method not only model the via effect, ground bounce effect, but also exploits the pattern structure of the layered P/G grids to optimize the memory usage and boost the performance of traditional preconditioner. Experimental results have demonstrated the advantage of the proposed method over traditional preconditioned conjugate gradient iterative methods.

8. ACKNOWLEDGMENTS

The author would like to thank Dr. Eli. Chiprout from Intel Strategy CAD lab for his great help to this research work. He gave a lot of insightful suggestions and we've learned a lot from weekly discussions.

References and Citations

[1]. http://www.itrs.net/Common/2004Update/2004Update.htm

[2]. 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, no.10, Oct. 2002, 1148-1160

[3]. H. Qian, S. R. Nassif, S. S. Sapatnekar: "Random walks in a supply network", DAC 2003 Proceedings, 93-98

[4]. W. Guo, S. X. D. Tan, "Circuit level alternation-direction-implicit approach to transient analysis of power distribution networks", International Conference on ASIC Proceedings, 2003, Beijing, 246-249

[5]. J. M. Wang and T. V. Nguyen, "Expended Krylov Subspace method for reduced order analysis of linear circuits with multiple sources", In Proceeding of IEEE/ACM Design Automation Conference, pp. 247-252, Los Angeles, Jun. 2000.

[6]. M. Zhao, R. V. Panda, S. S. Sapatnekar and D. Blaauw, "Hierarchical analysis of power distribution networks", IEEE Trans. On Computer Aided Design, vol. 21, no. 2, pp. 159–168, Feb. 2002.

[7]. E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells" ICCAD 2004 Proceeding, pp 485-488

[8]. T. Chen and C. C. Chen: "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods", DAC2001 Proceedings, pp. 559-562

[9]. Saad, Yousef, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, 1996.