L Extracted? Now What?
Introduction to An Inductance-Wise Interconnect Simulation Engine (INDUCTWISE)

Tsung-Hao Chen , Hyoung-Suk Kim and Charlie Chung-Ping Chen
Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, 53706

Abstract

In this paper, we propose an efficient, accurate, and inductance-wise interconnect simulation engine, INDUCTWISE, which integrates novel inductance sparsification methods under time-domain PEEC model, advanced circuit formulation techniques, and excellent matrix solving algorithms. We develop a shared group calculation method to accelerate computing sparse reciprocal inductance matrix that is potentially up to 17X faster than the K-method[1][2] with almost identical accuracy. The nodal analysis formulation and the Cholesky decomposition enable INDUCTWISE to directly take K-elements as well as L-elements and help verifying the stability of reciprocal inductance matrices. Extensive simulations show that the combination of inductance reduction method (8X) and advanced matrix solving techniques (50X) achieve a total 400X speed-up over SPICE3.

1 Introduction

Parasitic on-chip inductance is growing as another design concern as the VLSI technologies march toward ultra-deep sub-micron and frequency approaches in the gigahertz range. Inductive coupling effect becomes more important because of the higher frequency signal contents, denser geometries, and reduction of both resistance and capacitance by copper and low-K devices. Inductance effects present not only in IC packages, but also in on-chip interconnects such as power grids, clock nets and bus structures. It causes overshoot, undershoot and oscillation of signals, and aggravates crosstalk and power grid noise. All of these seriously impact signal integrity on-chip. The importance and difficulty of on-chip inductance extraction and analysis are addressed in [3][4].

One of the main problems with inductance modeling is the long range coupling effects and the uncertainty of return paths since inductance is a function of a closed loop, which is hard to predict in advance before simulations. In fact, return paths are also frequency dependent and may change in a cycle-by-cycle basis due to different logic switch patterns and gate resistance. For this reason, A. Ruehli developed the famous Partial Element Equivalent Circuit method (PEEC) [5] models for parasitics modeling. PEEC method defines the partial self and mutual inductance for each wire segment with the assumption that each wire has return loop at infinity. Basically, PEEC models can be used to solve the inductance issues in either frequency and time domain. In the frequency domain, FastHenry [6] utilizes a multi-pole acceleration technique to speed up the extraction process. In the time domain, [7] proposes to directly simulate the PEEC model in the time domain to determine the return paths. This method has been shown to be accurate in a wide range of frequencies.

The PEEC models, however, leads to a large scale dense inductance matrix since the long range effects of inductive coupling, and the uncertainty of current return paths require the consideration of millions of mutual inductance terms to capture the inductance effects. The traditional circuit simulation engines may require hours or even days for such a large scale dense matrix simulation. To effectively reduce the mutual inductance terms, sparsification is crucial. However, it has been shown that direct truncation of the inductance matrix could result in instability [8]. Thus, an provable stable shift and truncate method was proposed by Krauter, et al [9]. This method assumes the return path is no longer at infinity but in a centered shell with radius r_0. Other methods such as Halo method [10] and block diagonal method [11] also reduce the number of mutual inductances by limiting the return path to the nearest power and ground returns. Later Beattie, et al [12] develops an exponential shell return paths for further sparsfication and shows that the sparsity is close to that of the K-method mentioned below.

Recently, a new K element method has been presented by Hao Ji, et al [1][2]. K is the reciprocal inductance matrix that is the inverse of the partial inductance matrix L. Since K has higher degree of locality similar to the capacitance matrix, it is more satisfactory to sparsify K without losing too much accuracy. Furthermore, [1] also shows that the reciprocal inductance matrix is diagonal-dominant and hence positive definite. The off diagonal terms are negative and can be safely deleted with sacrificing stability. Later Beattie, et al [12] also proposes to do double inversion on the inductance matrix and perform sparsification on both inductance and susceptance matrix.

There are, however, several issues for the existing inductance handling flow. First, after inductance extraction, it is required to perform circuit simulation to verify signal integrity issues. There is, unfortunately, a lack of effort to fundamentally speed up circuit simulation engine for inductance. Note that, one of the major reason for inductance sparsification is that the traditional circuit simulation engines cannot handle large-scale dense inductance matrix efficiently. Every sparsification algorithm is more-or-less trade-off runtime with accuracy. A capable circuit simulator can greatly enhance both the analysis of turn-around time and accuracy by including more mutual inductance terms. Second, the traditional circuit simulation engines cannot or do not handle the K-elements directly or efficiently. Although the double inversion algorithm has been proposed, the runtime is compromised by the double inversion time. Third, when the window size requirement is huge, the K-method can be very slow due to the inversion of the large-scale dense inductance matrix. Finally, we discovered that the diagonal-dominance property of the reciprocal inductance matrix does not sustain for general wire configurations and hence the stability of the K-method is actually questionable.

All the above issues lead to the urge for an inductance-oriented circuit simulation engines and that is exactly what we intend to provide in this paper. In this paper, we propose an efficient, accurate, and inductance-wise interconnect simulation engine, INDUCTWISE, which integrate novel inductance sparsification methods, advanced circuit formulation techniques, and excellent matrix solving algorithms. The runtime of the K-method will be significantly reduced when the accuracy requirement takes many conductors into consideration, since it needs to invert partial inductance matrix with some window size. For this reason, we develop a novel inductance sparsification method called SGC (Shared Group Calculation) method, which effectively reduces the number of matrix inversions while maintaining good accuracy. The SGC potentially accelerates the K-method up to 17X with almost identical accuracy. In addition, the utilization of nodal analysis formulation and the Cholesky decomposition provides two benefits. First it enables INDUCTWISE to directly take K-element for simulation. Second, the Cholesky decomposition is the best-known efficient method to check the positive definiteness of a matrix. Third, the Cholesky decomposition is 2 times more efficient and can achieve good accuracy regardless of the matrix ordering. Finally, the use of Time-domain PEEC model can simultaneously consider the frequency dependent effects as well as capacitance return paths. The user does not need to specify the return path, because the circuit simulator will find the return path itself. Extensive simulations show that the combination of inductance reduction method (8X) and advanced matrix solving techniques (50X) achieves a total 400X speed-up over SPICE3.

The rest of this paper is organized as follow. In section 2, we discuss the physical meaning of reciprocal inductors, the accuracy issues, and the stability issues more deeply, and proposed a group calculation algorithm. In section 3, we explain how we solve the reciprocal inductance matrix by the nodal analysis (NA) method, what the advantage of using NA is over the modified nodal analysis, and some other issues that affect the symmetricity of the system matrix. Section 4 shows the runtime and memory usage comparisons.

2 Inductance Extraction

In this section, we will first argue that time-domain rather than frequency domain PEEC method is an excellent tool to analysis on-chip inductance issues and provide additional reasons why we believe that efficient time-domain solver can be built with extreme efficiency. Later, we will introduce the basis of inductance analysis, K-method, and its issues.

2.1 Why Build Fast Time-Domain RLKC Solvers?

There are several ways to pursue inductance and return path issues. In the frequency domain, under the quasi-static assumption and accelerated by the Multipole expansion, FastHenry [6] is indeed an effective tool to produce accurate effective inductance on specific frequencies while only considering resistance and inductance. Users have to specify the most significant frequency for accurate results. When capacitance present, FastHenry needs to be used in a careful manner to ensure the correctness. A common mistake is the assumption of current return paths or the trial of finding the loop current without capacitance present since capacitance acts perfect return paths especially in the high frequency.

On the other hand, time-domain PEEC analysis does not have those issues. First, time-domain PEEC analysis can directly take resistance, inductance, and capacitance into consideration. Second, time-domain PEEC models are valid for wide range of frequency. Third, unlike the frequency domain method, the time-domain simulation does not involve complex number manipulation and hence can be solved in a more efficient manner. Furthermore, since PEEC models only consist of traditional lumped elements such as resistance, inductance and capacitance, the model circuits can be directly analyzed by any SPICE-compatible circuit simulators.

The showstopper of time-domain PEEC is that there is a lack of capable solvers that can simulate such a large-scale RLKC circuit efficiently. The traditional circuit simulators were not built for this purpose since parasitic inductance matrix is much denser than manually designed circuits. Not to mention that FastHenry has the multipole expansion algorithm to speed up.

However, we are very confident to build an efficient time-domain RLKC simulator that is at least two orders of magnitude faster than SPICE3. First, although inductance has longer-range effects than capacitance, the return currents usually bounded by the power/ground wires. For this short range of interaction, the multipole expansion may not provide enough runtime and memory benefit. Furthermore, with the advancement of inductance sparsification algorithms, the complexity and stability have been significantly improved. Second, SPICE was developed in 60 to 70's. The matrix solving techniques have been significantly improved in the last two decades. Furthermore, SPICE was mainly developed for human designed semiconductor circuits. It was not optimized for large-scale parasitic simulation and did not take advantage of the linear nature of parasitics.

In this paper, we not only developed an inductance-optimized efficient linear circuit simulator, but also developed and integrated a novel inductance sparsification algorithm which is up to 17X faster than the K-method. The integration of both methods shows over 400X speed over SPICE3. The runtime and experimental result will be shown in later sessions.

2.2 Inductance and Reciprocal Inductance Matrix

Given an inductance matrix $\mathcal{L}$, the reciprocal inductance matrix $\mathcal{S}$ is defined in the famous circuit book [13] by Kuh et al as $\mathcal{S} = \mathcal{L}^{-1}$. In [12], $\mathcal{S}$ was called susceptance. However, according to [14] susceptance is a general term for the imaginary part of the admittance matrix that can be caused by capacitance or inductance. We think the name of reciprocal inductance matrix is more specific to the inverse of the inductance matrix.

Each element in the partial inductance matrix is given by

$$\mathcal{L}_{ij} = \frac{\mu_0}{4\pi a_i a_j}\left[\int_{a_i}\int_{a_j}\int_{l_i}\int_{l_j}\frac{dl_i \cdot dl_j}{r_{ij}} da_i da_j\right] \tag{1}$$

where a_i and a_j are cross-sections of segment i and j, respectively, and r_{ij} is the geometric distance between two points in segment i and j. The magnetic vector potential along segment i caused by current $\mathbf{I}_j$ in segment j is defined as follows

$$A_{ij} = \frac{\mu_0}{4\pi a_j}\left[\int_{a_j}\int_{l_j}\frac{\mathbf{I}_j}{r_{ij}} dl_j da_j\right] \tag{2}$$

Therefore, for an $n \times n$ partial inductance matrix, the corresponding linear system equation can be written as follows

$$\begin{bmatrix} \mathcal{L}_{11} & \mathcal{L}_{12} & \cdots \\ \mathcal{L}_{21} & \mathcal{L}_{22} & \cdots \\ \cdots & \cdots & \mathcal{L}_{nn} \end{bmatrix}\begin{bmatrix} i_1 \\ : \\ i_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}(\int A_{1i} \cdot dl_1) \\ : \\ \sum_{i=1}^{n}(\int A_{ni} \cdot dl_n) \end{bmatrix} \tag{3}$$

Representing the system equation with $\mathcal{S}$, we get

$$\begin{bmatrix} \mathcal{S}_{11} & \mathcal{S}_{12} & \cdots \\ \mathcal{S}_{21} & \mathcal{S}_{22} & \cdots \\ \cdots & \cdots & \mathcal{S}_{nn} \end{bmatrix}\begin{bmatrix} \sum_{i=1}^{n}(\int A_{1i} \cdot dl_1) \\ : \\ \sum_{i=1}^{n}(\int A_{ni} \cdot dl_n) \end{bmatrix} = \begin{bmatrix} i_1 \\ : \\ i_n \end{bmatrix} \tag{4}$$

To effectively reduce the insignificant terms in the matrix, we can perform matrix sparsfications in either the inductance or the reciprocal inductance matrix, or even both. Since it was shown that reciprocal inductance elements have better locality than inductances [2], to sparsify $\mathcal{S}$ is more efficient than to sparsify $\mathcal{L}$.

2.3 Diagonal Dominance Issues of the Reciprocal Inductance Matrix

In [2], Hao Ji, et. al developed an advanced inductance sparsification method on $\mathcal{S}$ called K-method. They proved the stability of their algorithm based on the diagonal dominance property, which is derived from the assumption that all off-diagonal terms of the $\mathcal{S}$ matrix are negative. We now show that the property does not hold for general interconnect configurations.

From Equation (4), the physical meaning of each entry of $\mathcal{S}$ ($\mathcal{S}_{ij}$) is defined by the induced current in the $j-$th conductor caused by an active current in the $i-th$ conductor when the magnetic vector potential drop along conductor j is equal to one and those along all other conductors are set to zero. For example, as shown in Figure 1, to get the 3rd row of $\mathcal{S}$, we apply voltage 1 to conductor 3 and ground the other end as well as the two ends of other conductors. The induced currents on other conductors will be the off diagonal terms. It is positive if the current direction is the same, or negative otherwise. [2] argues that all the induced current are negative and use this property to proof the stability of their algorithm. We find out that the off-diagonal terms are not necessary to be negative for general wire cases. We now use Figure 1 to explain this.

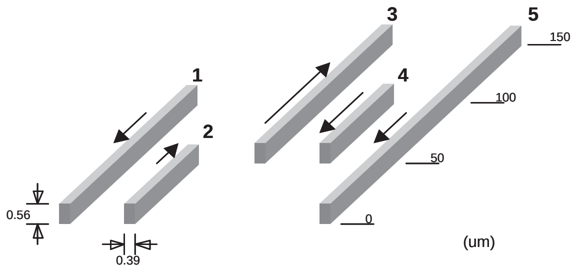

Figure 1: Example parallel conductors with unequal lengths

We calculate the partial inductance matrix using FastHenry [6] for Figure 1, and get

$$\mathcal{L} = \begin{bmatrix} 1.17 & 0.40 & 0.43 & 0.30 & 0.63 \\ 0.40 & 0.52 & 0.09 & 0.07 & 0.32 \\ 0.43 & 0.09 & 1.17 & 0.40 & 0.77 \\ 0.30 & 0.07 & 0.40 & 0.52 & 0.47 \\ 0.63 & 0.32 & 0.77 & 0.47 & 1.88 \end{bmatrix} \times 10^{-10} H \ . \tag{5}$$

Inverting $\mathcal{L}$, and the $\mathcal{S}$ matrix can be obtained as following

$$\mathcal{S} = \left[\begin{array}{rrrrr} 1.44 & -0.96 & -0.23 & -0.42 & -0.12 \\ -0.96 & 2.82 & 0.24 & 0.31 & -0.34 \\ -0.23 & 0.24 & 1.38 & -0.65 & -0.37 \\ \hline -0.42 & 0.31 & -0.65 & 3.02 & -0.41 \\ -0.12 & -0.34 & -0.37 & -0.41 & 0.88 \end{array}\right] \times 10^{10} H^{-1} \ . \tag{6}$$

It is clear to see that some of the off-diagonal terms are positive in (6). For instance, when calculating the 3rd row in (6), a unit magnetic vector potential is assigned on conductor 3, which demands a positive current along conductor 3 to accomplish. This current induces a positive magnetic vector potential drop on all other conductors (let's consider only conductors 1 and 2 in this explanation). To compensate this effect and make the gross magnetic vector potential drops along conductors 1 and 2 equal to zero, they have to carry negative currents. However, the current along conductor 1 also induces a current along conductor 2. Since the coupling effect between 1 and 2 is much stronger than the effect between 3 and 2, the overall effect causes conductor 2 carrying a positive current direction.

Thus, the physical definition of $\mathcal{S}$ should be calibrated as follows: *The magnetic vector potential drop along all conductors except the j^{th} are set to zero, and the magnetic vector potential drop along the j^{th} conductor is set to one. To satisfy the condition, there exists some current running along each conductor. The elements S_{ij} is the current flowing through the i^{th} conductor whose overall effect satisfies the vector potential drop assumption.*

Since the result is due to the overall effect (not a signal active line), negative off-diagonal elements are not guaranteed any more. This invalids the proof of the diagonal dominance property and hence the stability for the K-method becomes unsure.

2.4 Shared Group Calculation Algorithm

In this session, we present the runtime issue of the K-method and propose a new algorithm to resolve this issue.

The K-method was developed by Hao Ji, et. al [2] to perform sparsification during the construction of the reciprocal inductance matrix. It first selects a window size and successfully inverts the inductance matrix within this window. After inversion, it only takes a single column for the target conductor as the new column of the $\mathcal{S}$ matrix. It then iteratively performs this operation to all conductors. Using Figure 2 as an example, when the window size is 1, the K-method first inverts the $\mathcal{L}$ in the small window that contain only three conductors. Then it copies only the column of the inverse matrix to the $\mathcal{S}$ matrix as shown in Figure 3(a). Let n and m be the total number of conductors and the window size respectively. The total runtime of K-method is of $O(n(2m+1)^3)$. The runtime of K-method can be very huge when the window size requirement is huge. Most of the operations are wasted for the matrix inversion.

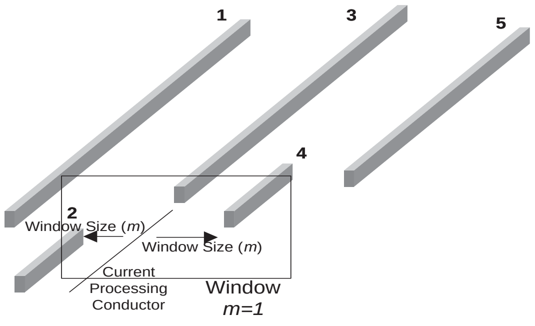

Figure 2: Window size limitation

For this reason, we develop SGC (Shared Group Calculation) method to improve the runtime. The key idea of the SGC method is to generate several columns of the S matrix at one inversion. Instead of taking only one conductor, we take p conductors, together with the m conductors from the windows on both sides, we only have to invert a $(2m+p) \times (2m+p)$ matrix once to get p columns. This matrix illustration can be found in Figure 3(b). We now present the algorithm of SGC as follows:

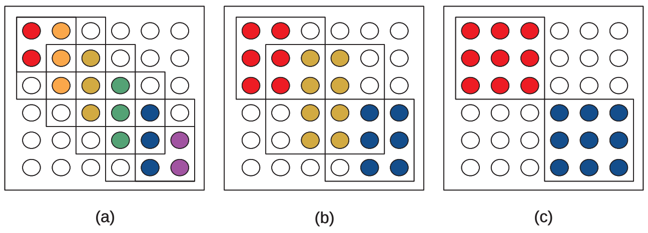

Figure 3: Illustration of the group calculation algorithm (a) Equivalent to K method when $p = 1$ (b) $m = 1$, $p = 2$ (c) Equivalent to block diagonal method when $m = 0$.

Shared Group Calculation Algorithm

1. At ith iteration, choose disjoint p conductors $l_{i_1}...l_{i_p}$ as active conductors and $2m$ neighbor conductors, and build the partial inductance matrix $\mathcal{L}^{(i)}$.

2. Solve the following equations with the small matrix $\mathcal{L}^{(i)}$,

$$\begin{aligned}\mathcal{L}^{(i)}\mathbf{x}^{(l_{i_1})} &= \mathbf{e}^{(l_{i_1})}\\ &\vdots\\ \mathcal{L}^{(i)}\mathbf{x}^{(l_{i_p})} &= \mathbf{e}_{(l_{i_p})}\end{aligned}$$

   The solution $\mathbf{x}^{(l_{i_1})}$ is the corresponding column l_{i_1} in sparsified reciprocal inductance matrix, and $\mathbf{x}^{(l_{i_2})}\cdots\mathbf{x}^{(l_{i_p})}$ are analogous.

3. Choose another p conductors as active conductors, and repeat steps 1 and 2 until all the conductors have been calculated.

Since we only have to perform matrix inversion once for every p conductors, the complexity of the SGC becomes $\mathbf{O}(\frac{n}{p}(2m+p)^3)$. Compare the runtime of the SGC algorithm and the K-method:

$$\frac{\text{SGC Algorithm}}{\text{K}-\text{Method}} = \frac{\frac{n}{p}(2m+p)^3}{n(2m+1)^3} = \frac{1}{p}\frac{(2m+p)^3}{(2m+1)^3} \quad .$$

If m is large (e.g. 50) and p is small (e.g. 4), the above equation is approach to

$$\frac{\text{SGC Algorithm}}{\text{K}-\text{Method}} \approx \frac{1}{p} \; ,$$

which shows p times speed-up can be achieved! Figure 4 shows the runtime speed-up of our method over the K-method. From this figure, we can see that up to 17X speed-up could be achieved. Further runtime improvement can be obtained by performing the LU or the Cholesky decomposition on $\mathcal{L}^{(i)}$ only once and solving only the columns that are required to form the $S^{(i)}$ matrix. In this way, we can avoid wasting computation effort on the unused columns.

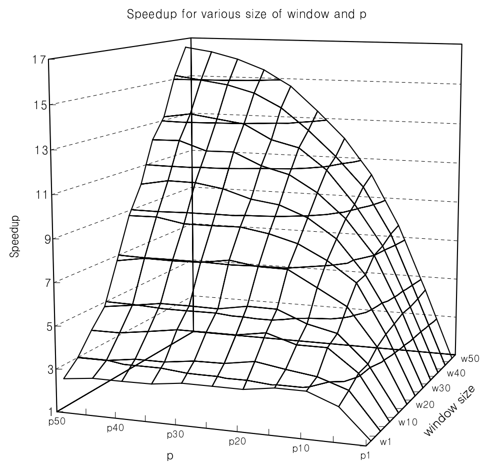

Figure 4: Speed-up of the Shared Group Calculation Algorithm over the K-method

The SGC method is actually a generalization of both the K-method and the block diagonal algorithm proposed in [11]. When $p = 1$, The SGC method degenerates to the K method in [2]. When $m = 0$, it becomes the block diagonal method in [11]. Figure 3 illustrates these different variations. Figure 3(a) shows the K-method, in which $m = 1$ and $p = 1$. Figure 3(b) shows an example of our group operation method, in which $m = 1$ and $p = 2$. Since the inverse of a block diagonal matrix is also a block diagonal matrix, Figure 3(c) is equivalent to the block diagonal method when $m = 0$ and $p = 3$.

2.5 Accuracy Comparisons

Figure 5 shows the simulation results of one example circuit. This circuit contains 128 parallel conductors whose lengths are randomly generated. The near end of each conductor connects to a 10Ω drive resistor, and then to the ground except the active line. The active line has an unit step input voltage. The far end of each conductor connects to a $20fF$ load capacitor, and then to the last conductor, which is assumed to be the power/ground line.

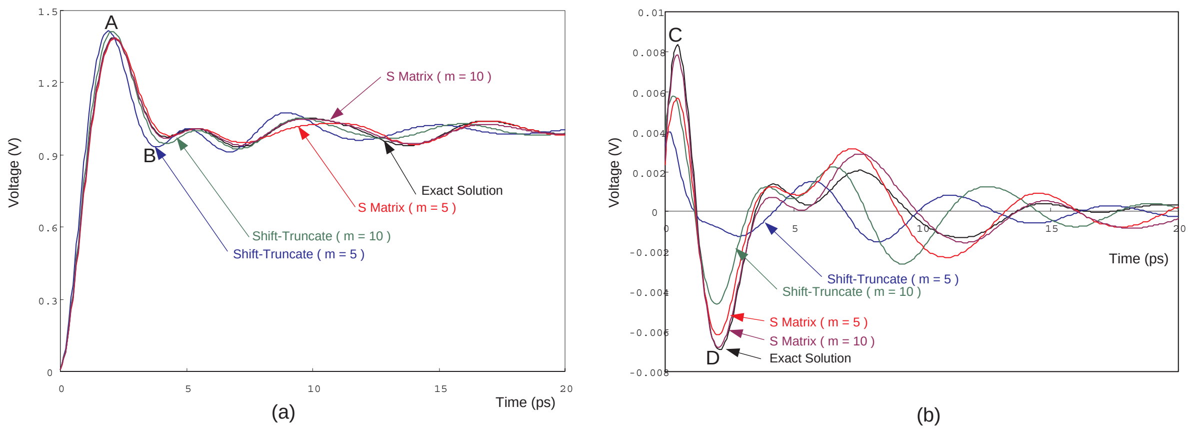

Figure 5: Comparison of the results of different sparsification methods (a) far end of the active conductor (b) far end of the 7th neighbor away from the active conductor

Four different sparsification algorithms are implemented, the K-method and the shift-truncate method both with window size 5 and 10. Figure 5(a) shows the far end of the active line, and Figure 5(b) shows the far end of the 7th neighbor away from the active conductor. From this figure, all implementations approach the exact solution for the active conductor pretty well. However, the 7th neighbor is covered by window size 10, but not window size 5. The shift-truncate method with window size 5 almost cannot reflect the far-away inductance effect.

In order to get detailed accuracy information, we extensively simulate 1000 randomly generated test cases according to the $0.13\mu m$ technologies. We calculate the respective errors of time and voltage on the 1st overshoot peak (e.g. A and C as shown in Figure 5) and 1st under-shoot peak (e.g. B and D) for both the active conductor and the 7th neighbor. The result is shown in Table 1. All of the setting in these five methods are the same as we mentioned previously.

	active line				7th neighbor			
	1st overshoot		1st undershoot		1st overshoot		1st undershoot	
Methods	time	voltage	time	voltage	time	voltage	time	voltage
Shift-truncate ($m = 5$)	9.56%	3.25%	17.52%	5.07%	36.80%	59.04%	40.79%	58.76%
Shift-truncate ($m = 10$)	6.62%	2.90%	11.34%	3.85%	18.27%	35.53%	20.08%	39.21%
K-method ($m = 5$)	1.90%	0.78%	6.29%	2.03%	16.46%	24.81%	24.50%	29.11%
K-method ($m = 10$)	0.74%	0.33%	3.41%	1.04%	4.25%	10.53%	12.32%	14.87%
SGC ($m = 10$, $p = 2$)	0.74%	0.33%	3.41%	1.08%	4.32%	11.07%	11.99%	15.11%

Table 1: Average errors of 1000 randomly generated cases for different sparsification methods

Table 1 shows that both the K-method and the SGC method are much more accurate than the shift-truncate method for the same window size. For the active line, the K-method with window size 10 (85% sparsity of $\mathcal{S}$ matrix) gives almost no error ($< 1\%$) for the first overshoot, which is concerned most in the signal integrity analysis, while the shift-truncate method can only reach 6.62% and 2.9% errors for the time and voltage respectively. As shown in the last row of Table 1, we can observe that the accuracy of our SGC algorithm is almost identical with the K-method.

Both the K- and shift-truncate methods cause a larger error on the far-away conductors. The window size actually affects the accuracy of the far-away neighbor rather than the active line. From this result, it turns out that a larger window size is necessary if we want to obtain better accuracy. Therefore, we conclude that *all the above mentioned inductance sparsification algorithms are not panacea. For the conductors do not cover by the windows size for any method and impacted by the active conductors, the accuracy will not be very good.* For example, in Figure 5, the waveform of conductor 7 will not matched when the windows size is less than 7 in any method. This reason urges the need for large window size and leads the need for the development of efficient large linear solvers in the next session.

3 Inductance-Wise Interconnect Simulation Engine

In this session, we present our efficient time domain RLKC simulator, INDUCTWISE. We will first focus on two circuit matrix formulations MNA (Modified Nodal Analysis) and NA (Nodal Analysis). Later, the way to deal with independent source in the NA formulation and the pros and cons of these two formulations will be discussed.

3.1 MNA Approach

First, we briefly review the MNA equations. Given a linear circuit, the adjacency matrix, $\mathbf{A}$, can be determined from the directed graph by the following rule.

$$\mathbf{A}_{ij} = \begin{cases} +1 & \text{if node } j \text{ is the source of branch } i \\ -1 & \text{if node } j \text{ is the sink of branch } i \\ 0 & \text{otherwise} \end{cases}$$

This matrix represents the connectivity of a circuit, and the Kirchhoff's law in terms of it is as follows,

$$\text{KCL}: \quad \mathbf{A}^T \mathbf{i}_b = \mathbf{0} \quad \text{and} \quad \text{KVL}: \quad \mathbf{A}\mathbf{v}_n = \mathbf{v}_b, \tag{7}$$

where $\mathbf{i}_b$ and $\mathbf{v}_b$ are the vectors of branch currents and voltages respectively, and $\mathbf{v}_n$ is the vector of the node voltages. For a circuit with RLKC elements and current sources, the adjacency matrix, the branch voltages and the branch currents can be partitioned into these forms.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{A}_g \\ \mathbf{A}_c \\ \mathbf{A}_l \end{bmatrix}, \quad \mathbf{v}_b = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_g \\ \mathbf{v}_c \\ \mathbf{v}_l \end{bmatrix}, \quad \mathbf{i}_b = \begin{bmatrix} \mathbf{i}_i \\ \mathbf{i}_g \\ \mathbf{i}_c \\ \mathbf{i}_l \end{bmatrix}$$

The subscripts i, g, c, and l stand for branches which contain independent current sources, resistors, capacitors, and inductors, respectively. The relationships between branch currents and voltages are as follows

$$\mathbf{i}_i = -\mathbf{I}_s(t), \quad \mathbf{i}_g = \mathcal{G}\mathbf{v}_g, \quad \mathbf{i}_c = \mathcal{C}\tfrac{d}{dt}\mathbf{v}_c, \quad \mathbf{v}_l = \mathcal{L}\tfrac{d}{dt}\mathbf{i}_l \tag{8}$$

In Equation(8), $\mathbf{I}_s(t)$ is the vector of current sources. The conductance matrix $\mathcal{G}$ and capacitance matrix $\mathcal{C}$ are diagonal matrices. Implementing the partial inductance extraction, the inductance matrix $\mathcal{L}$ is no more a sparse matrix, but whose diagonal elements are the values of self inductances and the off-diagonal terms are mutual inductances. It is known that $\mathcal{L}$ is symmetric and positive definite.

MNA combines Equation (7) and (8), and eliminates unnecessary branch currents. In RLKC circuits, only the branch currents running through inductors have to be kept in the equations. Then we obtain the following:

$$\tilde{\mathbf{G}}\mathbf{x} + \tilde{\mathbf{C}}\dot{\mathbf{x}} = \mathbf{b} \ , \tag{9}$$

$$\text{in which} \quad \begin{array}{l} \tilde{\mathbf{G}} = \begin{bmatrix} \mathbf{G} & \mathbf{A}_l^T \\ -\mathbf{A}_l & \mathbf{0} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{v}_n \\ \mathbf{i}_l \end{bmatrix}, \\ \tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathcal{L} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -\mathbf{A}_i^T \mathbf{i}_i \\ \mathbf{0} \end{bmatrix}. \end{array} \tag{10}$$

In (10), $\mathbf{G} = \mathbf{A}_g^T \mathcal{G} \mathbf{A}_g$ and $\mathbf{C} = \mathbf{A}_c^T \mathcal{C} \mathbf{A}_c$, which are symmetric and positive definite.

For transient analysis, the trapezoidal integration approximation of Equation (9) over the time interval $[kh, (k+1)h]$ is given by

$$\tilde{\mathbf{G}}\left(\frac{\mathbf{x}^{k+1} + \mathbf{x}^k}{2}\right) + \tilde{\mathbf{C}}\left(\frac{\mathbf{x}^{k+1} - \mathbf{x}^k}{h}\right) = \frac{\mathbf{b}^{k+1} + \mathbf{b}^k}{2}$$

It can be rewritten as follows,

$$\left(\tilde{\mathbf{G}} + \tfrac{2}{h}\tilde{\mathbf{C}}\right)\mathbf{x}^{k+1} = \left(-\tilde{\mathbf{G}} + \tfrac{2}{h}\tilde{\mathbf{C}}\right)\mathbf{x}^k + \mathbf{b}^{k+1} + \mathbf{b}^k \tag{11}$$

The MNA approach works for ordinary inductance sparse approximations, but not for the susceptance matrix. In this paper, we use this method to solve the exact solution and the shift-truncate sparsification method.

3.2 NA Approach

Although MNA provides a good solution for general circuits, the introduction of extra current variables makes the system matrix $\left(\tilde{\mathbf{G}} + \frac{2}{h}\tilde{\mathbf{C}}\right)$ in Equation (11) asymmetric, which crucially affects the runtime of decomposition procedure. In this subsection, we will show that Nodal Analysis is feasible for sparse reciprocal inductance matrix, and even has more advantages than MNA methods.

Substituting Equation (10) into (11) and performing block matrix operations, we can obtain two equations as follows,

$$\left(\mathbf{G} + \frac{2}{h}\mathbf{C}\right)\mathbf{v}_n^{k+1} + \mathbf{A}_l^T \mathbf{i}_l^{k+1} = \left(-\mathbf{G} + \frac{2}{h}\mathbf{C}\right)\mathbf{v}_n^k + \mathbf{A}_l^T \mathbf{i}_l^k - \mathbf{A}_i^T\left(\mathbf{I}_s^{k+1} + \mathbf{I}_s^k\right) \tag{12}$$

$$-\mathbf{A}_l \mathbf{v}_n^{k+1} + \frac{2}{h}\mathcal{L}\mathbf{i}_l^{k+1} = \mathbf{A}_l \mathbf{v}_n^k + \frac{2}{h}\mathcal{L}\mathbf{i}_l^k \tag{13}$$

Rearranging Equations (12) and (13), we get the following equations,

$$\left(\mathbf{G} + \frac{2}{h}\mathbf{C} + \frac{h}{2}\mathbf{S}\right)\mathbf{v}_n^{k+1} = \left(-\mathbf{G} + \frac{2}{h}\mathbf{C} - \frac{h}{2}\mathbf{S}\right)\mathbf{v}_n^k - 2\mathbf{A}_l^T \mathbf{i}_l^k - \mathbf{A}_i^T\left(\mathbf{I}_s^{k+1} + \mathbf{I}_s^k\right) \tag{14}$$

$$2\mathbf{A}_l^T \mathbf{i}_l^{k+1} = h\mathbf{S}\left(\mathbf{v}_n^{k+1} + \mathbf{v}_n^k\right) + 2\mathbf{A}_l^T \mathbf{i}_l^k \tag{15}$$

where $\mathbf{S} = \mathbf{A}_l^T \mathcal{S} \mathbf{A}_l$, and $\mathcal{S}$ is the reciprocal inductance matrix that equals to $\mathcal{L}^{-1}$. Let $\mathbf{Y} = \left(\mathbf{G} + \frac{2}{h}\mathbf{C} + \frac{h}{2}\mathbf{S}\right)$, since $\mathbf{G}$, $\mathbf{C}$ and $\mathbf{S}$ are all admittance. $\mathbf{Y}$ has shown to be symmetric and positive definite [15]. Thus, the Cholesky Decomposition or Preconditioned Conjugate Gradient iterative method can be applied to solve the matrix. Please refer to [15] for the proof and detail derivation.

```
Initialization:
    Solve DC solutions (or use initial conditions)
    for $\mathbf{v}_n^0$ and $\mathbf{i}_l^0$.
Iterations:
  Phase 1:
    Solve Equation (14) and get $\mathbf{v}_n^{k+1}$.
  Phase 2:
    Perform matrix multiplications and vector
    summations in Equation (15) and obtain
    $2\mathbf{A}_l^T \mathbf{i}_l^{k+1}$ for the next iteration.
End:
```

Table 2: Nodal Analysis for RLKC circuits

Using the sparsification method shown in previous section, matrix $\mathbf{S}$ is a sparse matrix, which keeps $\mathbf{Y}$ still a sparse matrix.

Table 2 shows our NA transient simulation procedure for RLKC circuits. Compared equations (11) of MNA and (14) of NA. We will perform a comparison studies between MNA and NA methods in the next session.

3.3 Implementation Considerations: A Comparison Study

We now compare the pros and cons between MNA and NA algorithms. Later we will introduce more advanced matrix solving techniques with an emphasis on the famous matrix reordering algorithms.

Since the MNA matrix $\left(\tilde{\mathbf{G}} + \frac{2}{h}\tilde{\mathbf{C}}\right)$ in (11) is a positive definite but asymmetric matrix, LU factorization is unavoidable for MNA analysis even when we switch the sign of $-\mathbf{A}_l$ and $\mathcal{L}$ in (10) to be an symmetric but indefinite matrix. On the other hand, since $\mathbf{Y}$ of NA is symmetric and positive definite, the Cholesky decomposition can be applied. There are several well-known benefits of the Cholesky decomposition over the LU decomposition. First, the runtime and memory requirements of the Cholesky decomposition in half as those of the LU decomposition since the former can take the advantage of symmetry. Second, LU decomposition requires advanced reordering and pivoting algorithm to enhance numerical conditions and avoid breaking down. However, it has been shown that the accuracy of the Cholesky decomposition is always the best regardless of the matrix ordering. Matrix reordering for the Cholesky is usually performed only for fill-in reduction. However, the sparsity of the NA formulation is often slightly worse than MNA since $\mathbf{S} = \mathbf{A}_l^T \mathcal{S} \mathbf{A}_l$ introduces more matrix entries than $\mathcal{L}$. However, we believe that the additional entries are offset by the saving of symmetry. Finally, the Cholesky decomposition is the best known most efficient way to check the positive definiteness of a matrix. Actually, it can serve a check for the stability of the inductance and reciprocal inductance matrix. When the Cholesky decomposition fails for the NA formulations, it sounds alarms for the instability of the $\mathcal{S}$ matrix. Therefore, it served as a guard for the K-method as well as our SGC method. It is also suggested to directly perform the Cholesky decomposition to $\mathcal{S}$ to check its stability.

It is well known that the computation time of the decomposition is dominated by the number of fill-ins and the matrix ordering plays a crucial role to the fill-ins. The reduction of fill-ins not only saves the runtime of the decomposition but also has tremendous benefit for later transient simulation since we have a smaller amount of matrix entries of the triangle matrices. It is also known that it is easier and more efficient to perform matrix reordering to symmetric matrices.

About matrix reordering algorithms, there are just so many of them such as RCM (Reverse Cuthill-McKee), MD (Minimum Degree), ND (Nested Dissection) methods and their variants. To the authors' knowledge and experimental results, we discover that MD is one of the most efficient ways to reduce fill-ins. In our work, we implemented a variant of the minimum degree (MD) algorithms. There are basically three forms of the Cholesky decomposition methods: row-major, column-major, and sub-matrix methods. Different methods have different runtime, memory consumption, and memory access patterns. In our experience, we discover column-major is easier, very efficient, and cache friendly. Finally, we note that when there exist huge amount of medium size dense blocks in the $\mathcal{S}$ or $\mathcal{L}$ matrix, it might be beneficial to use a hybrid sparse-dense matrix data structure.

3.4 Handling Independent Voltage Sources

In case there are independent voltage sources in the circuit, we have to add extra current variables in the MNA equations. Thus Equation (9) becomes

$$\begin{bmatrix} \tilde{\mathbf{G}} & \mathbf{A}_v{}^T \\ \mathbf{A}_v & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{i}_v \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{C}} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{i}}_v \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{v}_s, \end{bmatrix} \tag{16}$$

where $\mathbf{A}_v$ and $\mathbf{v}_s$ are the adjacency matrix and vector of values for voltage source elements respectively. In our implementation for NA, we transform the voltage source into Norton equivalent circuit as shown in Figure 6.

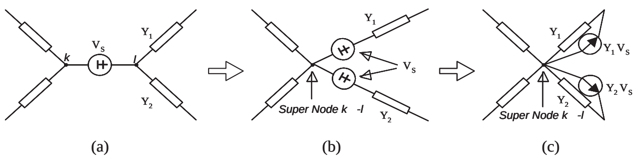

Figure 6: Voltage source transformation

If the voltage source connects to R or C elements, this method can be easily implemented. Norton equivalent circuits for R and C elements are available. However, coupling of inductances makes this transformation ineligible for L elements.

Consider the circuit shown is Figure 7(a), which shows a voltage source connects to one terminal of two coupled inductances. Using frequency domain analysis, the current-voltage equations on its two ports are as follows.

$$\begin{aligned} I_1 &= \frac{S_{11}}{s}(V_1 - V_S) + \frac{S_{12}}{s}V_2 \\ I_2 &= \frac{S_{12}}{s}(V_1 - V_S) + \frac{S_{22}}{s}V_2 \end{aligned}$$

These two equations can be rewritten as

$$\begin{aligned} I_1 &= \frac{S_{11}}{s}V_1 + \frac{S_{12}}{s}V_2 - \frac{S_{11}}{s}V_S \\ I_2 &= \frac{S_{12}}{s}V_1 + \frac{S_{22}}{s}V_2 - \frac{S_{12}}{s}V_S \end{aligned}$$

which can be represented as the circuit shown in Figure 7(b). The voltage source is replaced by current sources. Since all of conductance (G), capacitance (C) and reciprocal inductance (S) are admittance, they have the same property of equivalent circuits. Thus it can be applied to our NA analysis.

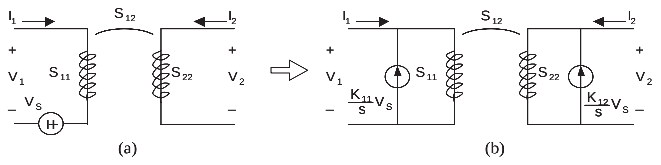

Figure 7: Norton equivalent liked transformation for inductances

4 Experimental Results

We implemented the shifted-truncated, K-method, and SGC methods in C/C++ language, and simulated on an AlphaPC DP264 666Mhz system. Table 3 shows two test cases that contain 256 and 512 parallel conductors respectively. For each test case, we simulate the exact solution and the shift-truncate method by both SPICE3 and INDUCTWISE, and the K-method using INDUCTWISE. We select the sparsity of the shift-truncate method, which makes it reach the same accuracy with the K-method. All of the transient simulations contain 400 time-steps. Table 3 shows that INDUCTWISE is 50X faster than SPICE3 for 512 conductors, and 2X memory saving.

	sparsity(%)	runtime(s)	memory(M)
# of conductors	256		
Excat (SPICE3)	-	266.91	21
Exact (Ours)	-	16.09	12
Shift (SPICE3)	45	132.94	17
Shift (Ours)	45	7.80	10
S Matrix (Ours)	90	1.88	5.6
# of conductors	512		
Excat (SPICE3)	-	4016.96	74
Exact (Ours)	-	77.61	43
Shift (SPICE3)	45	2334.59	57
Shift (Ours)	45	57.13	28
SGC (Ours)	90	10.99	15

Table 3: SPICE3 vs. INDUCTWISE

Table 4 compares the exact solution, the K-method, and the SGC algorithm with our simulator on two test cases, one is 1024 parallel conductors, and the other one is a large test case, 10000 conductors. In this table, inversion time means the run time K-method or the SGC algorithm takes, decomposition time means the period performing the LU or Cholesky decomposition, and iteration time means the time transient simulation takes for total 400 time steps. The output waveforms for 1024 conductors are shown in Figure 8. We can see that with window size 55, both the susceptance method and the SGC method have almost 100% accuracy, and saves about 96% of the decomposition time. With group number 4, the group calculation save 67% of the inversion time.

References

[1] A. Devgan, H.Ji, and W. Dai. How to efficiently capture on-chip inductance effects: Introducing a new circuit element k. In *ICCAD*, November 2000.

[2] H. Ji, A. Devgan, and W. Dai. Ksim: A stable and efficient rkc simulator for capturing on-chip inductance effect. In *DAC*, June 2001.

[3] M.W. Beattie and L.T. Pileggi. Inductance 101: Modeling and extraction. In *DAC*, June 2001.

	Exact	Basic	SGC
# of conductors	1024		
window size (m)	-	55	53
group number (p)	-	1	4
sparsity of $\mathcal{L}$ (%)	-	90	90
Inversion time (s)	-	2.90	0.97
Decomposition time (s)	229.61	7.67	5.45
Iteration time (s)	223.56	71.55	69.25
Memory (MB)	457	83	78
# of conductors	10000		
window size (m)	-	80	78
group number (p)	-	1	4
sparsity of $\mathcal{L}$ (%)	-	98.4	98.4
Inversion time (s)	-	86.83	24.37
Decomposition time (s)	-	176.68	110.1
Iteration time (s)	-	1348.39	1359.74
Memory (MB)	-	545	545

Table 4: Comparison of Exact solution, basic susceptance algorithm and group calculation algorithm

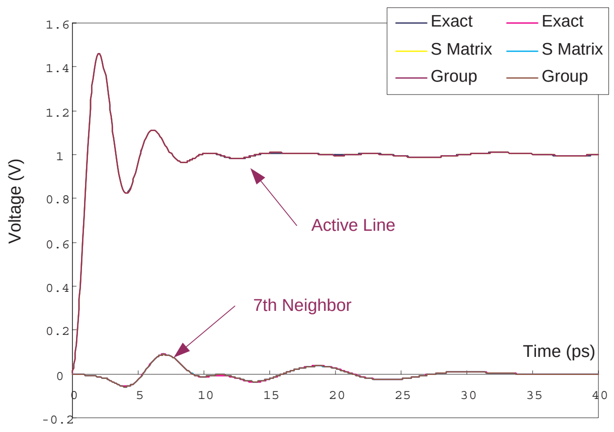

Figure 8: Waveforms of 3 different methods in Table 4 for 1024 conductors

[4] K. Gala, D. Blaauw, J. Wang, V. Zolotov, and M. Zhao. Inductance 101: Analysis and design issues. In *DAC*, June 2001.

[5] A.E. Ruehli. Inductance calculatioin in a complex integrated curcuit environment. *IBM Journal of Research and Development*, pages 470–481, September 1972.

[6] M. Kamon, M.J. Tsuk, and J.K. White. Fasthenry: A multipole-accelerated 3-d inductance extraction program. In *DAC*, June 1993.

[7] N. Chang, S. Lin, J. Lillis, and C.K. Cheng. Inteconnect analysis and synthesis.

[8] Z. He, M. Celik, and L.T. Pileggi. Spie: Sparse partial inductance extraction. In *DAC*, 1997.

[9] B. Krauter and L.T. Pileggi. Generating sparse partial inductance matrices with guaranteed stability. In *ICCAD*, November 1995.

[10] K.L. Shepard and Z. Tian. Return-limited inductances: A practical approach to on-chip inductance extraction. *IEEE Trans. on Computer Aided Design of Integrated Ciurcuits and Systems*, 19(4):425–436, April 2000.

[11] K. Gala, V. Zolotov, R. Panda, B. Young, J. Wang, and D. Blaauw. On-chip inductance modeling and analysis. In *DAC*, June 2000.

[12] M.W. Beattie and L.T. Pileggi. Modeling magnetic coupling for on-chip interconnect. In *DAC*, June 2001.

[13] L.O. Chua, C.A. Desoer, and E.S. Kuh. Linear and nonlinear circuit. McGRAW-HILL, 1987.

[14] http://whatis.techtarget.com/definition/0,,sid9_gci531197,00.html.

[15] T.H. Chen and C.C.P. Chen. Efficient large-scale power grid analysis based on preconditioned krylov-subspace iterative methods. In *DAC*, June 2001.