

SASIMI: Sparsity-Aware Simulation of Interconnect-Dominated Circuits with Non-Linear Devices

Jitesh Jain, Stephen Cauley, Cheng-Kok Koh, and Venkataramanan Balakrishnan
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907-1285
{jjain,stcauley,chengkok,ragu}@ecn.purdue.edu

Abstract

We present a technique for the fast and accurate simulation of large-scale VLSI interconnects with nonlinear devices, called SASIMI. The numerical efficiency of this technique is realized through linear-algebraic techniques that exploit the sparsity and structure of the matrices that are encountered in VLSI structures. Numerical results show that SASIMI is up to 1400 times as fast as commercial-grade SPICE, for moderate-size circuits, with little sacrifice in simulation accuracy.

1 Introduction

With aggressive technology scaling, the accurate and efficient modeling and simulation of interconnect effects has become (and continues to be) a problem of central importance. For accurate modeling of the distributive effects of interconnects, it is necessary to model a long wire using many segments of lumped RLC elements. Owing to the inductive and capacitive coupling between these elements, direct simulation of the resulting models comes at a very high (and often unacceptable) simulation cost. It has been highlighted in the past that SPICE [11] is not amenable to the simulation of interconnect-dominated structures, such as power/ground networks, clock networks, and busses.

There have been many studies to eliminate the bottleneck due to SPICE in the past several years. A representative selection of these advances in the simulation of large-scale interconnects, which exploit the *locality* of capacitive and inductive coupling effects, can be classified into techniques at the modeling and simulation levels as follows:

Modeling level techniques: In [5], the authors propose a new circuit element K to capture the inductive coupling using the inverse of the inductance matrix. Window-based extraction of K elements are proposed in [2], [15], and [16]. In [8], the authors use controlled voltage and current sources to construct a SPICE-compatible circuit model for K elements. A similar concept, called Vector Potential Equivalent Circuit (VPEC), is introduced in [12] to obtain a localized circuit model for inductive interconnects. These techniques involve the inversion of the inductance matrix. To avoid matrix inversion, the authors of [16] propose a wire duplication-based interconnect model, in which the authors construct a sparse equivalent circuit by windowing the inductance matrix. However, it should be emphasized that all these techniques ultimately rely on SPICE (and its variants) for the simulation of dynamic responses, and thus are constrained by the limitations of SPICE.

Simulation level techniques: The underlying solver in a simulation tool essentially addresses the problem of solving $Ax = b$ fast. In [14], the authors propose a hierarchical analysis of power distribution networks. In this work the authors partition the power grid and model each partition as a macro-model with a sparsified port admittance matrix. In [3], INDUCTWISE, an efficient simulation tool for cir-

cuits modeled using conductance (G), capacitance (C), and K elements, is proposed. In [14], and [3], it is shown that the Cholesky factorization of the A matrix, which is composed of G , C , and/or K , can be very efficient due to the inherent sparsity of A . However, the inverse of A is dense, which in turn implies that each simulation step involves dense matrix-vector multiplications. In contrast, the authors of [7] perform linear circuit simulation using a new formulation, called RLP, to model circuits by resistance (R), inductance (L), and the inverse of capacitance (P). Although the resulting A matrix in the RLP formulation is dense, its inverse is sparse, which enables fast sparse matrix inversion and sparse matrix-vector multiplication in each simulation step. The above methods however, are unable to handle non-linear devices. This problem is addressed in [4], where the approach in [14] is extended to handle non-linear devices. However, it also inherits the limitation that the sparsity of matrix A is exploited only for fast Cholesky factorization, and not in each simulation step.

In this paper, we propose SASIMI which uses *sparsity-aware simulation techniques for interconnect-dominated circuits with non-linear devices*. Our contribution in this paper is two fold. First, we extend the RLP formulation as described in [7] to include non-linear devices, without sacrificing the computational benefits achieved due to sparsity of the linear system. It should be noted that the A matrix involved in the solution of the linear system is constant throughout the simulation. In contrast, the A matrix involved in solving the non-linear system changes in each simulation step. However, the A matrix is sparse. Due to the sparse and time varying nature of the problem at hand Krylov subspace based iterative methods could be used for efficient simulation. Our second contribution is to introduce a novel preconditioner constructed based on the sparsity structure of the non-linear system. The inverse of the preconditioner has a compact representation in the form of the Hadamard product [10], which facilitates not only the fast computation of the inverse, but also the fast dense matrix-vector product. Experimental results show that SASIMI is up to 1400 times faster than commercial grade SPICE, even for moderate-size circuits.

2 Mathematical Preliminaries

VLSI interconnect structures, with non linear devices can be analyzed using the Modified Nodal Analysis (MNA) formulation which yields equations of the following form

$$\tilde{G}x + \tilde{C}\dot{x} = b, \quad (1)$$

where

$$\tilde{G} = \begin{bmatrix} \tilde{G} & A_l^T \\ -A_l & 0 \end{bmatrix}, \quad \tilde{C} = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}, \quad x = \begin{bmatrix} v_n \\ i_l \end{bmatrix},$$

$$b = \begin{bmatrix} A_i^T I_s + I_{nl} \\ 0 \end{bmatrix}, \quad G = A_g^T R^{-1} A_g, \quad \text{and} \quad C = A_c^T C A_c.$$

R denotes the resistance matrix. The matrices G , L and C are the conductance, inductance and capacitance matrices respectively, with corresponding adjacency matrices A_g , A_l and A_c . I_s is the current source vector with adjacency matrix A_i , and v_n and i_l are the node voltages and inductor currents respectively.

Vector, I_{nl} captures the effect of non-linear loads and depends on the node voltages as $I_{nl} = f(v_n)$. f is a function which varies depending on the load characteristics and in general can be a non-linear function.

With N denoting the number of inductors, we note that

$$L, C, R \in \mathbf{R}^{N \times N}, \quad C, G \in \mathbf{R}^{2N \times 2N}.$$

Differential equations such as (1) can be numerically solved using standard algorithms like the trapezoidal method [1]. Considering a uniform discretization of the time axis with resolution h , $x^k = x(kh)$. Using the approximations

$$\left. \frac{d}{dt} x(t) \right|_{t=kh} \approx \frac{x^{k+1} - x^k}{h} \quad \text{and} \quad x^k \approx \frac{x^{k+1} + x^k}{2}$$

over the interval $[kh, (k+1)h]$, the determination of x^{k+1} from x^k requires the solution of a set of linear and nonlinear equations:

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

and

$$I_{nl}^{k+1} = f(v_n^{k+1}). \quad (3)$$

The nonlinearity in the above set of equations can be handled by the standard Newton-Raphson technique of linearization (3) and iterating until convergence: Equation (2) is a linear equation of the form $\mathcal{L}(x) = 0$, where we have omitted the iteration index k for simplicity. Equation (3) is a nonlinear equation of the form $g(x) = 0$. Let $g(x) \approx \mathcal{G}(x)$ be a linear approximation of $g(x)$, linearized around some $x = x_0$. Then, simultaneously solving $\mathcal{L}(x) = 0$ and $\mathcal{G}(x) = 0$ yields numerical values for x and hence v_n . These values are then used to obtain a new linear approximation $g(x) \approx \mathcal{G}_{\text{new}}(x)$, and the process is repeated until convergence. A good choice of the point x_0 for the initial linearization at the k th time-step is given by the value of v_n from the previous time-step.

A direct implementation of this algorithm requires $O(pqn_1^3)$ operations, where p is the number of time steps, q is the maximum number of Newton-Raphson iterations in each time step, and $n_1 = 3N$.

3 The RLP formulation

The mathematical framework that underlies our approach is an alternative formulation of the MNA equations that uses the resistance, inductance and the inverse of the capacitance matrix. This is the so-called ‘‘RLP formulation’’, first proposed in [7].

We begin by decomposing C , A , and A_i as:

$$C = \begin{bmatrix} C_{cc} & C_{cv} \\ C_{vc} & C_{vv} \end{bmatrix}, \quad A^T = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix}, \quad A_i^T = \begin{bmatrix} A_{i1}^T \\ A_{i2}^T \end{bmatrix}$$

$$I_{nl} = \begin{bmatrix} 0 \\ I_v \end{bmatrix}, \quad v_n = \begin{bmatrix} v_c \\ v_v \end{bmatrix}. \quad \text{Here } C_{vv} \text{ denotes the sub-matrix}$$

of the capacitance matrix that changes amid the simulation, while all other sub-matrices remain constant. The matrix C_{vv} captures the drain, gate and bulk capacitances of all devices, which are voltage-dependent, while C_{cc} , and C_{cv} are the capacitance matrices that arise from interconnects and are hence constant.

For typical interconnect structures, the above decomposition allows us to manipulate the MNA equations (2) and (3):

$$\begin{aligned} & \underbrace{\left(\frac{L}{h} + \frac{R}{2} + \frac{h}{4} A_1 P_{cc} A_1^T \right)}_X i_l^{k+1} \\ &= \underbrace{\left(\frac{L}{h} - \frac{R}{2} - \frac{h}{4} A_1 P_{cc} A_1^T \right)}_Y i_l^k \\ &+ A_1 v_c^k + \frac{h}{4} A_1 P_{cc} A_{i1}^T (I_s^{k+1} + I_s^k) \\ &- A_1 P_{cc} C_{cv} (v_v^{k+1} - v_v^k) + \frac{A_2}{2} (v_v^{k+1} + v_v^k), \end{aligned} \quad (4)$$

$$\begin{aligned} v_c^{k+1} &= v_c^k - \frac{h}{2} P_{cc} A_2^T (i_l^{k+1} + i_l^k) + \frac{h}{2} P_{cc} A_{i1}^T (I_s^{k+1} + I_s^k) \\ &- P_{cc} C_{cv} (v_v^{k+1} - v_v^k), \end{aligned} \quad (5)$$

$$\begin{aligned} C_{vv} v_v^{k+1} &= C_{vv} v_v^k - \frac{h}{2} A_2^T (i_l^{k+1} + i_l^k) + \frac{h}{2} A_{i2}^T (I_s^{k+1} + I_s^k) \\ &- C_{vc} (v_c^{k+1} - v_c^k) + \frac{h}{2} (I_v^{k+1} + I_v^k), \end{aligned} \quad (6)$$

$$I_v^{k+1} = f(v_v^{k+1}). \quad (7)$$

Here r denotes the size of interconnect structure connected directly to non linear circuit, and given $l = N - r$ we note that

$$C_{cc} \in \mathbf{R}^{l \times l}, \quad C_{vv} \in \mathbf{R}^{r \times r}.$$

$P_{cc} = C_{cc}^{-1}$ is the inverse capacitance matrix, and A is the adjacency matrix of the circuit. A is obtained by first adding A_g and A_l and then removing zero columns (these correspond to intermediate nodes, representing the connection of a resistance to an inductance).

The development thus far is similar to that in [7], with the major difference being the addition of (6) and (7), which account for the nonlinear elements. The main contribution in [7] was the fast solution of (4) and (5), where all matrices are constant over the simulation period. We will show in §4.2 that the techniques in [7] can be extended to handle the case when nonlinear elements are present.

For future reference, we will call the technique of directly solving (4), (5), (6), and (7) as the ‘‘Exact-RLP’’ algorithm. It can be shown that the computational complexity of the Exact-RLP algorithm is $O(l^3 + pq(l^2 + r^3))$. For large VLSI interconnect structures we have $l \gg r$, reducing the complexity to $O(l^3 + pq(l^2))$.

4 Computationally efficient implementation

We now turn to the fast solution of equations (4) through (7). Recall that the nonlinear equation (7) is handled via the Newton-Raphson technique. This requires, at each time step, linearizing (7) and sub-

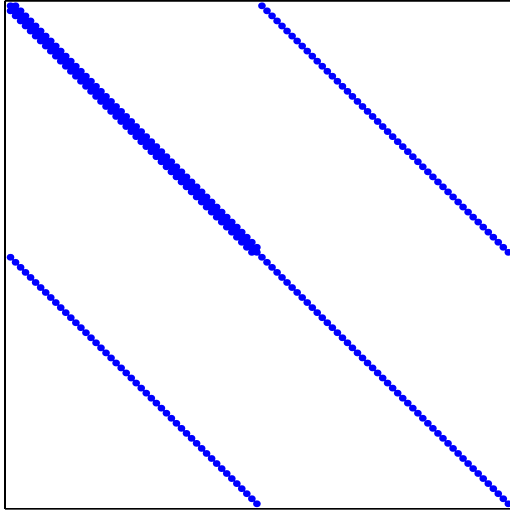


Figure 1: Sparsity structure of A . The nonzero entries are shown darker.

stituting it into (6). The resulting set of linear equations have very specific structure:

- Equations (4) and (5) are of the form $Ax = b$ where A is fixed (does not change with the time-step). Moreover, A^{-1} is typically approximately sparse (For details, see §4.2).
- Equation (6) (after the substitution of the linearized (7)) is again of the form $Ax = b$, where the matrix A is obtained by adding C_{vv} and the coefficient of the first-order terms in the linearized equation (7). Recall that the matrix C_{vv} captures the drain, gate and bulk capacitances of all devices. It also contains the interconnect coupling capacitances between gates and drains of different non-linear devices in the circuit. As each non-linear device is connected to only a few nodes and the capacitive effects of interconnects are localized, *the A matrix is observed to be sparse in practice* (For details, see §4.1). Note that A changes with each Newton-Raphson iteration and with the time-step.

Thus the key computational problem is the solution of a sparse time-varying set of linear equations, coupled with a large fixed system of linear equations $Ax = b$ with A^{-1} being sparse.

4.1 Solving sparse time-varying linear equations

Krylov subspace methods have been shown to work extremely well for sparse time-varying linear equations [6]. Specifically, the GMRES (Generalized Minimum Residual) method of Saad and Schultz [13] allows the efficient solution of a sparse, possibly non-symmetric, linear system to within a pre-specified tolerance. This method performs a directional search along the orthogonal Arnoldi vectors which span the Krylov subspace of A . That is, given an initial guess x_0 and corresponding residual $r_0 = b - Ax_0$, orthogonal vectors $\{q_1, q_2, \dots, q_m\}$ are generated with the property that they span S_m , the solution search space at iteration m .

$$\begin{aligned} S_m &= x_0 + \text{span}\{r_0, Ar_0, \dots, A^m r_0\} \\ &= x_0 + \kappa(A, r_0, m) \\ &\subseteq \text{span}\{q_1, q_2, \dots, q_m\}. \end{aligned} \quad (8)$$

These vectors are chosen according to the Arnoldi iteration: $AQ_m = Q_{m+1}H_m$ where $Q_m = \{q_1, q_2, \dots, q_m\}$ is orthogonal and $H_m \in \mathbf{R}^{(m+1) \times m}$

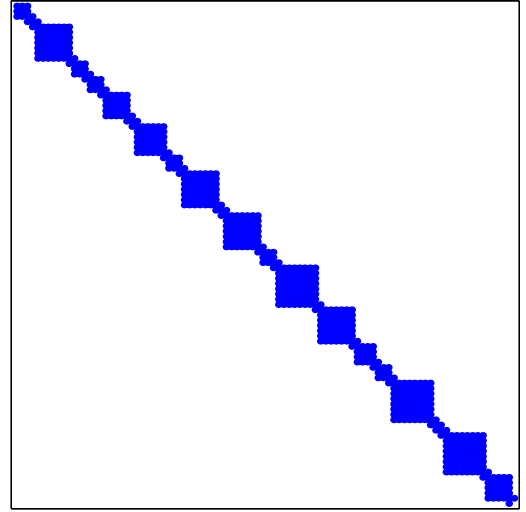


Figure 2: Sparsity structure of A . The non-zero entries are shown darker.

is an upper Heisenberg matrix.

For these methods the choice of a preconditioner matrix M , which is an approximation of A , can greatly affect the convergence. A good preconditioner should have the following two properties:

- $M^{-1}A \approx I$.
- It must accommodate a fast solution to an equation of the form $Mz = c$ for a general c .

Figure 1 depicts the sparsity structure of the A matrix for a circuit example of parallel wires driving a bank of inverters. For such a sparsity structure, an appropriate choice of the preconditioner could be of the form as shown in Figure 3. Although we have chosen a circuit with only inverters for simplicity, a more complicated circuit structure would simply distribute the entries around the diagonal and off-diagonal bands and lead to possibly more off diagonal bands. To see this, consider an extreme case where the circuit under consideration has only non-linear devices and does not comprise of interconnects. In this case the sparsity pattern of the A matrix is as shown in Figure 2. Therefore, the chosen preconditioner would encompass not only the sparsity structure shown in Figure 1 but also other sparsity patterns that might arise with the analysis of more complicated non-linear devices. Correspondingly the structure of the preconditioner (see Figure 3) would have additional bands.

Matrices of the form shown in Figure 3 have the following two properties which make them an ideal choice for preconditioner.

- The inverses of the preconditioner matrix can be computed efficiently in linear time, $O(r)$ (r denotes the size of interconnect structure directly connected to non-linear devices), by exploiting the Hadamard product formulation as shown in [10].
- It can also be shown that this formulation facilitates the fast matrix-vector products, again in linear time ($O(r)$), which arise while solving linear systems of equations with the preconditioner matrix.

A simple example which best illustrates these advantages is a

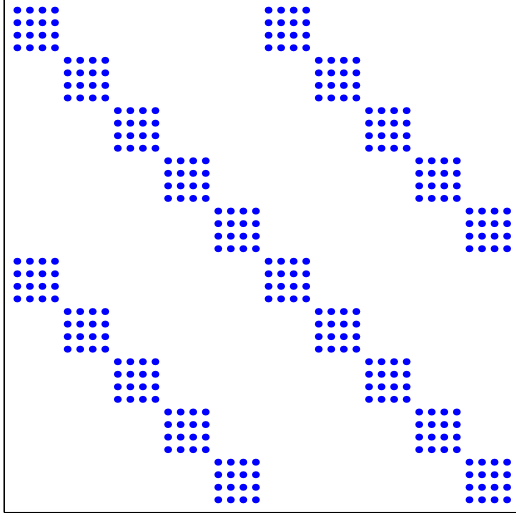


Figure 3: Preconditioner matrix.

symmetric tridiagonal matrix.

$$B = \begin{pmatrix} a_1 & -b_1 & & & \\ -b_1 & a_2 & & & \\ & & \ddots & & \\ & & & -b_{n-2} & a_{n-1} & -b_{n-1} \\ & & & & -b_{n-1} & a_n \end{pmatrix} \quad (9)$$

The inverse of B can be represented compactly as a Hadamard product of two matrices, which are defined as follows:

$$B^{-1} = \underbrace{\begin{pmatrix} u_1 & u_1 & \cdots & u_1 \\ u_1 & u_2 & \cdots & u_2 \\ \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & \cdots & u_n \end{pmatrix}}_U \circ \underbrace{\begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_2 & v_2 & \cdots & v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n & v_n & \cdots & v_n \end{pmatrix}}_V. \quad (10)$$

There exists an explicit formula to compute the sequences $\{u_i\}, \{v_i\}$ efficiently in $O(n)$ operations which is detailed in [10]. In this case, if we are interested in solving a linear system of equations $By = c$, we only need to concern ourselves with the matrix-vector product $B^{-1}c = y$. This computation can also be performed efficiently in $O(n)$ computations as outlined below:

$$\begin{aligned} P_{u_i} &= \sum_{j=1}^i u_j c_j, & P_{v_i} &= \sum_{j=i}^n v_j c_j, & i &= 1, \dots, n, \\ y_1 &= u_1 P_{v_1}, \\ y_i &= v_i P_{u_{i-1}} + u_i P_{v_i}, & i &= 2, \dots, n. \end{aligned} \quad (11)$$

The above formulation for a tridiagonal matrix could be easily extended to handle the more general case when the preconditioner matrix is a zero padded block tridiagonal matrix (matrix with zero diagonals inserted between the main diagonal and the non-zero super-diagonal and sub-diagonal of tridiagonal matrix) as in Figure 3. Elementary row and column block permutations could be performed on such a matrix to reduce it into a block tridiagonal matrix. This has

been shown with a small example as below.

$$B = \begin{pmatrix} a_1 & 0 & -b_1 & 0 \\ 0 & a_2 & 0 & -b_2 \\ -b_1 & 0 & a_3 & 0 \\ 0 & -b_2 & 0 & a_4 \end{pmatrix} \quad (12)$$

$$= P \underbrace{\begin{pmatrix} a_1 & -b_1 & 0 & 0 \\ -b_1 & a_2 & 0 & 0 \\ 0 & 0 & a_3 & -b_2 \\ 0 & 0 & -b_2 & a_4 \end{pmatrix}}_X P^T, \quad (13)$$

where

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Hence

$$\begin{aligned} B^{-1} &= P X^{-1} P^T \\ &= \underbrace{\begin{pmatrix} u_1 & 0 & u_1 & 0 \\ 0 & u_2 & 0 & u_2 \\ u_1 & 0 & u_3 & 0 \\ 0 & u_2 & 0 & u_4 \end{pmatrix}}_U \circ \underbrace{\begin{pmatrix} v_1 & 0 & v_3 & 0 \\ 0 & v_2 & 0 & v_4 \\ v_3 & 0 & v_3 & 0 \\ 0 & v_4 & 0 & v_4 \end{pmatrix}}_V. \end{aligned} \quad (14)$$

We have not included block matrices for simplicity of presentation, however the zero padded block tridiagonal case is a natural extension of the above example. All the entries in U, V matrices have to be now replaced by blocks and accordingly the row and column permutations would be replaced by their block counterparts with an identity matrix of appropriate size replacing the 'ones' in the P matrix. Table 1 gives the comparison for Incomplete-LU preconditioner and the zero-padded (Z-Pad) preconditioner. Simulations were done on circuits consisting of busses with parallel conductors driving bank of inverters. 'Size' denotes the number of non-linear devices. All the results are reported as a ratio of run-time and iteration-count (number of iterations for the solution to converge to within a tolerance of $1e-10$) of Z-Pad to the Incomplete-LU preconditioner. As can be seen from Table 1, Z-Pad offers a substantial improvement in run time as compared to the Incomplete-LU preconditioner.

Size	400	800	1600	3200
Runtime	.44	.42	.42	.43
Iterations	5/10	5/10	5/10	5/10

Table 1: Preconditioner comparison.

4.2 Solving $Ax = b$ with a constant, approximately sparse A^{-1}

We now turn to the solution of equations (4) and (5). As mentioned earlier, these equations reduce to the form $Ax = b$ with a constant, approximately sparse A^{-1} . A (corresponding to X in (4)) is composed of L, R and P . Each of these matrices has a sparse inverse for typical VLSI interconnects which then leads to an approximately sparse A^{-1} (Note that this argument is used for motivating the sparsity inherent in A^{-1} and cannot be used as a theoretical proof for the same). In addition this sparsity has a regular pattern which can be explained on the basis of how inductance and capacitance matrices are extracted. The distributed RLC effects of VLSI interconnects can be modeled by dividing conductors into small subsets of segments,

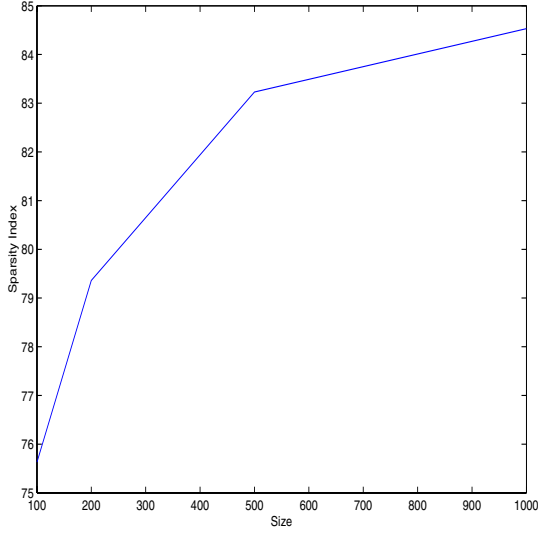


Figure 4: Average sparsity versus circuit size.

each of which are aligned [9, 3]. Each of these subsets leads to a sparsity pattern (corresponding to a band in A^{-1}). All the effects when summed up lead to a A^{-1} matrix that has a regular sparsity pattern. Window selection algorithm as described in [16, 3] could then be employed to find out the sparsity pattern in A^{-1} . It has been recognized in earlier work that this property (sparsity) yields enormous computational savings; it has been shown in [7] that an approximate implementation of the Exact-RLP algorithm, referred to simply as the “RLP algorithm” provides an order-of-magnitude in computational savings with little sacrifice in simulation accuracy.

To proceed, we rewrite (4) and (5) as

$$i_l^{k+1} = X^{-1}Y_l^k + X^{-1}A_1v_c^k + \frac{h}{4}X^{-1}A_1P_{cc}A_{il}^T (I_s^{k+1} + I_s^k) - X^{-1}A_1P_{cc}C_{cv} (v_v^{k+1} - v_v^k) + \frac{A_2}{2} (v_v^{k+1} + v_v^k), \quad (15)$$

$$X^{-1}A_1v_c^{k+1} = X^{-1}A_1v_c^k - X^{-1}A_1\frac{h}{2}P_{cc}A_2^T (i_l^{k+1} + i_l^k) + \frac{h}{2}X^{-1}A_1P_{cc}A_{il}^T (I_s^{k+1} + I_s^k) - X^{-1}A_1P_{cc}C_{cv} (v_v^{k+1} - v_v^k). \quad (16)$$

Although X is a dense matrix, X^{-1} turns out to be an approximately sparse matrix. Moreover the matrices $X^{-1}Y$, $X^{-1}A_1$, $X^{-1}A_1P_{cc}A_{il}^T$, $X^{-1}A_1P_{cc}C_{cv}$ are also approximately sparse [7]. This information can be used to reduce the computation significantly by noting that each step of trapezoidal integration now requires only sparse vector multiplications. Solving sparse (15) and (16) along with (6) and (7) is termed as the RLP algorithm (SASIMI). To analyze the computational saving of the approximate algorithm over the Exact-RLP algorithm, we denote “sparsity index” of a matrix A as ratio of the number of entries of A with absolute value less than ϵ to the total number of entries. The computation required for each iteration of (15) and (16) is then $O((1-v)l^2)$, where v is the minimum of the sparsity indices the matrices $X^{-1}Y$, $X^{-1}A_1$, $X^{-1}A_1P_{cc}A_{il}^T$, $X^{-1}A_1P_{cc}C_{cv}$. Figure 4 provides the average sparsity for the matrices for a system with parallel conductors driving a bank of inverters. The sizes in consideration are 100, 200, 500 and 1000. On top of this the computation time of X^{-1} can be reduced to $O(l)$ by using the windowing techniques (details in [16]). Hence the computational complexity of RLP is $O(pq(1-v)l^2)$ as compared to $O(pqn^3)$ for the MNA approach.

σ	$\rho=5$	$\rho=20$	$\rho=50$
100	.0054	.0053	.0088
200	.0078	.0052	.0071
500	.0006	.0022	.0001
1000	.0003	.0005	.0004
2000	.0003	.0004	.0004

Table 2: RMSE comparison.

5 Numerical results and conclusions

We implemented the Exact-RLP and RLP (SASIMI) algorithms in C++. A commercially available version of SPICE with significant speed-up over the public-domain SPICE has been used for reporting all results with SPICE. Simulations were done on circuits consisting of busses with parallel conductors driving bank of inverters, with wires of length 1mm, cross section $1\mu\text{m} \times 1\mu\text{m}$, and with a wire separation of $1\mu\text{m}$. A periodic 1V square wave with rise and fall times of 6ps each was applied to the first signal with a time period of 240ps. All the other lines were assumed to be quiet. For each wire, the drive resistance was 10Ω . A time step of 0.15ps was taken and the simulation was performed over 30 ps (or 200 time steps). For the inverters the W/L ratio of NMOS and PMOS were taken to be $.42\mu\text{m}/.25\mu\text{m}$ and $1.26\mu\text{m}/.25\mu\text{m}$ respectively.

In order to explore the effect of the number of non-linear elements relative to the total, three cases were considered. With ρ denoting the ratio of the number of linear elements to that of non-linear elements, the experiments were performed for ρ equaling 5, 20 and 50. The number of linear elements in the following results is denoted by σ .

We first present results comparing the accuracy in simulating the voltage waveforms at the far end of the first line (after the inverter load). The metric for comparing the simulations is the relative mean square error (RMSE) defined as

$$\frac{\sum_i (v_i - \tilde{v}_i)^2}{\sum_i v_i^2}$$

where v and \tilde{v} denote the waveforms obtained from Exact-RLP and SASIMI respectively.

Table 2 presents a summary of the results from the study of simulation accuracy. It can be seen that the simulation accuracy of the Exact-RLP algorithm is almost identical to that of SPICE, while the SASIMI has a marginally inferior performance as measured by the RMSE. The error values for SASIMI are compared simply with the Exact-RLP as it had the same accuracy as SPICE results for all the experiments run. A plot of the voltage waveforms at the far end of the active line, obtained from SPICE, Exact-RLP and SASIMI algorithms, is shown in Figure 5. (The number of conductors in this simulation example is 200.) There is almost no detectable simulation error between the SASIMI, Exact-RLP and SPICE waveforms over 200 time steps. To give a better picture, the accuracy results reported are for a larger simulation time of 2200 time steps.

We now turn to a comparison of the computational requirements between Exact-RLP, SASIMI and SPICE. Table 3 summarizes the findings. For a fair comparison our total simulation time is compared against the transient simulation time for SPICE (i.e we have not included any of the error check or set up time for SPICE). As can be seen from the table, SASIMI outperforms the Exact-RLP algorithm and SPICE. For the case of 500 conductors with $\rho = 50$, the Exact-RLP algorithm is 390 times as fast compared to SPICE. SASIMI is about 1400 times faster as compared to SPICE, and more than three times faster than Exact-RLP. As can be seen, the computational savings increase as the ratio of linear to non-linear elements is

σ	$\rho=5$			$\rho=20$			$\rho=50$		
	SPICE	Exact-RLP	SASIMI	SPICE	Exact-RLP	SASIMI	SPICE	Exact-RLP	SASIMI
100	11.96	1.34	1.26	13.73	.27	.21	13.54	.15	.12
200	100.25	3.28	2.68	68.72	.64	.28	67.68	.55	.22
500	3590.12	17.13	4.872	1919.21	13.47	3.01	1790.67	4.58	1.30
1000	>12hrs	87.75	22.71	>10hrs	79.07	16.49	>10hrs	77.56	15.20
2000	> 1day	545.6	78.06	> 1day	526.23	59.33	> 1day	408.54	56.05

Table 3: Run time (in seconds) comparisons.

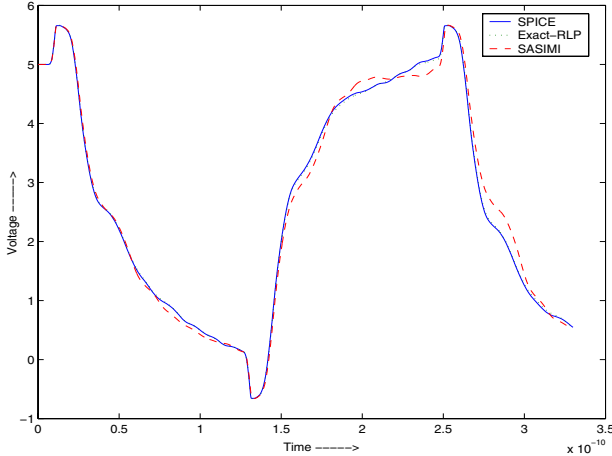


Figure 5: The voltage waveforms obtained through SPICE, Exact-RLP and SASIMI.

increased from 5 to 50. The savings also increase with increase in the size of the problem considered. The computational efficiency of the SASIMI can be explained on the use of sparsity-aware algorithms for both the linear and non-linear parts of the problem.

6 Acknowledgments

This material is based on work supported by the NASA, under Award NCC 2-1363, and by National Science Foundation under Award CCR-9984553 and CCR-0203362.

References

- [1] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.
- [2] M. Beattie and L. Pileggi. Modeling magnetic coupling for on-chip interconnect. In *Proc. Design Automation Conf.*, pages 335–340, 2001.
- [3] T. H. Chen, C. Luk, H. Kim, and C. C.-P. Chen. INDUCTWISE: Inductance-wise interconnect simulator and extractor. In *Proc. Int. Conf. on Computer Aided Design*, pages 215–220, 2002.
- [4] T.-H. Chen, J.-L. Tsai, C. C.-P. Chen, and T. Karnik. HISIM: Hierarchical interconnect-centric circuit simulator. In *Proc. Int. Conf. on Computer Aided Design*, pages 489–496, 2004.
- [5] A. Devgan, H. Ji, and W. Dai. How to efficiently capture on-chip inductance effects: Introducing a new circuit element K. In *Proc. Int. Conf. on Computer Aided Design*, pages 150–155, 2000.
- [6] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1996.
- [7] J. Jain, C.-K. Koh, and V. Balakrishnan. Fast simulation of VLSI interconnects. In *Proc. Int. Conf. on Computer Aided Design*, pages 93–98, 2004.
- [8] H. Ji, Q. Yu, and W. Dai. SPICE compatible circuit models for partial reluctance K. In *Proc. Asia South Pacific Design Automation Conf.*, pages 786–791, 2004.
- [9] T. Lin, M. W. Beattie, and L. T. Pileggi. On the efficacy of simplified 2D on-chip inductance. In *Proc. Design Automation Conf.*, pages 757–762, 2002.
- [10] R. Nabben. Decay rates of the inverses of nonsymmetric tridiagonal and band matrices. *SIAM Journal on Matrix Analysis and Applications*, 20(3):820–837, May 1999.
- [11] L. W. Nagel. SPICE2: A computer program to simulate semiconductor circuits. Technical report, U.C. Berkeley, ERL Memo ERL-M520, 1975.
- [12] A. Pacelli. A local circuit topology for inductive parasitics. In *Proc. Int. Conf. on Computer Aided Design*, pages 208–214, 2002.
- [13] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM Journal on Scientific Computing*, pages 856–869, 1986.
- [14] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw. Hierarchical analysis of power distribution networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 159–168, February 2002.
- [15] H. Zheng, B. Krauter, M. Beattie, and L. Pileggi. Window-based susceptance models for large scale RLC circuit analyses. In *Proc. Design Automation and Test in Europe Conf.*, pages 628–633, 2002.
- [16] G. Zhong, C.-K. Koh, and K. Roy. On-chip interconnect modeling by wire duplication. In *Proc. Int. Conf. on Computer Aided Design*, pages 341–346, 2002.