# A Quasi-Newton Preconditioned Newton-Krylov Method for Robust and Efficient Time-Domain Simulation of Integrated Circuits with Strong Parasitic Couplings*

Zhao Li† and C.-J. Richard Shi

Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA
{lz2000, cjshi}@ee.washington.edu

**Abstract:** In this paper, the Newton-Krylov method is explored for robust and efficient time-domain VLSI circuit simulation. Different from the LU-factorization based direct method, the Newton-Krylov method uses a preconditioned Krylov-subspace iterative method for linear system solving. Our key contribution is to introduce an effective quasi-Newton preconditioning scheme for Krylov-subspace methods to reduce the number and cost of LU factorizations during time-domain circuit simulation. Experimental results on a collection of digital, analog and RF circuits have shown that the quasi-Newton preconditioned Krylov-subspace method is as robust and accurate as SPICE3. The proposed Newton-Krylov method is especially attractive for simulating circuits with a large amount of parasitic RLC elements for post-layout verification.

## 1. Introduction

In modern deep-submicron meter very-large-scale integrated (VLSI) circuits, parasitic effects are no longer ignorable with the higher operation frequency, lower supply voltage and smaller device feature size. It is desirable, especially for sensitive analog and mixed-signal circuit design, to perform full-chip time-domain simulation of massively parasitic-coupled systems [14] before tape-out using SPICE-like circuit simulators [12]. This is, however, an extremely time-consuming task, since SPICE-like circuit simulators use LU factorization based *direct* methods for solving systems of nonlinear differential equations. For massively parasitic coupled systems, the complexity of LU factorization is approaching its worst case $O(n^3)$, where $n$ is the size of a circuit, instead of its average case $O(n^{1.1\text{-}1.5})$ [12].

Existing work to reduce the cost of LU factorization for circuit simulation can be classified into three categories. The first category is so-called relaxation-based methods [15], including relaxation methods for nonlinear iteration, such as Gauss-Jacobi, Gauss-Seidel, Successive Over-Relaxation (SOR), etc. [17]. In [2], the preconditioner for the Gauss-Seidel method has been carefully studied to improve its robustness, which leads to the partial Gauss-Seidel (PGS) method. For time-domain simulation, semi-implicit integration methods [20], including alternating-direction-implicit methods [17], have been applied to substrate analysis [14] and power/ground network analysis [10]. The drawback of relaxation-based methods is that their stability and convergence properties strongly depend on circuit structures.

The second category, which has been explored extensively since the inception of SPICE, belongs to a class of methods known as quasi-Newton methods [5]. The purpose of quasi-Newton methods is to reduce the number of LU factorizations for circuit simulation. For numerical integration, several fixed leading coefficient integration methods [4][7][11] have been proposed for variable time step-size integration to keep the circuit matrix constant – thus to have a less number of LU factorizations. For nonlinear iteration, virtually all SPICE-like circuit simulators explore quasi-Newton methods to reduce the number of LU factorizations [1][11], also known as Jacobian matrix bypass. However, it is well known that the convergence rate of quasi-Newton methods for nonlinear iteration can degrade to linear.

The third category is to apply Krylov-subspace based iterative methods [17] to solve the system of linearized equations in the inner Newton-Raphson iteration, so called the Newton-Krylov method [9]. The Newton-Krylov method can have the same stability and convergence properties as the classical Newton-Raphson method provided that the linearized system can be solved accurately by Krylov-subspace methods. The key issue is how to construct an effective preconditioner for iterative methods. For special applications such as in the shooting-Newton method and harmonic balance analysis for RF circuit simulation [18], where a circuit matrix is generally block-diagonally dominant, a preconditioner can be well constructed. Similarly, Krylov-subspace methods are applied for model order reduction of interconnect lines [13], substrates [8], and power/ground networks and direct analysis of well-structured large-scale linear circuits, such as substrate [14] and power/ground networks [3].

In this paper, we explore Krylov-subspace methods for time-domain simulation of general nonlinear circuits. The basic idea of our approach is to reuse the LU factorized matrices from the previous time point/nonlinear iteration as the preconditioner for Krylov-subspace methods. With this, the number and cost of LU factorizations can be reduced dramatically. Additionally, the stability and convergence properties will not be compromised, thanks to Krylov-subspace methods. The Texas Instrument research team explored a similar idea using a conjugate gradient squared method with a partial LU preconditioner, called PLUCGS [2]. However, it was concluded in [2] that the PLUCGS method was even less efficient than the PGS method, a variant of the Gauss-Seidel method.

The main contribution of our work is a systematic method to construct effective preconditioners based on quasi-Newton principles to perform as few LU factorizations as possible during the whole time domain nonlinear circuit simulation. Our implementation is as robust as LU factorization based direct methods, and can be orders of magnitude faster than SPICE for time-domain simulation of VLSI circuits with a large amount of linear parasitic elements.

Specifically, the efficiency comes from the following four ideas:

1. The preconditioner is kept constant when time step-sizes vary within a predefined range.
2. We propose a generalized way to partition the entire operating region of nonlinear devices into piecewise weakly nonlinear (PWNL) regions. With this, the preconditioner is kept constant if all nonlinear devices reside in their present operating PWNL regions.
3. When nonlinear devices switch their operating PWNL regions during nonlinear iteration, the low-rank update technique [6] is applied to update the preconditioner efficiently rather than performing new LU factorization.
4. We further explore incomplete LU preconditioners derived from factorized full L and U matrices for the best efficiency.

This paper is organized as follows. In Section 2, we provide an overview of quasi-Newton methods and Krylov-subspace methods. Section 3 introduces Newton-Krylov based time-domain nonlinear circuit simulation with the quasi-Newton preconditioning scheme.

* This research was supported by the DARPA NeoCAD Program under Grant No. N66001-01-8920, an NSF CAREER Award under Grant No. 9985507, and the SRC/NSF Mixed-Signal Initiative under Contract No. CCR0120371.

† Zhao Li is now with Cadence Design Systems, Inc. (zhaoli@cadence.com)

0-7803-9451-8/06/$20.00 ©2006 IEEE.

Section 4 describes the proposed time-domain nonlinear circuit simulation flow. Experimental results on general nonlinear circuits and power/ground network examples are reported in Section 5. Section 6 concludes the paper.

## 2. Iterative Methods for Time-Domain Circuit Simulation

LU factorization based direct methods are efficient for small-scale to medium-scale circuit simulation. However, the cost of LU factorization is becoming the dominant per-iteration cost for large-scale circuit simulation incorporating parasitic effects [11][14]. To tackle this problem and to continue exploiting the robustness of LU factorization, a key idea is to reuse the previous LU factorization to solve circuit matrix equations $\boldsymbol{Ax=b}$ for as many time points and/or nonlinear iteration steps as possible. This leads to two categories of iterative methods – quasi-Newton methods and Krylov-subspace methods.

### 2.1 Quasi-Newton methods

Suppose that we have a LU factorized matrix $\boldsymbol{M}$, which is considered to be close enough to the circuit matrix $\boldsymbol{A}$, circuit matrix equations $\boldsymbol{Ax=b}$ can be solved by Eq. (1) derived from the first-order Taylor expansion with the matrix $\boldsymbol{M}$ as the approximate Jacobian matrix (representing the first-order derivatives).

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k-1)}) \quad (1)$$

For nonlinear circuits, Eq. (1) is further written as follows,

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}(\boldsymbol{b}^{(k-1)} - \boldsymbol{A}^{(k-1)}\boldsymbol{x}^{(k-1)}) = \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}(-\boldsymbol{f}^{(k-1)}) \quad (2)$$

where $\boldsymbol{f}$ is the vector contributed by input sources, nonlinear devices, and numerical integration of charge/flux storage devices. It should be noted that Eq. (2) will reduce to the Newton-Raphson method if $\boldsymbol{M=A}$.

It should be noted that the search direction with quasi-Newton methods is $\boldsymbol{M}^{-1}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k-1)})$ for each nonlinear iteration step.

### 2.2 Krylov-subspace methods

Given an initial guess $\boldsymbol{x}^{(0)}$ to the circuit matrix equation $\boldsymbol{Ax=b}$, Krylov-subspace methods seek an approximate solution $\boldsymbol{x}^{(m)}$ from the subspace of $\boldsymbol{x}^{(0)}+\mathbf{K}_m(\boldsymbol{A}, \boldsymbol{x}^{(0)})$ by imposing the Petrov-Galerkin condition [17],

$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(m)} \perp \mathbf{L}_m(\boldsymbol{A}, \boldsymbol{x}^{(0)}) \quad (3)$$

where $\mathbf{K}_m(\boldsymbol{A}, \boldsymbol{x}^{(0)}) = span\{\boldsymbol{r}^{(0)}, \boldsymbol{A}\boldsymbol{r}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}^{(0)}\}$, $\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}$, and $\mathbf{L}_m(\boldsymbol{A}, \boldsymbol{x}^{(0)})$ is a subspace of dimension $m$.

It is well known that a preconditioner [17] (or a preconditioning matrix) $\boldsymbol{M}$ is the key to the fast convergence of Krylov-subspace methods. The purpose of a preconditioner is to make the preconditioned matrix $\boldsymbol{M}^{-1}\boldsymbol{A}$ as close to the identity matrix as possible. With left-preconditioned Krylov-subspace methods, circuit matrix equations to be solved become $\boldsymbol{M}^{-1}\boldsymbol{Ax} = \boldsymbol{M}^{-1}\boldsymbol{b}$ and the Krylov subspace $\mathbf{K}_m$ is defined as follows,

$$\mathbf{K}_m = span\{\boldsymbol{r}^{(0)}, \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{r}^{(0)}, (\boldsymbol{M}^{-1}\boldsymbol{A})^2\boldsymbol{r}^{(0)}, \ldots, (\boldsymbol{M}^{-1}\boldsymbol{A})^{m-1}\boldsymbol{r}^{(0)}\} \quad (4)$$

where $\boldsymbol{r}^{(0)} = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)})$. It is not surprising that the effect of the preconditioner $M$ on preconditioned Krylov-subspace methods is similar to that of the approximate Jacobian matrix $M$ on quasi-Newton methods.

The advantage of preconditioned Krylov-subspace methods over quasi-Newton methods is that, for each nonlinear iteration step, an orthogonal Krylov subspace $K_m$ is used for constructing the search direction, rather than only one single search direction $\boldsymbol{M}^{-1}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k-1)})$ as in quasi-Newton methods. As the result, the search direction of the Newton-Raphson method could be well approximated with the Newton-Krylov method.

We choose to use the flexible GMRES (FGMRES) method [16], an extension of the original right-preconditioned GMRES method [17], to solve the linearized circuit equations $(\boldsymbol{AM}^{-1})(\boldsymbol{Mx})=\boldsymbol{b}$.

## 3. Quasi-Newton Preconditioner Construction

In this section, we present a systematic method to construct effective preconditioners based on quasi-Newton methods. Section 3.1 presents adaptive time-step size control for preconditioner computation. Section 3.2 describes the generation of generalized PWNL definition of nonlinear devices**.**

### 3.1 Quasi-Newton preconditioners by adaptive time step-size control

Suppose that $h$ is the base time-step size, and $h_n$ is the current time-step size. To develop a guideline for adaptive time step-size control for preconditioner computation, let us write the system of linearized circuit equations as:

$$\boldsymbol{G}\boldsymbol{x} + \boldsymbol{C}\dot{\boldsymbol{x}} = \boldsymbol{b} \quad (5)$$

where $\boldsymbol{G}$ and $\boldsymbol{C}$ represent the conductance and susceptance (capacitance) matrices, and $\boldsymbol{b}$ is the vector due to input sources and nonlinear devices. Replace time derivatives by the standard trapezoid formula, we have

$$\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{\alpha h}\right)\boldsymbol{x}_n^{(k)} = \frac{2\boldsymbol{C}}{\alpha h}\boldsymbol{x}_{n-1} + \boldsymbol{C}\dot{\boldsymbol{x}}_{n-1} + \boldsymbol{b} \quad (6)$$

where $h_n = \alpha h$. To solve the above equation with preconditioned Krylov-subspace methods, the preconditioner we use is chosen to be $\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{h}\right)$, which should be as close to $\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{\alpha h}\right)$ as possible. Therefore, we introduce a parameter $0 < \eta < 1$ so that the preconditioner should satisfy the following inequality

$$\left\| \left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{h}\right)^{-1}\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{\alpha h}\right) - \boldsymbol{I} \right\| = \left\| \left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{h}\right)^{-1}\left(1 - \frac{1}{\alpha}\right)\frac{2\boldsymbol{C}}{h} \right\| < \eta < 1 \quad (7)$$

where $\|\bullet\|$ represents the spectral radius of the matrix. The above inequality can be re-written as

$$\left| \frac{1 - 1/\alpha}{1 - z} \right| < \eta < 1 \quad (8)$$

where $z = -h/(2\tau)$ and $\tau$ is an eigenvalue of the matrix $\boldsymbol{G}^{-1}\boldsymbol{C}$. Let us refer to the region defined by the above inequality as the effective preconditioner region. To ensure the effective preconditioner region to include the left half of the complex-$z$ plane for covering all poles of a decaying system, we always choose $\eta = |1-1/\alpha| < 1$ so that the effective preconditioner region is $|z-1| > \frac{|1-1/\alpha|}{\eta} = 1$. Then we can draw the effective preconditioner region in the complex-z plane as in Fig. 1, in which the black region represents the effective preconditioner region.

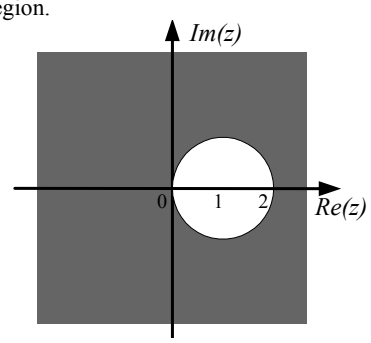


**Figure 1. The effective preconditioner region.**

A large range of $\alpha$ is helpful to reduce the number of LU factorizations when time steps vary. However, the preconditioner $\boldsymbol{M}^{-1}$ will diverge from $\boldsymbol{A}^{-1}$ when $\alpha$ is far away from 1. This will unfortunately increase the cost of Krylov-subspace methods. In our implementation, $0.625 < \alpha < 2.5$ is used for a tradeoff. We note that

the effective preconditioner region is similar to the convergence region for the iterative integration formulae in SILCA [11].

### 3.2 Quasi-Newton preconditioners by piecewise weakly nonlinear definition of nonlinear devices

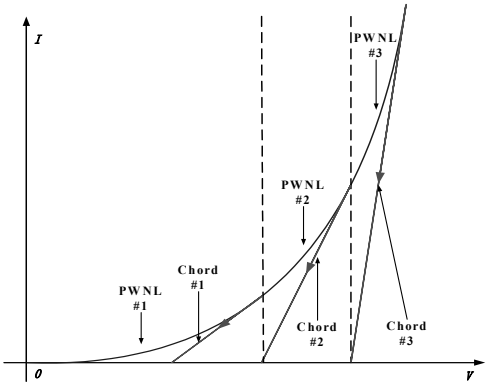


**Figure 2. The PWNL definition of a nonlinear function.**

In this sub-section, we present a systematic method for quasi-Newton preconditioner computation during nonlinear iteration. The central idea is to partition any nonlinear device function into a collection of regions, where in each region the nonlinear function is a weakly nonlinear function, i.e., its derivatives can be approximated by a constant (called a chord). As an example, Figure 2 shows an example of the PWNL definition of a nonlinear function, where three PWNL regions are defined with three different chords (fixed first-order derivatives).

Now consider how to generate PWNL regions automatically for arbitrary nonlinear functions. Suppose that nonlinear iteration is performed within a PWNL region of a nonlinear function $f(x)$ to solve $f(x)=0$, the nonlinear iteration equation can be expressed by,

$$x_{i+1} = x_i - \frac{f(x_i)}{g} \tag{9}$$

where $g$ is the chord for this PWNL region. Let the exact solution be $x^* = x_i + \varepsilon_i = x_{i+1} + \varepsilon_{i+1}$. Subtracting $x^*$ from the both sides of Eq. (9) gives

$$\varepsilon_{i+1} = \varepsilon_i + \frac{f(x_i)}{g} \tag{10}$$

After applying the Taylor expansion on $f(x)$ at $x_i$, we obtain the following error estimation,

$$\varepsilon_{i+1} \approx \varepsilon_i \left(1 - \frac{f'(x_i)}{g}\right) - \varepsilon_i^2 \frac{f''(x_i)}{2g} \tag{11}$$

From Eq. (11), if $g$ is always equal to $f'(x_i)$, as in the Newton-Raphson method, the convergence rate is quadratic. The smaller the $|1- f'(x_i)/g|$ is, the closer to the quadratic convergence rate Eq. (11) is. On the other hand, the larger the $|1- f'(x_i)/g|$ is, the larger the range of a PWNL region could be. In practice, we define the following condition with a parameter $0<\delta<1$,

$$\left|1 - \frac{f'(x_i)}{g}\right| < \delta \tag{12}$$

In our implementation, for simplicity, the chord is chosen to be the maximum first-order derivative in each PWNL region. Note that PWNL regions for a nonlinear function are equivalent to piecewise constant (PWC) regions for first-order derivatives of the same nonlinear function.

Now that the chord is chosen to be the maximum first-order derivative in each PWNL region, PWNL regions for MOSFETs can be generated automatically with the following rules:

1. The maximum voltages of $V_{ds}$ and $V_{gs}$ are predefined. In our experiments, we use $V_{dd}$ as the maximum voltage for both of them. Given model parameters, the maximum $g_{ds}$ and $g_m$ for all operating regions can then be calculated.
2. With a predefined $0<\delta<1$, PWC region values for $g_{ds}$ and $g_m$ can be calculated as follows,
   $$g_n = g_{\max}$$
   $$g_{i-1} = (1-\delta) g_i, \qquad i = n, n-1, ..., 2$$
3. A lower bound of $g_{ds}$ and $g_m$ is predefined, so that rule (2) will stop whenever $g_{ds}$ and $g_m$ are less than the predefined lower bound. This is necessary to avoid a PWC region for $g_{ds}$ and $g_m$ to be too narrow.
4. A voltage step-size is chosen so that at least one PWC region exists for each of the calculated PWC region values of $g_{ds}$ and $g_m$ in the $V_{ds}$–($V_{gs}$-$V_{th}$) plane. A uniform voltage step-size has been used in our implementation for simplicity. Once the voltage step-size is finalized, $g_{ds}$ and $g_m$ at each grid point of the $V_{ds}$–($V_{gs}$-$V_{th}$) plane can be evaluated, so that each patch of the $V_{ds}$–($V_{gs}$-$V_{th}$) plane will be allocated to a PWC region of $g_{ds}$ and $g_m$.

As an example, the PWC regions for $g_{ds}$ of the MOSFET level 1 model are shown in Fig. 3, where $\delta$ is set to 1/3. It can be seen that there are a total of six PWC region values for $g_{ds}$ (including the cutoff region #0).

The proposed method can be easily incorporated into model compilers [19] for automatic generation of piece-wise weakly nonlinear regions for any nonlinear device with any device model. It should be noted that rules (1)~(4) are applicable to multi-dimensional PWNL region generation. However, one thing to keep in mind is that the PWNL definition of nonlinear devices is only used for the preconditioner, rather than the circuit matrix equation ***Ax=b***. Therefore, reasonable approximation can be made to ease the PWNL region generation.

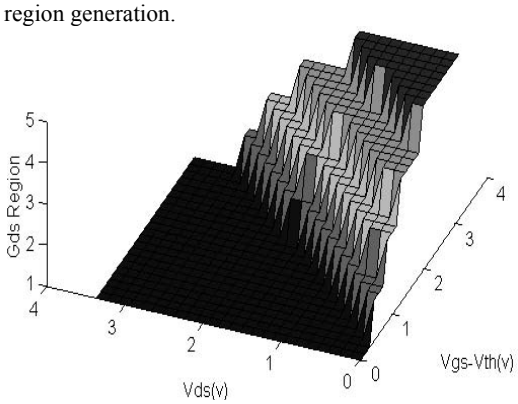


**Figure 3. PWC regions for $g_{ds}$ in the $V_{ds}$-($V_{gs}$-$V_{th}$) plane.**

The PWNL definition of nonlinear devices will introduce discontinuous first-order derivatives of a device model equation. It is well known that the continuity of a device model equation and its first-order derivatives is important for the successful convergence of the Newton-Raphson method. However, this requirement is not necessary for the Newton-Krylov method. The merit of the Newton-Krylov method in our framework is that the exact Newton-Raphson direction, which is determined by the exact first-order derivatives of a device model equation, can be well represented by the Krylov subspace constructed based on the approximate first-order derivatives (i.e., PWC first-order derivatives).

### 3.3 Low-Rank Updating

If only a few nonlinear devices change their operating PWNL regions during nonlinear iteration, the low-rank update technique [6] can be used to efficiently update the previously factorized L and U matrices rather than LU factorization. Therefore, in practice, the ratio of the number of nonlinear devices switching their operating PWNL regions vs. the total number of nonlinear and linear devices

could be used as a guide for choosing either the low-rank update technique or LU factorization. To utilize the low-rank update technique, it is required that the new circuit matrix $\boldsymbol{A}_{new}$ be derived from the old circuit matrix $\boldsymbol{A}_{old}$ as follows,

$$\boldsymbol{A}_{new} = \boldsymbol{A}_{old} + \boldsymbol{cr}^{T} \tag{13}$$

where $\boldsymbol{A}_{new}$ and $\boldsymbol{A}_{old}$ are both $n \times n$ matrices, $\boldsymbol{c}$ and $\boldsymbol{r}$ are both $n \times m$ matrices, and $m<<n$.

When a MOSFET switches its operating PWNL region within either the normal operating mode or the reverse operating mode (i.e., the drain and source terminals are flipped), its ($g_{ds}$, $g_m$, $g_{mbs}$) stamp on a circuit matrix will change. The stamp change of this MOSFET on the circuit matrix can be represented as follows,

$$\begin{array}{c} \\ \mathrm{D} \\ \mathrm{S} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{D} & \mathrm{G} & \mathrm{S} & \mathrm{B} \end{array} \\ \begin{bmatrix} \Delta g_{ds} & \Delta g_{m} & -\Delta g_{ds} - \Delta g_{m} - \Delta g_{mbs} & \Delta g_{mbs} \\ -\Delta g_{ds} & -\Delta g_{m} & \Delta g_{ds} + \Delta g_{m} + \Delta g_{mbs} & -\Delta g_{mbs} \end{bmatrix} \end{array}$$

where D, G, S, and B represent the drain, gate, source and bulk terminal indexes in the circuit matrix, respectively. It can be further represented in the following rank-one update ($m$=1) format,

$$\begin{bmatrix} \sqrt{a} \\ -\sqrt{a} \end{bmatrix} \begin{bmatrix} \dfrac{\Delta g_{ds}}{\sqrt{a}} & \dfrac{\Delta g_{m}}{\sqrt{a}} & -\dfrac{\Delta g_{ds} + \Delta g_{m} + \Delta g_{mbs}}{\sqrt{a}} & \dfrac{\Delta g_{mbs}}{\sqrt{a}} \end{bmatrix},$$

$$a = \max\left(\left|\Delta g_{ds} + \Delta g_{m} + \Delta g_{mbs}\right|, \left|\Delta g_{ds}\right|, \left|\Delta g_{m}\right|, \left|\Delta g_{mbs}\right|\right)$$

When a MOSFET switches its operating PWNL region from the normal mode to the reverse mode, the contribution of this MOSFET to the circuit matrix is changed similarly.

The low-rank update is efficient when the number of switching nonlinear devices is much less than the total number of nonlinear devices and linear elements. This is generally true in our framework, due to the two reasons. First, the PWNL definition of nonlinear devices are used for preconditioner construction, which requires a fairly coarse region partition than in the traditional piece-wise linear device modeling for device model evaluation. Second, our focused application is parasitic-coupled VLSI systems, where the number of linear parasitic elements is generally dominant. Therefore, we always use low-rank update during nonlinear iteration.

## 4. FGMRES-based Transient Simulation Flow

**Algorithm I. Transient simulation flow.**

```
DC operating point analysis
Choose an initial step size h_0, the basis step size h = h_0, t = 0
WHILE (t<T_final){
    OUTER LOOP: do{
        α= h_n/h, iter_no = 0
        INNER LOOP: do{
            IF(0.625<α<2.5){
                IF(PWNL region is changed)
                    Apply low-rank update on L/U matrices
            }ELSE{
                IF(iter_no==0)
                    Apply LU factorization
                ELSE{
                    IF(PWNL region is changed)
                        Apply low-rank update on L/U matrices
                }
            }
            Apply the preconditioned FGMRES method
            iter_no = iter_no + 1
        } while (not converged)
        Choose a new h_n based on LTE requirement
    } while (LTE greater than predefined error limit)
    t = t + h_n
}
```

The flow of the proposed method for time-domian nonlinear circuit simulation is shown in Algorithm I. It can be seen that LU factorization is only performed when time step-sizes vary out of the predefined $h_n/h$ range ($0.625 < h_n/h < 2.5$ has been set to make comparison with SILCA [11]). In other cases, L and U matrices are either kept unchanged or updated by the low-rank update technique when nonlinear devices change their operating PWNL regions. During the whole process, L and U matrices are used for preconditioning the FGMRES method.

Two types of preconditioners have been tested in our experiments:

1) A LU preconditioner composed of factorized full L and U matrices.
2) An incomplete LU preconditioner composed of matrices approximated from the factorized full L and U matrices – a matrix element $l(i,j)$ is removed if $|l(i,j)|<c\cdot\max(|l(*,j)|)$ in L or $|u(i,j)|< c\cdot\max(|u(i,*)|)$ in U. $c$ is a coefficient for the incomplete LU factorization, 0.001 is mainly used in our experiments. Since the incomplete LU preconditioner we use is derived from the already factorized L and U matrices, it is more robust and effective than general incomplete LU preconditioners [17] at the cost of more memory resources. The incomplete LU preconditioner is helpful to reduce the cost of forward/backward substitution during the preconditioning process.

## 5. Experimental Results

### 5.1 General nonlinear circuits

To verify the robustness of the proposed method for the simulation of general nonlinear circuits, several digital, analog and RF circuits have been tested. The simulation results are summarized in Table I. During the test, the full LU preconditioner has been used for the FGMRES method. Our implementation is based on SPICE3. For simplicity, the MOSFET level 1 model is used in our test. Parameter $\delta$ is set to 1/3 to automatically generate PWNL regions for MOSFETs.

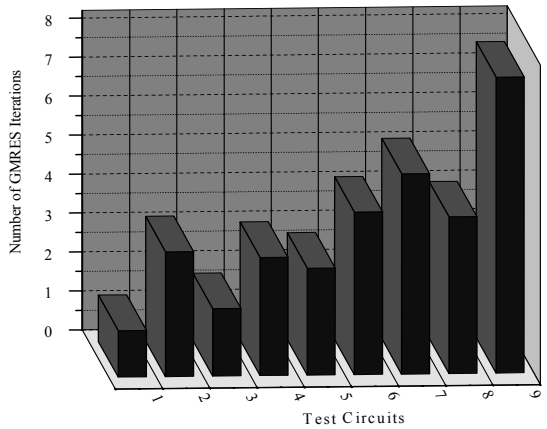


**Figure 4. The average number of FGMRES iterations for general nonlinear circuits.**

It can be seen from Table I that, with the preconditioned FGMRES method, the number of LU factorizations (#Tran LU) during transient simulation is reduced dramatically compared to that with SPICE3 (#Tran LU = #Tran iteration). Further, the number of total simulated time points (#Total points), the number of accepted time points (#Accepted points), and the number of transient iterations (#Tran iteration) with the preconditioned FGMRES method are kept almost the same as those with SPICE3, and are generally less than those with quasi-Newton base SILCA [11]. Figure 4 shows the average number of FGMRES iterations in each FGMRES solving process for test circuits (the dimension $m$ of the Krylov subspace $\mathbf{K}_m$). The average number of FGMRES iterations is below 5 for most of test circuits.

### 5.2 Power/ground network examples

To test the efficiency of the proposed method for the simulation of VLSI circuits with a large amount of parasitic elements, a power/ground network example as that used in [11] is simulated. The power and ground supply networks are modeled as two RCL mesh layers. In our example, between these two layers is a 20-stage inverter chain representing nonlinear circuits, different inverters of which are connected to different power/ground nodes. Furthermore, RCL loads are added for each inverter to model interconnect lines between adjacent stages. The size of two RCL meshes is changed to vary the number of linear parasitic elements (#Elemts in Table II and III). Parameter $\delta$ is set to 1/3 to generate PWNL regions for MOSFETs.

Table II summarizes the simulation results for the power/ground network example using SPICE3 and the FGMRES method with the full LU preconditioner. The error tolerance $\varepsilon$ is set to 1e-10 for the preconditioned FGMRES method. The speedup over SPICE3 is over 10$X$ for the largest example we test. It can be expected that more speedup could be achieved for larger power/ground networks. The average number of FGMRES iterations in each FGMRES solving process (#FGMRES Iter / #Tran Iter) is about 5. It is worthy noting that the number of LU factorization (#Tran LU) is reduced greatly with the preconditioned FGMRES method compared to SPICE3 (#Tran LU = #Tran Iter). As shown in Table II, the number of transient iterations with the preconditioned FGMRES method has been kept almost the same as that with SPICE3, which shows that the SPICE-like convergence property has been preserved.

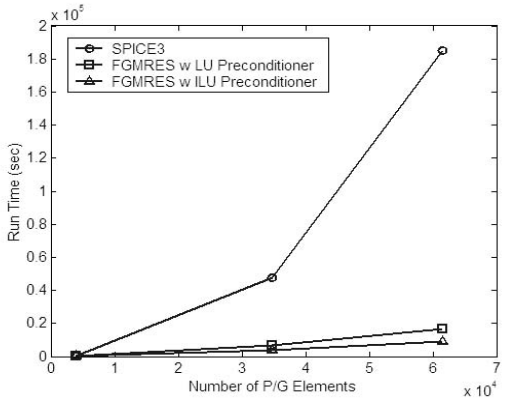


**Figure 5. The run time vs. the number of elements in the power/ground network.**

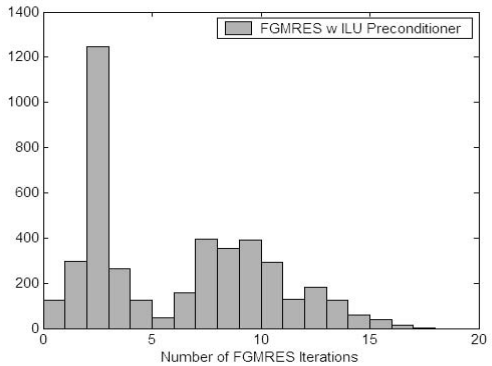


**Figure 6. The histogram of the number of FGMRES iterations with the ILU preconditioner.**

The simulation results of the FGMRES method with the incomplete LU (ILU) preconditioner are shown in Table III. It is seen that the FGMRES method with the ILU preconditioner achieves the best speedup over SPICE3 for the largest power/ground network – 20.68$X$, which is about 2$X$ speedup over the FGMRES method with the LU preconditioner (The run time comparison is shown in Fig. 5.). The reason is that the number of matrix elements in the ILU preconditioner is much less than those in the LU preconditioner. For the power/ground network example with 61602 elements, the histogram of the number of FGMRES iterations per FGMRES solving process is shown in Fig. 6. The average number of FGMRES iterations is about 6 to 7.

Finally, to test how the efficiency of the proposed method is affected by the percentages of linear parasitic elements in a circuit, we vary the amount of transistors in the logic circuit of the power-ground example. The results are shown in Table IV. Clearly the larger amount of linear parasitic elements, the more speed up.

From Table IV, we can see that for the P/G example (1600/4482), the cost of LU factorization has been reduced greatly, and the cost of FGMRES is comparable to that of LU factorization. However, the cost of low-rank update is becoming dominant. Shown in Fig. 7 is the histogram of the number of MOSFETs switching PWNL regions during one low-rank update. It can be seen that the number of MOSFETS switching their PWNL regions can be more than 200 during one low-rank update, and it is more expensive to compute LU factors by low-rank update than by full LU factorization. Therefore, in practice, whether to use low-rank update or full LU factorization should be determined based on the pre-set ratio of the number of switching nonlinear devices over that of total nonlinear devices. It is worthy noting that from Fig. 7, for most low-rank updates, only a few MOSFETS switch their PWNL regions.

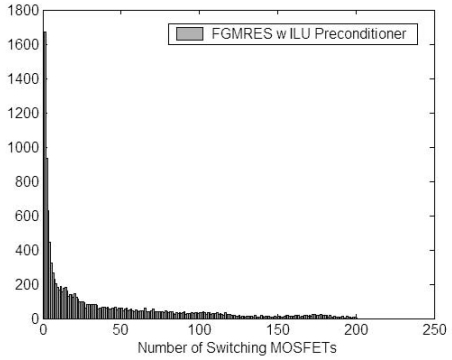


**Figure 7. The histogram of the number of FGMRES iterations with the ILU preconditioner.**

## 6. Conclusions and Future Research

In this paper, a quasi-Newton preconditioned Newton-Krylov method has been presented and implemented for efficient, accurate, and robust time-domain simulation of VLSI circuits with large amount of parasitic elements. Systematic methods of adaptive time-step size control and piece-wise weakly nonlinearity (PWNL) partitioning of any nonlinear device, combined with low-rank updating, have been proposed to minimize the number and cost of LU factorizations during the entire variable time step-size transient simulation. Orders of magnitude speedup has been achieved on power/ground network examples with the SPICE-like accuracy and robustness.

## References

[1] E. Acar, F. Dartu, and L. T. Pileggi, "TETA: Transistor-level waveform evaluation for timing analysis", IEEE Trans. on Computer-Aided Design, vol. 21, no. 5, pp. 605-616, May 2002.

[2] R. Burch, P. Yang, P. Cox, and K. Mayaram, "A new matrix solution technique for general circuit simulation", IEEE Trans. on Computer-Aided Design, vol. 12, no. 2, pp. 225-241, Feb. 1993.

[3] T.-H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods", Proc. IEEE/ACM Design Automation Conference, pp. 559-562, June 2001.

[4] P. F. Cox, R. G. Burch, P. Yang, and D. E. Hocevar, "New implicit integration method for efficient latency exploration in circuit

simulation", IEEE Trans. on Computer-Aided Design, vol. 8, no. 10, pp. 1051-1064, Oct. 1989.

[5] J. E. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory", SIAM Review, vol. 19, no. 1, pp. 46-89, Jan. 1977.

[6] T. Fujisawa, E. S. Kuh, and T. Ohtsuki, "A sparse matrix method for analysis of piecewise-linear resistive networks", IEEE Trans. on Circuit Theory, vol. CT-19, no. 6, pp. 571-584, Nov. 1972.

[7] K. R. Jackson and R. Sacks-Davis, "An alternative implementation of variable step-size multistep formulas for stiff ODEs", ACM Trans. on Math. Soft., vol. 6, no. 3, pp.295-318, Sept. 1980.

[8] K. J. Kerns, I. L. Wemple, and A. T. Yang, "Stable and efficient reduction of substrate model networks using congruence transforms", Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 207-214, Nov. 1995.

[9] D. A. Knoll and D. E. Keyes, "Jacobian-free Newton-Krylov methods: A survey of approaches and applications", Journal of Computational Physics, vol. 193, pp. 357-297, 2004.

[10] Y.-M. Lee and C. C.-P. Chen, "Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit method", Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 75-80, Nov. 2001.

[11] Z. Li and C.-J. R. Shi, "SILCA: Fast-yet-accurate time-domain simulation of VLSI circuits with strong parasitic coupling effects", Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 793-799, Nov. 2003.

[12] L. W. Nagel, SPICE: A Computer Program to Simulate Semiconductor Circuits, University of California, Berkeley, Tech. Rep., UCB/ERL M520, May 1975.

[13] A. Odabasioglu, M. Celik, and L. T. Pillegi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm", IEEE Trans. on Computer-Aided Design, vol. 17, no. 8, pp. 645-654, Aug. 1998.

[14] J. R. Phillips and L. M. Silveira, "Simulation approaches for strongly coupled interconnect systems", Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 430-437, Nov. 2001.

[15] A. R. Newton and A. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation", IEEE Trans. Computer-Aided Design, vol. 3, no. 4., pp. 308-331, 1984.

[16] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm", SIAM J. Scientific Computing, vol. 14, no. 2, pp. 461-469, Mar. 1993.

[17] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, SIAM, 2003.

[18] R. Telichevesky, K. Kundert, and J. White, "Fast simulation algorithms for RF circuits", Proc. IEEE Custom Integrated Circuit Conf, pp. 437-444, May 1996.

[19] B. Wan, B. Hu, L. Zhou and C.-J. R. Shi, ``MCAST: An abstract-syntax-tree based model compiler for circuit simulation", Proc. IEEE Custom Integrated Circuits Conf., pp. 249-252, San Jose, CA, Sept. 2003.

[20] J. K. White and A. Sangiovanni-Vincentelli, Relaxation Techniques for the Simulation of VLSI Circuits, Kluwer Academic Publishers, 1987.

**Table I. Simulation results on test circuits.**

| Test Circuits | | SPICE3 | | | FGMRES w LU Preconditioner | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | #Total points | #Accepted points | #Tran iteration | #Total points | #Accepted points | #Tran iteration | #Tran LU | #Low-rank update |
| 1 | Inverter | 142 | 127 | 340 | 141 | 127 | 338 | 63 | 64 |
| 2 | 20-stage inverter chain | 356 | 260 | 1159 | 369 | 266 | 1185 | 69 | 643 |
| 3 | Nand2 | 132 | 123 | 305 | 132 | 123 | 305 | 64 | 51 |
| 4 | One-shot trigger | 431 | 371 | 1360 | 431 | 371 | 1348 | 169 | 639 |
| 5 | Comparator | 140 | 126 | 410 | 140 | 126 | 403 | 47 | 163 |
| 6 | Opamp follower | 220 | 148 | 814 | 221 | 149 | 819 | 18 | 44 |
| 7 | Ring oscillator | 243 | 173 | 1020 | 256 | 176 | 1060 | 21 | 746 |
| 8 | VCO | 1281 | 887 | 4529 | 1280 | 887 | 4524 | 30 | 2251 |
| 9 | Power amplifier | 873 | 587 | 3451 | 840 | 581 | 3331 | 43 | 1761 |

**Table II. Simulation results for the power/ground network example ($\varepsilon$=1e-10, LU preconditioner).**

| #Elems | SPICE3 | | | Preconditioned FGMRES | | | | | | Speedup |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Tran Iter | Tran LU (sec) | Tot Tran (sec) | #Tran Iter | #Tran LU | #FGMRES Iter | Tran LU (sec) | FGMRES (sec) | Tot Tran (sec) | |
| 4002 | 4023 | 371.20 | 403.99 | 4106 | 54 | 19982 | 4.89 | 92.70 | 110.00 | 3.67 |
| 34802 | 4006 | 4.549e4 | 4.760e4 | 4087 | 55 | 18879 | 730.97 | 5919.82 | 6944.40 | 6.85 |
| 61602 | 4377 | 1.797e5 | 1.848e5 | 4253 | 53 | 20341 | 2279.88 | 13647.74 | 16556.52 | 11.16 |

**Table III. Simulation results for the power/ground network example ($\varepsilon$=1e-10, ILU preconditioner).**

| #Elems | #Tran Iter | #Tran LU | #FGMRES Iter | Tran LU (sec) | FGMRES (sec) | Tot Tran (sec) | Speedup |
|---|---|---|---|---|---|---|---|
| 4002 | 4241 | 52 | 24208 | 4.83 | 67.20 | 84.01 | 4.81 |
| 34802 | 4199 | 53 | 28311 | 688.40 | 3081.03 | 4066.19 | 11.71 |
| 61602 | 4254 | 56 | 28901 | 2323.74 | 5995.01 | 8938.69 | 20.68 |

**Table IV. Simulation results for the power/ground network example ($\varepsilon$=1e-10, ILU preconditioner).**

| #MOSFETs vs. #RCL | SPICE3 (sec) | | | | FGMRES w ILU Preconditioner (sec) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | LU | FBS | Load | Total | LU | Low-rank | FGMRES | Load | Total | Speed-up |
| 400/4122 | 2742.41 | 90.24 | 158.62 | 2991.27 | 59.74 | 184.09 | 234.84 | 55.21 | 533.88 | 5.60 |
| 800/4242 | 5269.12 | 297.40 | 544.76 | 6111.28 | 158.69 | 679.79 | 803.22 | 224.80 | 1866.50 | 3.27 |
| 1600/4482 | 35998.68 | 750.74 | 1235.39 | 37984.81 | 1723.75 | 10254.65 | 1533.65 | 472.36 | 13984.41 | 2.72 |