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, *Fellow, IEEE*

***Abstract*—In this paper, the Newton–Krylov method is explored for robust and efficient time-domain simulation of integrated circuits with large amount of parasitic elements. Different from LU-factorization-based direct methods used in SPICE-like circuit simulators, the Newton–Krylov method uses a preconditioned Krylov-subspace iterative method for solving linearized-circuit equations. A key contribution of this paper is to introduce an effective quasi-Newton preconditioning scheme for Krylov-subspace methods to reduce the number and cost of LU factorization during an entire time-domain circuit simulation. The proposed quasi-Newton preconditioning scheme consists of four key techniques: 1) a systematic method for adaptively controlling time step sizes; 2) automatically generated piecewise weakly nonlinear (PWNL) definition of nonlinear devices to construct quasi-Newton preconditioners; 3) low-rank update techniques for incrementally updating preconditioners; and 4) incomplete-LU preconditioning for efficiency. 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 the direct method used in SPICE. The proposed Newton–Krylov method is attractive for simulating circuits with massive parasitic *RLC* elements for postlayout verification. For a nonlinear circuit with power/ground networks with tens-of-thousand elements, the CPU time speedup over SPICE3 is over 20X, and it is expected to increase further with the circuit size.**

***Index Terms*—Circuit simulation, time-domain analysis.**

I. Introduction

IN MODERN deep-submicrometer 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. As a result, a circuit netlist extracted from a layout contains not only the originally designed circuit composed of transistors and passive devices but also a large amount of linear parasitic elements arising from modeling of power/ground planes, substrates, interconnects, vias, etc. It is desirable, especially for sensitive analog and mixed-signal circuit design, to perform full-chip time-domain simulation of such massively parasitic-coupled systems [26] before tape-out using SPICE-like circuit simulators [24]. 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 coupled systems, the complexity of LU factorization is approaching its worst case $\mathrm{O}(n^3)$, where n is the size of a circuit, instead of its average case $\mathrm{O}(n^{1.1-1.5})$ [24]. It is well known that for circuits of medium to large size, LU-factorization cost dominates the overall circuit-simulation time [23], [24].

Existing work to reduce the cost of LU factorization for circuit simulation can be classified into three categories. The first category is the so-called relaxation-based methods [27]. The basic idea is to fix part of circuit unknowns with the solutions obtained from the previous iterations or time points, so that the proceeding process of LU factorization can be simplified. Relaxation can be applied at the stages of linear and nonlinear system solving such as Gauss–Jacobi, Gauss–Seidel, successive-over-relaxation [29], and tree relaxation [31], or at the stage of time marching such as semi-implicit integration methods [37] and alternating-direction-implicit methods [29]. Two well-studied methods—iterated timing analysis [30] and waveform relaxation [17], [37]—fall into this category. For a class of MOS circuits, relaxation methods have been demonstrated to be orders of magnitude faster than the direct method. However, their stability and convergence properties depend strongly on circuit structures. Research effort has been directed to address this problem. For example, Texas Instrument's circuit simulator developed a preconditioning scheme to improve the robustness of the Gauss–Seidel method, called partial Gauss–Seidel (PGS) [3]. Nevertheless, despite their early success on timing analysis of MOS integrated circuits and recent success on substrate analysis [26] and power/ground network analysis [16], relaxation methods only have limited applications in industry circuit simulators.

The second category, which has been explored extensively since the inception of SPICE, belongs to a class of methods known as quasi-Newton methods [8]. The aim of quasi-Newton

Manuscript received February 16, 2005; revised November 18, 2005. This work was supported in part by the U.S. Defense Advanced Research Projects Agency NeoCAD Program under Grant 66001-01-1-8920, in part by the National Science Foundation (NSF) CAREER Award under Grant 9985507, and in part by the NSF/Semiconductor Research Corporation Joint Mixed-Signal Initiative under Grant CCR0120371. This paper was recommended by Associate Editor J. R. Phillips.

Z. Li was with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA. He is now with Cadence Design Systems, Inc., San Jose, CA 95134 USA (e-mail: zhaoli@cadence.com).

C.-J. R. Shi is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: cjshi@ee.washington.edu).

Digital Object Identifier 10.1109/TCAD.2006.882483

methods is to reduce the number of LU factorizations for circuit simulation. For numerical integration, several fixed leading coefficient integration methods [7], [11], [19] 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 Jacobian matrix bypassing to reduce the number of LU factorizations. As a special application, the successive-chord method [23] was implemented in a fast transistor-level gate-delay calculator TETA [1], where each transistor is modeled as a fixed linear resistor (called chord) combined with a variable nonlinear current source. To improve convergence, a variant of the successive-chord method—the successive-variable-chord method—has been proposed in [19] as an alternative to the Newton–Raphson method [23]. However, it is well known that the convergence rate of quasi-Newton methods for nonlinear iteration can degrade from quadratic, as in the case of the Newton–Raphson method, to linear.

The third category is to apply Krylov-subspace-based iterative methods [29] to solve the system of linearized equations in the inner Newton–Raphson iteration; this has been named the Newton–Krylov method [14]. 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 Krylov-subspace methods. For special applications such as in the shooting-Newton method and harmonic-balance analysis for RF circuit simulation [9], [18], [34]–[35], where a circuit matrix is generally block-diagonally dominant, a preconditioner can be well constructed. We note that Krylov-subspace methods have been successfully applied to the analysis of well-structured large-scale linear networks, such as substrate [26] and power/ground networks [5], as well as model order reduction of interconnect lines [25], substrates [13], and power/ground networks.

In this paper, we explore Krylov-subspace methods for time-domain simulation of nonlinear circuits with large amount of linear parasitic *RLC* elements. One natural idea is to use the LU matrices from the previous time points or nonlinear iterations as the preconditioner for Krylov-subspace methods. This was first explored by Yajima *et al.*, for circuit simulation in 1988 [38], and further enhanced by the Texas Instrument circuit-simulator team [2]. The Texas Instrument researchers used a conjugate-gradient squared method with a partial LU preconditioner, called PLUCGS. However, it was concluded in [2] and [3] that the PLUCGS method was even less efficient than the PGS method, a variant of the Gauss–Seidel method. Despite these progresses, it is generally known that the overhead associated with robust preconditioning is so high that Krylov-subspace iterative methods are not as efficient as the LU-based direct method as used in SPICE.

The main contribution of this paper is a systematic method to construct effective preconditioners based on quasi-Newton principles to use as few LU factorizations as possible during the whole time-domain nonlinear circuit simulation process. To this end, we utilize and generalize ideas developed in a quasi-Newton-based circuit simulator called SILCA [19], [22]. The efficiency of the proposed method is particularly due to the following four considerations on preconditioner construction.

1) The preconditioner is kept constant when time step sizes vary within a predefined range. A theory is developed as a guideline for recomputing LU factorization to update the preconditioner only if time step sizes vary violently.
2) A systematic method is proposed to partition the entire operating region of nonlinear devices into PWNL regions, so that 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 [4], [10] is applied to update the preconditioner efficiently rather than perform a complete new LU factorization.
4) Incomplete-LU (ILU) preconditioners derived from factorized full $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices are used for the best efficiency so as to reduce the cost of forward/backward substitution during the preconditioning process.

Previously, we have demonstrated the effectiveness of the preconditioned flexible generalized minimal residual (FGMRES) method in the quasi-Newton-based circuit-simulator SILCA [20].

Some preliminary results of this paper were presented in [21]. The remainder of this paper is organized as follows. In Section II, we provide an overview of quasi-Newton iterative methods and Krylov-subspace iterative methods. Section III introduces Newton–Krylov-based time-domain nonlinear-circuit simulation with the quasi-Newton preconditioning scheme. The systematic methods of adaptively controlling time step sizes and efficiently generating PWNL definition of nonlinear devices are further proposed. Section IV summarizes the proposed time-domain nonlinear circuit simulation flow. Experimental results on general nonlinear circuits and power/ground network examples are reported in Section V. Section VI concludes this paper.

II. Iterative Methods for Time-Domain Circuit Simulation

Despite the rapid progress in semiconductor design methodologies in the past 30 years, industry-wide circuit simulators are virtually all based on a public-domain software package called SPICE, developed in the early 1970s [24]. SPICE is based on the direct method. As shown in Fig. 1, SPICE time-domain circuit simulation consists of three major steps [23].

1) Numerical integration methods are used to discretize time and convert nonlinear differential equations to nonlinear algebraic equations.
2) The Newton–Raphson method is used to linearize nonlinear devices and solve nonlinear differential equations by iteratively solving a system of linearized equations.
3) The system of linearized equations $\boldsymbol{Ax} = \boldsymbol{b}$ is solved by first applying LU factorization to the circuit matrix $\boldsymbol{A}$ and then performing forward/backward substitution.

The direct method is 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

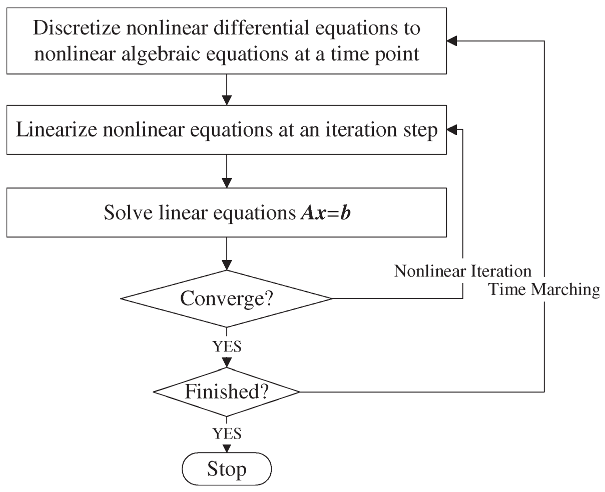

Discretize nonlinear differential equations to nonlinear algebraic equations at a time point
Linearize nonlinear equations at an iteration step
Solve linear equations Ax=b
Nonlinear Iteration
Time Marching
Converge?
YES
Finished?
YES
Stop

Fig. 1. Illustration of time-domain nonlinear circuit simulation.

circuit simulation incorporating parasitic effects [19], [26]. 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 $\boldsymbol{Ax} = \boldsymbol{b}$ 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.

A. Quasi-Newton Iterative 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 $\boldsymbol{Ax} = \boldsymbol{b}$ can be solved by (1) derived from the first-order Taylor expansion with the matrix $\boldsymbol{M}$ as the approximate Jacobian matrix (representing the first-order derivatives)

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}\left(\boldsymbol{b} - \boldsymbol{Ax}^{(k-1)}\right). \quad (1)$$

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

$$\begin{aligned}\boldsymbol{x}^{(k)} &= \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}\left(\boldsymbol{b}^{(k-1)} - \boldsymbol{A}^{(k-1)}\boldsymbol{x}^{(k-1)}\right)\\ &= \boldsymbol{x}^{(k-1)} + \boldsymbol{M}^{-1}\left(-\boldsymbol{f}^{(k-1)}\right)\end{aligned} \quad (2)$$

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 (2) will reduce to the Newton–Raphson method if $\boldsymbol{M} = \boldsymbol{A}$.

If $\boldsymbol{M}$ is chosen as a constant matrix during nonlinear iteration, (2) reduces to the successive-chord method [23]. To achieve a similar convergence rate to the Newton–Raphson method, various quasi-Newton methods [8], such as Broyden–Fletcher–Glodfarb–Shanno and Davidson–Fletcher–Powell, have been proposed to update the matrix $\boldsymbol{M}$ (or its inverse $\boldsymbol{M}^{-1}$) during nonlinear iteration. Recently, the successive-variable-chord method [19] has been proposed to use a constant $\boldsymbol{M}$ matrix when nonlinear devices reside in their present operating PWNL regions and update the $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices of $\boldsymbol{M}$ only when nonlinear devices change their operating PWNL regions. However, the convergence rate of quasi-Newton methods can degrade from quadratic, as that of the Newton–Raphson method, to linear. It has been reported in [19] that the number of nonlinear iterations with the successive-variable-chord method generally increases up to two to four times of that of the Newton–Raphson method. We note that the search direction with quasi-Newton methods is $\boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{Ax}^{(k-1)})$ for each nonlinear iteration step.

B. Krylov-Subspace Iterative Methods

Given an initial guess $\boldsymbol{x}^{(0)}$ to the circuit-matrix equation $\boldsymbol{Ax} = \boldsymbol{b}$, Krylov-subspace methods seek an approximate solution $\boldsymbol{x}^{(m)}$ from the subspace of $\boldsymbol{x}^{(0)} + \mathtt{K}_{\boldsymbol{m}}(\boldsymbol{A}, \boldsymbol{x}^{(0)})$ by imposing the Petrov–Galerkin condition [29]

$$\boldsymbol{b} - \boldsymbol{Ax}^{(\boldsymbol{m})} \perp \mathtt{L}_{\boldsymbol{m}}\left(\boldsymbol{A}, \boldsymbol{x}^{(0)}\right) \quad (3)$$

where $\mathtt{K}_{\boldsymbol{m}}(\boldsymbol{A}, \boldsymbol{x}^{(0)}) = \mathrm{span}\{\boldsymbol{r}^{(0)}, \boldsymbol{Ar}^{(0)}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}^{(0)}\}$, $\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{Ax}^{(0)}$, and $\mathtt{L}_{\boldsymbol{m}}(\boldsymbol{A}, \boldsymbol{x}^{(0)})$ is a subspace of dimension m. The Arnoldi's procedure [29] is generally applied to build an orthogonal basis of the Krylov-subspace $\mathtt{K}_{\boldsymbol{m}}$. Different choices of the subspace $\mathtt{L}_{\boldsymbol{m}}$ will lead to different types of Krylov-subspace methods. For example, $\mathtt{L}_{\boldsymbol{m}} = \mathtt{K}_{\boldsymbol{m}}$ gives rise to the Arnoldi method, and $\mathtt{L}_{\boldsymbol{m}} = \boldsymbol{A}\mathtt{K}_{\boldsymbol{m}} = \mathrm{span}\{\boldsymbol{Ar}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \ldots, \boldsymbol{A}^m\boldsymbol{r}^{(0)}\}$ leads to the GMRES method [29].

It is well known that a preconditioner [29] (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 $\mathtt{K}_{\boldsymbol{m}}$ is defined as follows:

$$\begin{aligned}&\mathtt{K}_{\boldsymbol{m}}\left(\boldsymbol{M}^{-1}\boldsymbol{A}, \boldsymbol{x}^{(0)}\right)\\ &\qquad = \mathrm{span}\left\{\boldsymbol{r}^{(0)}, \boldsymbol{M}^{-1}\boldsymbol{Ar}^{(0)}, \ldots, (\boldsymbol{M}^{-1}\boldsymbol{A})^{m-1}\boldsymbol{r}^{(0)}\right\}\end{aligned} \quad (4)$$

where $\boldsymbol{r}^{(0)} = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{Ax}^{(k-1)})$. It is not surprising that the effect of the preconditioner $\boldsymbol{M}$ on preconditioned Krylov-subspace methods is similar to that of the approximate Jacobian matrix $\boldsymbol{M}$ 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 $\mathtt{K}_{\boldsymbol{m}}$ is used for constructing the search direction, rather than only one single search direction $\boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{Ax}^{(k-1)})$ as in quasi-Newton methods. As a result, the search direction of the Newton–Raphson method could be well approximated with the Newton–Krylov method.

We use the FGMRES method [28], an extension of the original right-preconditioned GMRES method [29], to solve the system of linearized-circuit equations $(\boldsymbol{AM}^{-1})(\boldsymbol{Mx}) = \boldsymbol{b}$. The general flow FGMRES is illustrated in Table I. It consists of: 1) the initialization; 2) the Arnoldi procedure; 3) the QR factorization; and 4) the convergence testing. Clearly, one FGMRES iteration involves several matrix–vector multiplications and a QR factorization. The computational cost can

TABLE I
BASIC FLOW OF THE FGMRES METHOD

```
(1) Initialization:
    Choose an initial guess $x_0$ and a maximum iteration number $m$ for
      restarting the Arnoldi process.
    Initialize $(m+1) \times m$ matrix $H_m$ to zero.
(2) Arnoldi procedure:
    (a) Compute $r_0 = b - Ax_0$ and $v_1 = r_0 / \|r_0\|$;
    (b) For $j = 1, \ldots, m$
            Compute $z_j = M_j^{-1} v_j$,   $w = A z_j$;
            For $i = 1, \ldots, j$
                $h_{i,j} = \langle w, v_i \rangle$,   $w = w - h_{i,j} v_i$;
            Compute $h_{j+1,j} = \|w\|$ and $v_{j+1} = w / \|w\|$;
    (c) Define $Z_m = [z_1, \ldots, z_m]$.
(3) QR Factorization: Compute $x_m = x_0 + Z_m y_m$, $y_m$ minimizes
      $\|\beta e_1 - H_m y\|$.
(4) Convergence Testing. If converge, stop; else, $x_0 = x_m$, go to (2).
```

be significant if m is large. A good preconditioner is the key to reduce the dimension of the Krylov subspace (m) so that Krylov-subspace methods can be advantageous.

The FGMRES method is chosen since flexible preconditioners can be used during the GMRES solving process. It is known that the use of flexible preconditioners can lead to better efficiency and convergence properties [28], [32]. For example, Krylov-subspace methods have been used as preconditioners and shown to be effective for harmonic-balance analysis [18], [35]. Furthermore, preconditioners can be adjusted adaptively during the FGMRES solving process to explore the advantages of varying preconditioners for the best performance.

III. QUASI-NEWTON PRECONDITIONER CONSTRUCTION

In this section, we present a systematic method to construct effective preconditioners based on quasi-Newton methods. Section III-A presents adaptive time-step-size control for preconditioner computation. Section III-B describes the generation of generalized PWNL definition of nonlinear devices so that the same preconditioner can be used if nonlinear devices are operating in their current PWNL region. Section III-C describes how a low-rank update can be used to update, instead of to recompute, the preconditioners when a device changes its PWNL region.

First, we consider how to select effective preconditioners for FGMRES-based time-domain circuit simulation. In the context of general circuit simulation, $\boldsymbol{A}$ is a general invertible matrix and does not have the diagonal dominance property. Clearly, if we select $\boldsymbol{M}^{-1}$ to be $\boldsymbol{A}^{-1}$, $\boldsymbol{M}^{-1}\boldsymbol{A}$ will be the identity matrix. Therefore, we would like to select $\boldsymbol{M}^{-1}$ to be as close to $\boldsymbol{A}^{-1}$ as possible so that $\boldsymbol{M}^{-1}\boldsymbol{A}$ is as close to the identity matrix as possible. With circuit matrix $\boldsymbol{A}$ arising from the inner Newton–Raphson iteration loop, $\boldsymbol{M}^{-1}$ can be selected as $\boldsymbol{A}^{-1}$ is computed from a previous time point or a previous nonlinear-iteration step. Then, the problem of interest has the following objective—to compute as few $\boldsymbol{A}^{-1}$ as possible during the entire time-domain circuit simulation, which is exactly the objective of quasi-Newton methods for circuit simulation.

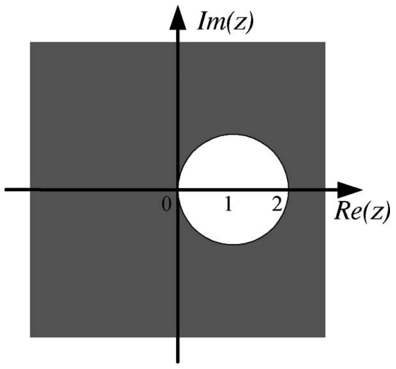


Fig. 2. Effective preconditioner region.

A. Adaptive Time-Step-Size Control for Preconditioners

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 follows:

$$\boldsymbol{Gx} + \boldsymbol{C\dot{x}} = \boldsymbol{b} \tag{5}$$

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

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

where $h_n = \alpha h$. To solve the above equation with preconditioned Krylov-subspace methods, the preconditioner we use is chosen to be $(\boldsymbol{G} + 2\boldsymbol{C}/h)$, which should be as close to $(\boldsymbol{G} + 2\boldsymbol{C}/(\alpha h))$ as possible.

Therefore, we introduce a parameter $0 < \eta < 1$ so that the preconditioner should satisfy the following inequality:

$$\begin{aligned}&\left\|\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{h}\right)^{-1}\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{\alpha h}\right) - \boldsymbol{I}\right\| \\ &\qquad = \left\|\left(\boldsymbol{G} + \frac{2\boldsymbol{C}}{h}\right)^{-1}\left(1 - \frac{1}{\alpha}\right)\frac{2\boldsymbol{C}}{h}\right\| < \eta < 1\end{aligned} \tag{7}$$

where $\| \bullet \|$ represents the spectral radius of the matrix. The above inequality can be rewritten as follows:

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

where $z = -h/(2\tau)$ and τ 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 $|z - 1| > |1 - 1/\alpha|/\eta = 1$. Then, we can draw the effective preconditioner region in the complex-z-plane, as in Fig. 2, in which the black region represents the effective preconditioner region.

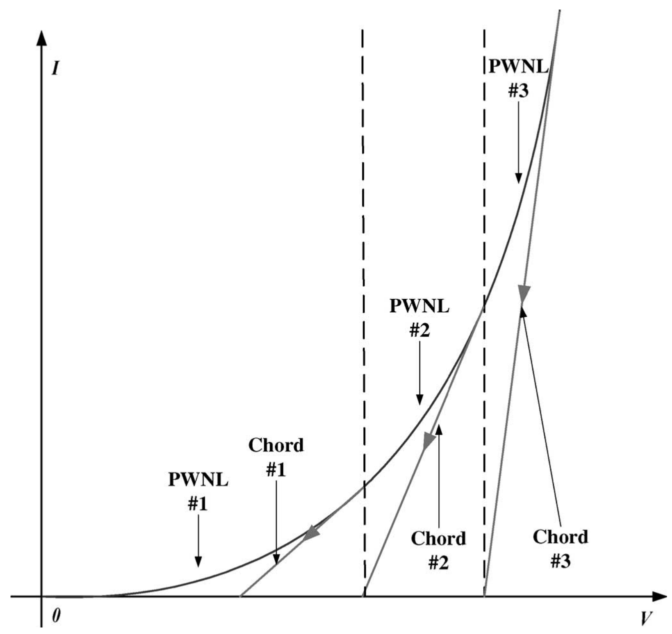


Fig. 3. PWNL definition of a nonlinear function.

A large range of α 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 α is far away from one. 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 formulas in SILCA [19].

B. PWNL Definition of Nonlinear Devices for Preconditioners

In this 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 well approximated by a constant (called a chord). Fig. 3 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 both sides of (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 (11), if g is always equal to $f'(x_i)$, as in the Newton–Raphson method, the convergence rate is quadratic. The smaller $|1 - f'(x_i)/g|$ is, the closer to the quadratic convergence rate (11) is. On the other hand, the larger $|1 - f'(x_i)/g|$ is, the larger the range of a PWNL region could be. Therefore, there exists a tradeoff between the convergence rate and the range of a PWNL region. In practice, we define the following condition with a parameter $0 < \delta < 1$:

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

If the maximum and minimum first-order derivatives for the studied PWNL region are $f'_{\max}$ and $f'_{\min}$, and both of them are positive (this assumption is generally true in practice except for specific nonlinear devices, such as tunnel diodes), it can be derived from (12) that the chord for this PWNL region should be chosen as follows:

$$\frac{f'_{\max}}{1+\delta} < g < \frac{f'_{\min}}{1-\delta}. \tag{13}$$

In our implementation, for simplicity, the chord is chosen to be the maximum first-order derivative in each PWNL region. The maximum first-order derivative could be computed with the knowledge of nonlinear device-model behaviors, such as monotonicity. 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 this paper, 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. It should be noted that the actual voltages of V_{ds} and V_{gs} could be larger than V_{dd}. In such cases, PWC-region values of g_{ds} and g_m are extended by the extrapolation based on rule 2).
2) With a predefined $0 < \delta < 1$, PWC-region values for g_{ds} and g_m can be calculated as follows:

$$g_n = g_{\max}$$
$$g_{i-1} = (1-\delta) g_i, \qquad i = n, n-1, \ldots, 2.$$

In case that extrapolation is required, extrapolated PWC-region values for g_{ds} and g_m can be calculated as follows:

$$g_n = g_{\max}$$
$$g_{i+1} = \frac{g_i}{(1-\delta)}, \qquad i = n, n+1, \ldots.$$

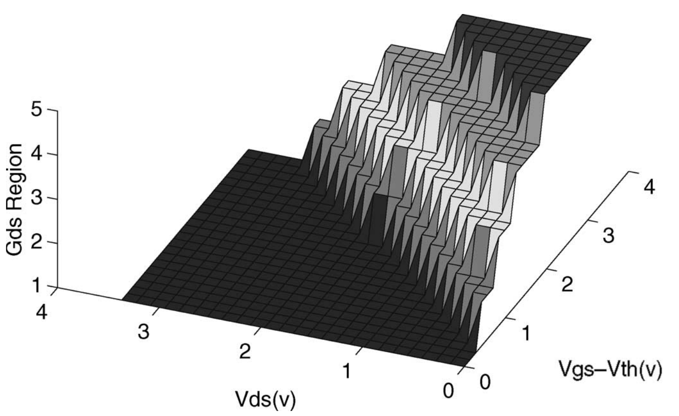


Fig. 4. PWC regions for g_{ds} in the $V_{ds} - (V_{gs}-V_{th})$ plane.

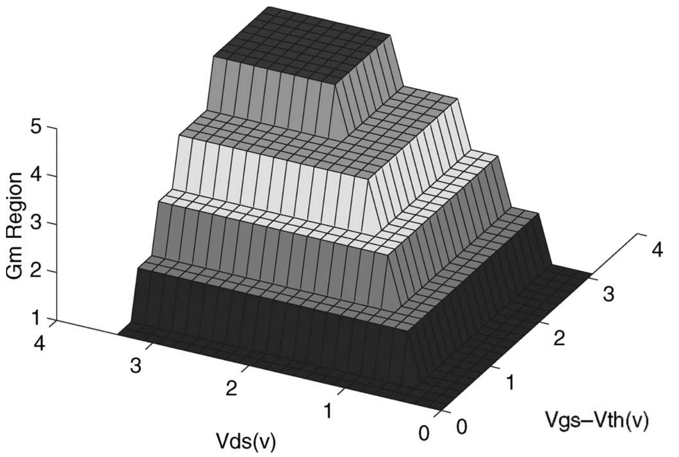


Fig. 5. PWC regions for g_m in the $V_{ds} - (V_{gs}-V_{th})$ plane.

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. Otherwise, either a smaller voltage step size should be chosen or the lower bound of g_{ds} and g_m should be adjusted in rule 3). 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, using the above rules, the PWC regions for g_{ds} and g_m of the MOSFET level 1 model are shown in Figs. 4 and 5, respectively, where δ is set to 1/3. It can be seen that there are a total of six PWC-region values for g_{ds} and g_m (including the cutoff region #0). It should be noted that effects due to V_{bs} have been incorporated into V_{th}. For the MOSFET level 1 model, g_{mbs} has a simple relationship with g_m [33]

$$g_{mbs} = g_m * \frac{dV_{th}}{dV_{sb}} = g_m * \frac{\gamma}{2\sqrt{\Phi + V_{sb}}}. \qquad (14)$$

For simplicity, we use the maximum $dV_{th}/dV_{sb} = \gamma/(2\sqrt{\Phi})$. Therefore, the PWC regions for g_{mbs} are the same as those for g_m. The proposed method can be incorporated into model compilers such as MCAST [36] for automatic generation of PWNL regions for nonlinear devices. It should be noted that rules 1)–4) are applicable to multidimensional PWNL-region generation.

We note that the PWNL definition of nonlinear devices is only used for constructing the preconditioners, rather than formulating the circuit-matrix equation $\boldsymbol{Ax} = \boldsymbol{b}$. This relaxes the accuracy requirement as in the traditional piecewise linear modeling or table lookup modeling of nonlinear devices, where many regions are needed to accomplish the required accuracy of model evaluation. For the preconditioner construction, much less numbers of regions are necessary.

Different δ values will lead to different numbers of PWNL regions generated for nonlinear devices. Although a smaller δ could reduce the number of FGMRES iterations during transient simulation, it would unfortunately increase the number of low-rank updates due to more PWNL regions generated for nonlinear devices. In this paper, to achieve a tradeoff between the number of low-rank updates and the number of FGMRES iterations, δ is chosen between 1/4 (8 PWNL regions) and 1/3 (6 PWNL regions) so that the number of PWNL regions for MOSFETs is below ten.

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 are important for the successful convergence of the Newton–Raphson method. However, this requirement is not necessary for the PWNL definition of nonlinear devices in our framework. The reason is that the PWNL definition of nonlinear devices is only used for constructing the preconditioner, while the circuit matrix equation $\boldsymbol{Ax} = \boldsymbol{b}$ is still formulated based on the original device model and its first-order derivatives. 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).

C. Low-Rank Update

If only a few nonlinear devices change their operating PWNL regions during nonlinear iteration, the low-rank-update technique [4], [10] can be used to efficiently update the previously factorized $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices rather than recompute LU factorization. Therefore, in practice, the ratio of the number of nonlinear devices switching their operating PWNL regions versus 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 $\boldsymbol{A}_{\mathbf{new}}$ be derived from the old circuit matrix $\boldsymbol{A}_{\mathbf{old}}$ as follows:

$$A_{\mathbf{new}} = A_{\mathbf{old}} + \boldsymbol{c}\boldsymbol{r}^{\boldsymbol{T}} \qquad (15)$$

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

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:

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

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:

$$\begin{bmatrix} \sqrt{a} \\ -\sqrt{a} \end{bmatrix} \begin{bmatrix} \frac{\Delta g_{ds}}{\sqrt{a}} & \frac{\Delta g_m}{\sqrt{a}} & -\frac{\Delta g_{ds}+\Delta g_m+\Delta g_{mbs}}{\sqrt{a}} & \frac{\Delta g_{mbs}}{\sqrt{a}} \end{bmatrix}$$

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

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, as shown in equation at the bottom of the page.

This can be represented by a similar rank-one update format. If multiple MOSFETs switch their operating PWNL regions during nonlinear iteration, a series of rank-one updates will be performed during one low-rank update.

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 two reasons. First, the PWNL definition of nonlinear devices is used for preconditioner construction, which requires a fairly coarse region partition than in the traditional piecewise linear-device modeling for device-model evaluation. This reduces the probability for nonlinear devices to switch PWNL regions. 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.

IV. FGMRES-Based Transient-Simulation Flow and Implementation

The flow of the proposed method for time-domain nonlinear-circuit simulation is shown in Table II. 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 [19]). In other cases, $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices are either kept unchanged or updated by the low-rank update technique when nonlinear devices change their operating PWNL regions. It should be noted that low-rank update is always used during nonlinear iteration in our implementation. During the whole process, $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices are used for preconditioning the FGMRES method.

TABLE II
Algorithm for FGMRES-Based Time-Domain Simulation

```
DC analysis
Choose an initial step-size $h_0$, the basis step-size $h = h_0$, $t = 0$
WHILE ($t<T_{\mathrm{final}}$){
  OUTER LOOP: do{
    $\alpha = h_n/h$, iter_no = 0
    INNER LOOP: do{
      IF(0.625<$\alpha$<2.5){
        IF(PWNL region is changed) {
          Apply low-rank update on L and U matrices
        }
      }ELSE{
        IF(iter_no==0) { Apply LU factorization }
        ELSE{
          IF(PWNL region is changed) {
              Apply low-rank update on L and 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)
  $t = t + h_n$
}
```

We emphasize that to preserve the SPICE-like convergence property for nonlinear circuits during transient simulation, the PWNL definitions of nonlinear devices are only used for computing the preconditioner for the FGMRES method. The original nonlinear device models are still used for constructing circuit-matrix equations $\boldsymbol{Ax} = \boldsymbol{b}$. This is advantageous when incorporating nonlinear capacitors into the proposed FGMRES method, since simplified linear capacitors could be used for the preconditioner while original complicated nonlinear capacitors are kept for building circuit-matrix equations.

The interface between a SPICE-like circuit simulator and the FGMRES package is described in Fig. 6. Note that a separated sparse-circuit matrix is stored in the FGMRES package. Therefore, more memory resources are required in our framework. However, since only the original sparse-circuit matrix before LU factorization is used in the FGMRES package, the extra memory resources are generally less than 10% of those required by the circuit matrix after LU factorization in a SPICE-like circuit simulator.

$$\begin{array}{c|cccc} & \mathrm{D} & \mathrm{G} & \mathrm{S} & \mathrm{B} \\ \hline \mathrm{D} & \Delta g_{ds} + g_m + g_{mbs} & \Delta g_m - 2g_m & -\Delta g_{ds} - \Delta g_m - \Delta g_{mbs} + g_m + g_{mbs} & \Delta g_{mbs} - 2g_{mbs} \\ \mathrm{S} & -\Delta g_{ds} - g_m - g_{mbs} & -\Delta g_m + 2g_m & \Delta g_{ds} + \Delta g_m + \Delta g_{mbs} - g_m - g_{mbs} & -\Delta g_{mbs} + 2g_{mbs} \end{array}$$

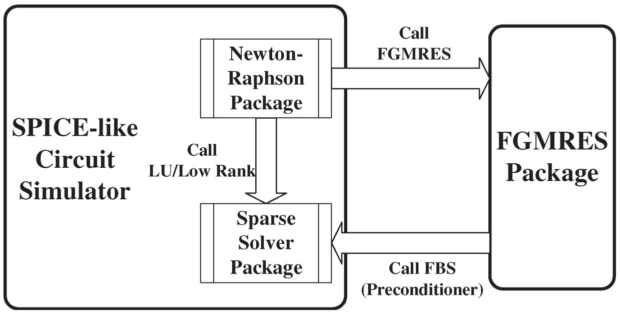


Fig. 6. Interface between a SPICE-like circuit simulator and FGMRES.

Three types of preconditioners have been tested in this paper.

1) An LU preconditioner composed of factorized full $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices.
2) An ILU preconditioner composed of matrices approximated from the factorized full $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices—a matrix element $l(i,j)$ is removed if $|l(i,j)| < c\max(|l(^*,j)|)$ in $\boldsymbol{L}$ or $|u(i,j)| < c\max(|u(i,^*)|)$ in $\boldsymbol{U}$. The coefficient for the ILU factorization is c; 0.001 is mainly used in this paper. Since the ILU preconditioner we use is derived from the already factorized $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices, it is more robust and effective than general ILU preconditioners [29] at the cost of more memory resources. The ILU preconditioner is helpful in reducing the cost of forward/backward substitution during the preconditioning process.
3) A hybrid preconditioner composed of the ILU preconditioner and a followed FGMRES preconditioner. In other words, for each preconditioning process, the ILU preconditioner is first applied to achieve an initial guess. After that, an FGMRES solver will be performed with the initial guess from the ILU preconditioner to reach a final solution of the hybrid preconditioner. The purpose is to reduce the dimensions of Krylov subspaces by exploiting more accurate preconditioners. Furthermore, it should be noted that the FGMRES preconditioner can be added to or dropped from the hybrid preconditioner for maximal efficiency. However, it should be noted that the preconditioning cost with the hybrid preconditioner is higher, which may offset the gain of reducing the dimensions of Krylov-subspaces.

V. Experimental Results

The proposed method has been implemented in SPICE3 and tested on a set of circuits ranging from general nonlinear analog, digital, and RF circuits to circuits with a large amount of linear parasitic elements. Section V-A describes the results of the robustness testing on general nonlinear circuits. Section V-B shows the efficiency of the proposed method on circuits with a large amount of parasitic elements.

A. Evaluation of 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 III. During the test, the full LU preconditioner has been used for the FGMRES method. Our implementation is based on SPICE3, which utilized the sparse matrix solver SPARSE1.3 [15]. For simplicity, the MOSFET level 1 model is used in our test. Parameter δ is set to 1/3 to automatically generate PWNL regions for MOSFETs.

For each test circuit, Table III shows the total number of time points simulated (#Total Points), the number of accepted time points (#Accepted points), and the total number of nonlinear-iteration numbers (#Transient Iterations, i.e., the number of solving linear equations) used by the original SPICE3 and the proposed method (FGMRES with LU preconditioner). For the proposed method, the total number of LU factorizations (#Tran LU) during transient simulation and the total number of low-rank updates (#Low-rank updates) are also shown. It is shown in Table III that with the preconditioned FGMRES method, the number of LU factorizations during transient simulation is reduced dramatically compared to that with SPICE3 $(\#\text{Tran LU} = \#\text{Tran iteration})$. Furthermore, the number of total simulated time points, the number of accepted time points, and the number of transient iterations with the preconditioned FGMRES method are almost the same as those using SPICE3 and are generally less than those using quasi-Newton-based SILCA [19].

Fig. 7 shows the histogram of the number of MOSFETs that switch their operating PWNL regions for each iteration in the 20-stage inverter chain. Clearly, the number of switching MOSFETs is much less than the total number of MOSFETs (40) in the inverter chain, owing to the PWNL definition of nonlinear devices introduced in Section III-B. It should be noted that one low-rank update can consist of one or more rank-one updates, determined by the number of switching MOSFETs during each iteration.

Fig. 8 shows the average number of FGMRES iterations for each FGMRES solving process for the test circuits (the dimension m of the Krylov-subspace $\mathtt{K}_{\boldsymbol{m}}$). The average number of FGMRES iterations is below five for most of the test circuits. For the power amplifier example, the average number of FGMRES iterations is about 7.6, which is higher than that for other test circuits. The reason is that most of the MOSFETs in the power amplifier operate in PWNL region #1 during transient simulation, which can be seen by comparing Figs. 4, 5, and 9. Since the PWNL region #1 is not modeled as accurately as other PWNL regions due to the PWNL definition rule 3), the efficiency of the proposed preconditioner becomes worse.

B. Evaluation of Power/Ground Networks

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 similar to that used in [6] and [19], shown in Fig. 10, is simulated. The power and ground supply networks are modeled as two *RCL* mesh layers (parasitic coupling capacitors are not shown in Fig. 10). 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

TABLE III
SIMULATION RESULTS OF GENERAL NONLINEAR 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
10	T481	231	211	691	231	211	690	99	322

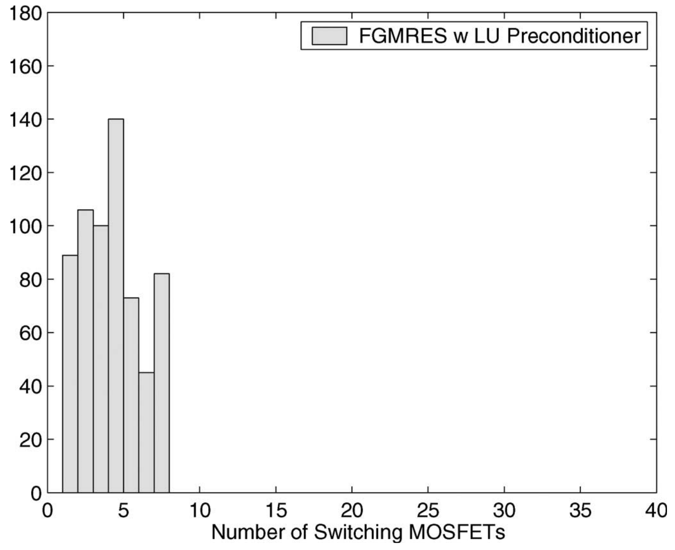


Fig. 7. Histogram of the number of switching MOSFETs in the inverter chain.

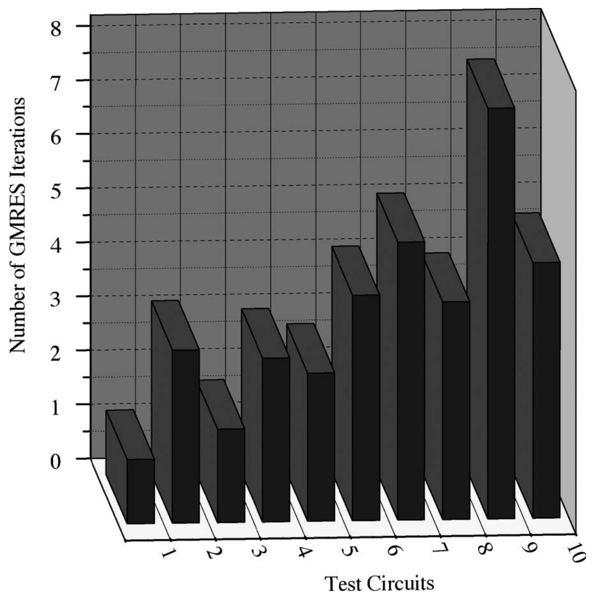


Fig. 8. Average number of FGMRES iterations for nonlinear test circuits.

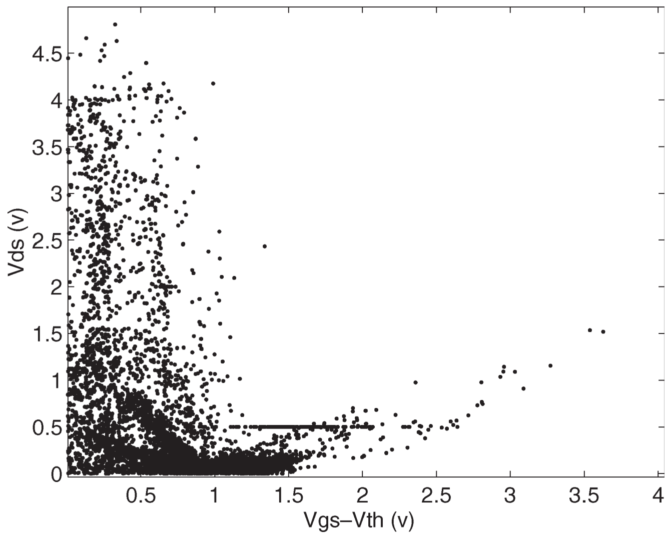


Fig. 9. Distribution of MOSFET operating points for the power amplifier.

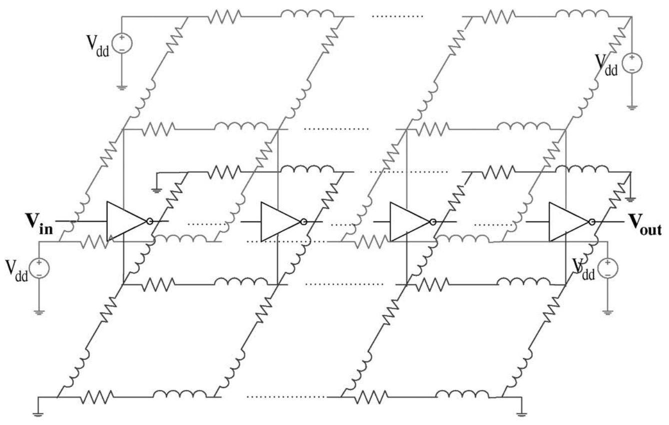


Fig. 10. Power/ground network example.

added for each inverter to model interconnect lines between adjacent stages. The sizes of two *RCL* meshes are changed to vary the number of linear parasitic elements (#Elems in Tables IV–VI). Parameter δ is set to 1/3 to generate PWNL regions for MOSFETs.

Table IV summarizes the simulation results for the power/ground network example using SPICE3 and the FGMRES method with the full LU preconditioner. The error tolerance ε is set to $1e$-10 for the preconditioned FGMRES method. The speedup over SPICE3 is over 10X for the largest example we tested. It can be expected that more speedup could be achieved for larger power/ground networks. The average number of

TABLE IV
SIMULATION RESULTS OF POWER/GROUND NETWORK EXAMPLES (LU PRECONDITIONER)

#Elems	SPICE3			Preconditioned FGMRES							Speedup
	#Tran Iter	Tran LU (sec)	Tot Tran (sec)	#Tran Iter	#Tran LU	#FGMRES Iter	Tran LU (sec)	Precond (sec)	FGMRES (sec)	Tot Tran (sec)	
4002	4023	371.20	403.99	4106	54	19982	4.89	71.51	92.70	110.00	3.67
34802	4006	4.549e4	4.760e4	4087	55	18879	730.97	5593.77	5919.82	6944.40	6.85
61602	4377	1.797e5	1.848e5	4253	53	20341	2279.88	13006.20	13647.74	16556.52	11.16

TABLE V
SIMULATION RESULTS OF POWER/GROUND NETWORK EXAMPLES (INCOMPLETE LU PRECONDITIONER)

#Elems	#Tran Iter	#Tran LU	#FGMRES Iter	Tran LU (sec)	Precond (sec)	FGMRES (sec)	Tot Tran (sec)	Speedup
4002	4241	52	24208	4.83	41.54	67.20	84.01	4.81
34802	4199	53	28311	688.40	2596.43	3081.03	4066.19	11.71
61602	4254	56	28901	2323.74	5097.20	5995.01	8938.69	20.68

TABLE VI
SIMULATION RESULTS OF POWER/GROUND NETWORK EXAMPLES (HYBRID PRECONDITIONER)

#Elems	#Tran Iter	#Tran LU	#Top-level FGMRES Iter	#Total FGMRES Iter	Tran LU (sec)	Precond (sec)	FGMRES (sec)	Tot Tran (sec)	Speedup
4002	4277	48	12661	27034	4.35	46.63	77.01	93.96	4.30
34802	4214	51	13164	33267	662.59	3030.31	3598.73	4560.36	10.44
61602	4116	57	12688	32252	2324.96	5086.80	6056.12	8936.80	20.68

FGMRES iterations in each FGMRES solving process (#FGMRES Iter / #Tran Iter) is about five. It is worth noting that the number of LU factorizations (#Tran LU) is reduced greatly by the preconditioned FGMRES method compared to SPICE3 (#Tran LU $=$ #Tran Iter). It is observed that the speedup over SPICE3 with the preconditioned FGMRES method is less than with SILCA [19]. This is due to the fact that the per-iteration cost using the preconditioned FGMRES method is higher than that in SILCA. However, as shown in Table IV, the number of transient iterations with the preconditioned FGMRES method has been kept close to that of SPICE3, which shows that the SPICE-like convergence property has been preserved. Furthermore, it will be shown next that greater speed can be achieved by applying efficient preconditioners. It should be noted that most of the FGMRES run time is consumed by preconditioning (Precond in Tables IV–VI), especially for large examples.

The simulation results of the FGMRES method with the ILU preconditioner are shown in Table V. The table shows that the FGMRES method with the ILU preconditioner achieves the best speedup over SPICE3 for the largest power/ground network $-$20.68X, which is about 2X speedup over the FGMRES method with the LU preconditioner. The run-time comparison is shown in Fig. 11.

The speedup reason is that the number of matrix elements in the ILU preconditioner is much less than the number in the LU preconditioner. For the power/ground network example with 61 602 elements, the histogram of the number of $\boldsymbol{L}$ and $\boldsymbol{U}$ matrix elements during transient simulation are shown in Fig. 12. The number of matrix elements in the full $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices is 3 116 915 for this example. The number of nonzero matrix elements in the ILU preconditioner is reduced to about 1/4 to 1/15 of that of the full LU preconditioner. Consequently, the cost of the preconditioning procedure with the ILU preconditioner has been reduced greatly. The histogram of the number of FGMRES iterations per FGMRES solving process is shown

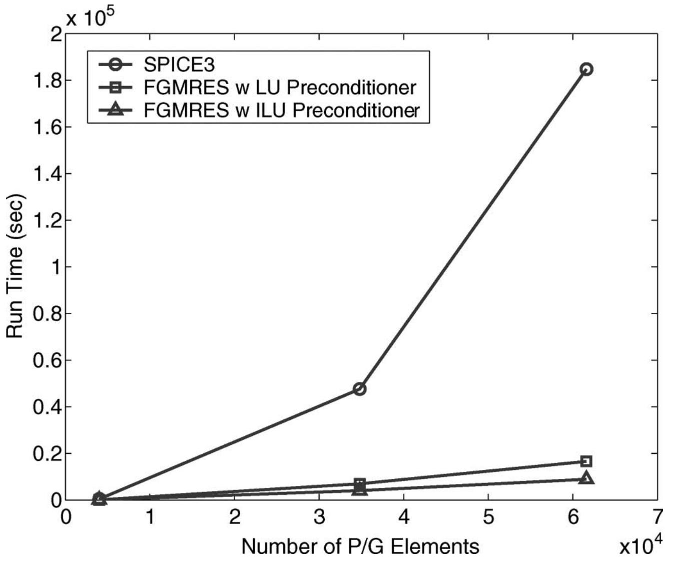


Fig. 11. Run time versus the number of elements in the power/ground network.

in Fig. 13. The average number of FGMRES iterations is about six to seven.

Fig. 14 shows the average number of FGMRES iterations in each FGMRES solving process compared to the coefficient c for the ILU preconditioner. It can be seen that the larger the coefficient c is, the more FGMRES iterations will be taken. The reason is that more matrix elements in the $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices will be removed with a larger coefficient c, which causes the ILU preconditioner less effective. However, the cost of evaluating the ILU preconditioner is reduced with an increased coefficient c. Therefore, it is expected that there will be an optimum coefficient c to minimize the FGMRES cost. This has been confirmed in Fig. 15, which shows the variation of the average FGMRES cost in each FGMRES solving process

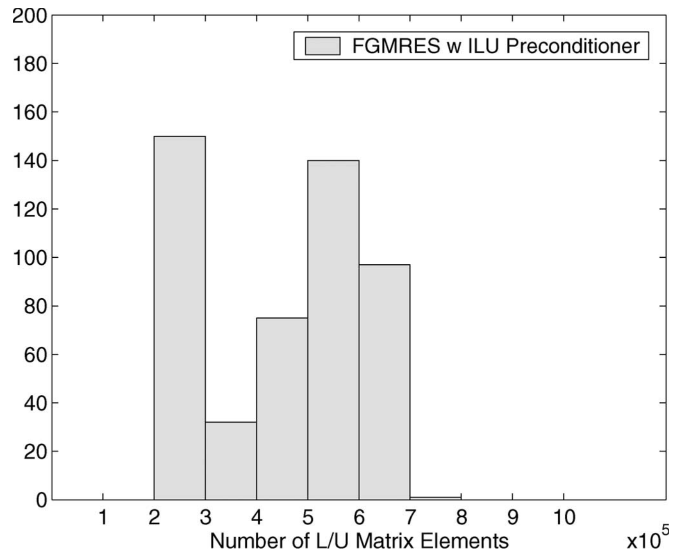


Fig. 12. Histogram of the number of $\boldsymbol{L}$ and $\boldsymbol{U}$ matrix elements.

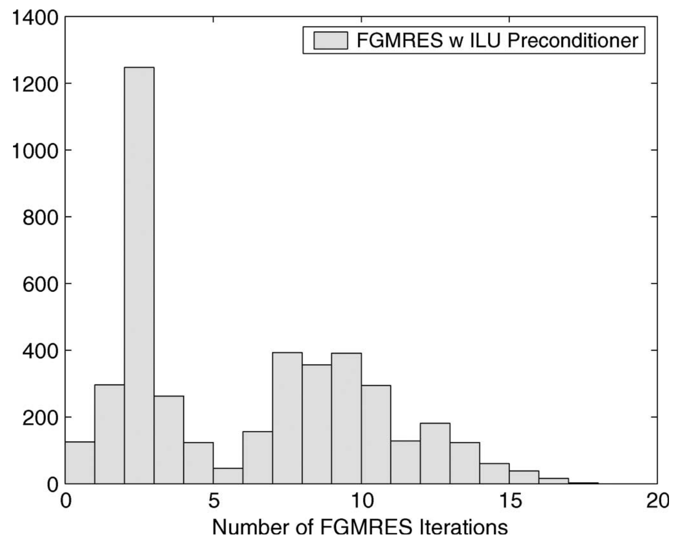


Fig. 13. Histogram of the number of FGMRES iterations with the ILU preconditioner.

with the coefficient c for the ILU preconditioner. The average FGMRES cost in each FGMRES solving process is minimized near $c = 0.001$, which has been used in this paper.

The simulation results of the FGMRES method with the hybrid preconditioner are shown in Table VI. A larger error tolerance $\varepsilon = 1e - 1$ has been set to the FGMRES preconditioner. Although the number of top-level FGMRES iterations has been reduced compared to that in Table V, the number of total FGMRES iterations has been increased, unfortunately. Therefore, the speedup over SPICE3 with the hybrid preconditioner is not as good as that with the ILU preconditioner alone. The main reason is that the ILU quasi-Newton preconditioner is already effective for transient simulation in terms of the number of FGMRES iterations per FGMRES solving process, as shown in Fig. 13. The overhead of further using the FGMRES preconditioner is more than the resulting savings. However, the hybrid preconditioner does reduce the number of FGMRES iterations per FGMRES solving process, as shown in Fig. 16.

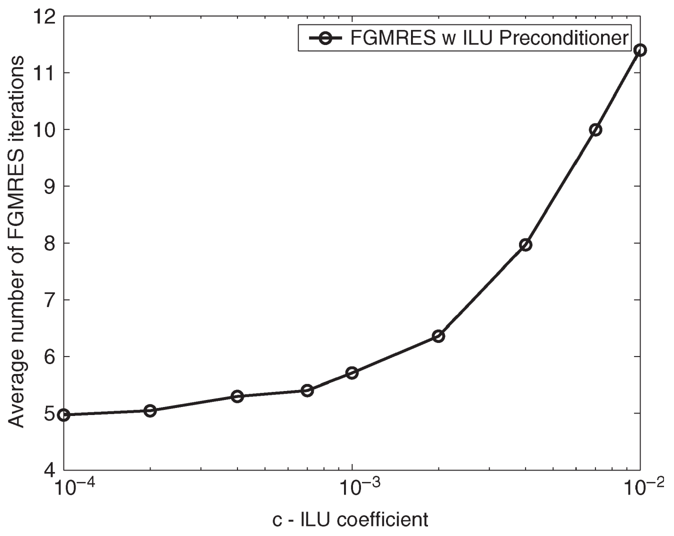


Fig. 14. Average number of FGMRES iterations versus the ILU coefficient.

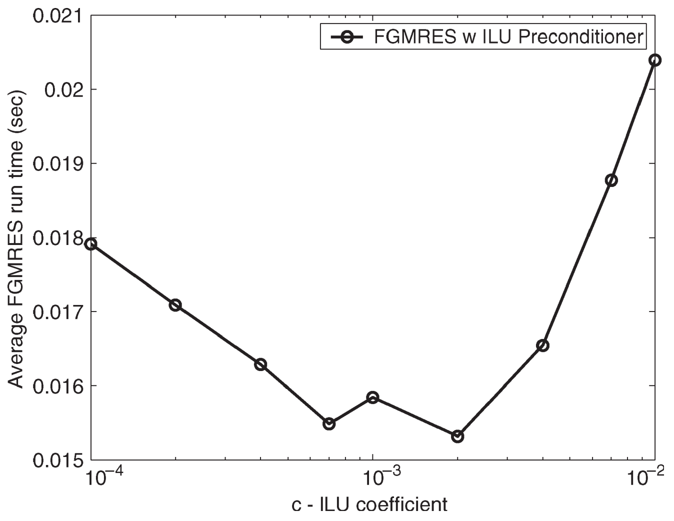


Fig. 15. Average FGMRES run time versus the ILU coefficient.

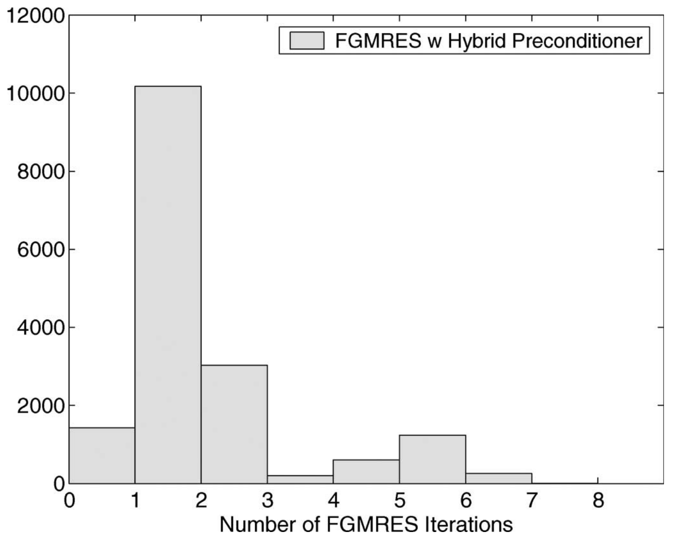


Fig. 16. Histogram of the number of FGMRES iterations with the hybrid preconditioner.

TABLE VII
SIMULATION RESULTS OF POWER/GROUND NETWORK EXAMPLES INCOMPLETE LU PRECONDITIONER)

#MOSFETs vs. #RCL	SPICE3 (sec)				FGMRES w ILU Preconditioner (sec)					
	LU	FBS	Load	Total	LU	Low-rank	FGMRES	Load	Total	Speedup
40/4062	1587.36	90.30	163.13	1840.79	22.31	9.87	306.49	46.93	385.60	4.77
200/4062	1852.90	98.13	170.83	2121.85	42.78	80.05	304.80	67.60	495.23	4.28
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

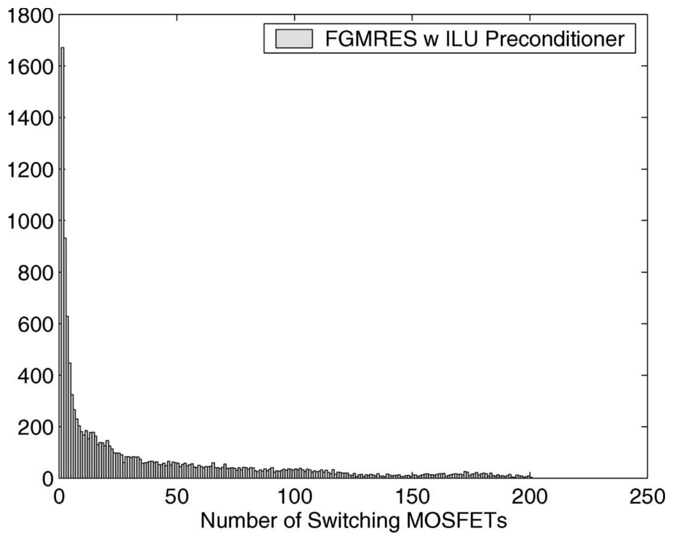


Fig. 17. Histogram of the number of switching MOSFETs in the power/ground network example (1600/4482).

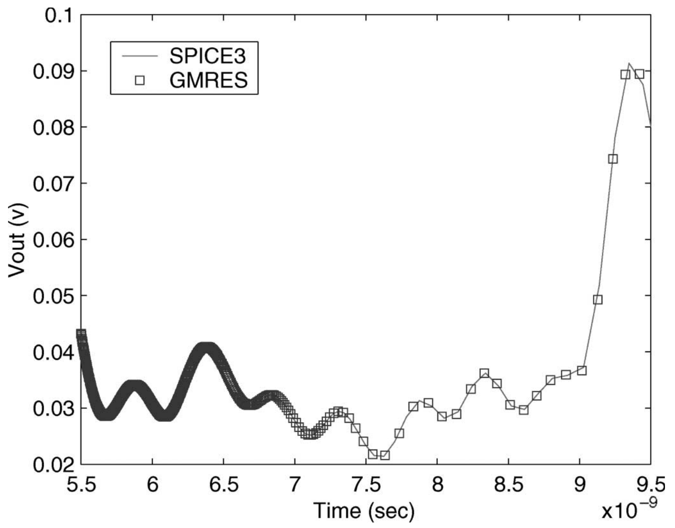


Fig. 18. Output waveform of the power/ground example.

Therefore, it can be expected that the hybrid preconditioner will gain when the convergence of the FGMRES method is slow, i.e., in case that the ILU preconditioner is not constructed based on full LU factorization for a memory saving tradeoff. In harmonic-balance analysis for RF circuits, it has been observed that Krylov-subspace methods with hybrid preconditioners [18], [35] converge better than those with block-diagonal preconditioners alone [9].

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 VII. Clearly, the larger the amount of linear parasitic elements, the greater the speedup. Even for the last example where the number of nonlinear elements (1600 transistors) is comparable to the number of parasitic elements (4482 RLC elements), the FGMRES method still gives a speedup close to three.

In Table VII, we can see that for the last power/ground network 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. 17 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. In this case, it is more expensive to update LU matrices by low-rank update than by full LU factorization.

Therefore, in practice, whether to use low-rank update or full LU factorization is determined based on the preset ratio of the number of switching nonlinear devices over that of the total nonlinear and linear devices. It is worth noting, from Fig. 17, that for most low-rank updates, only a few MOSFETs switch their PWNL regions.

The FGMRES method has been tested to produce the same accurate results as SPICE3. For example, Fig. 18 shows the output waveforms of the inverter chain between the power and ground networks simulated using SPICE3 and using the FGMRES. It can be seen that the low-level voltage of the output is larger than the ideal ground voltage due to the IR drop and $L^{*}dI/dt$ effects.

We see that the FMGRES method can be superior to the direct method when the size of a circuit is large and when the cost of LU factorization dominates the overall simulation time (i.e., the cost of LU factorization is much larger than the cost of forward and backward substitution and device evaluation). In this case, the cost of LU factorization can be reduced dramatically with the proposed quasi-Newton preconditioner, and the overhead associated with an FGMRES solving (Arnoldi procedure and QR factorization) can be offset by the gain in reducing the LU-factorization cost. We notice that the preconditioning cost can be further reduced by applying an ILU preconditioner and by using efficient low-rank update for updating preconditioners when the number of switching nonlinear devices is far less than the total number of nonlinear and linear devices in a circuit.

VI. Conclusion

In this paper, a quasi-Newton preconditioned Newton–Krylov method has been presented and implemented for efficient, SPICE-accurate, and robust time-domain simulation of integrated circuits with large amount of parasitic elements. Systematic methods of adaptive time-step size control and PWNL partitioning of nonlinear devices, combined with low-rank update, have been proposed to minimize the number and cost of LU factorizations used for preconditioner construction during the entire variable-time-step size transient simulation. Two types of preconditioners have been studied to further reduce the preconditioning cost: ILU preconditioners derived from factorized full $\boldsymbol{L}$ and $\boldsymbol{U}$ matrices and hybrid preconditioners composed of ILU preconditioners followed by an FGMRES preconditioner. Experimental results have shown that these techniques dramatically reduce the number and cost of required LU factorizations during transient simulation. Orders of magnitude of speedup has been achieved on power/ground network examples with the SPICE-like accuracy and robustness.

The proposed Newton–Krylov method with quasi-Newton preconditioners is as robust and accurate as the direct method used in SPICE. It has been shown to be very efficient for the simulation of large-size integrated circuits with a substantial amount of (dense) linear elements resulting from modeling of strong parasitic couplings. Very recently, we have successfully demonstrated that the proposed method can be effective in reducing the cost of process–voltage–temperature corner simulation, an increasingly important task in integrated-circuit design [12]. However, we note that for small-to-medium-size nonlinear circuits in the prelayout design, the overhead associated with preconditioning may dominate the gain in reducing the cost of LU factorization. Future work will be on techniques to reduce the preconditioning cost.

Acknowledgment

The authors would like to thank Prof. K. Mayaram of Oregon State University for many stimulating discussions and the three anonymous reviewers for their comments that greatly enhanced the presentation of this paper.

References

[1] E. Acar, F. Dartu, and L. T. Pileggi, “TETA: Transistor-level waveform evaluation for timing analysis,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 5, pp. 605–616, May 2002.

[2] R. Burch, K. Mayaram, J.-H. Chern, P. Yang, and P. Cox, “PGS and PLUCGS—Two new matrix solution techniques for general circuit simulation,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1989, pp. 408–411.

[3] R. Burch, P. Yang, P. Cox, and K. Mayaram, “A new matrix solution technique for general circuit simulation,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 2, pp. 225–241, Feb. 1993.

[4] H. W. Buurman, “From circuit to signal—Development of a piecewise linear simulator,” Ph.D. dissertation, Dept. Electr. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, Jan. 1993.

[5] T.-H. Chen and C. C.-P. Chen, “Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods,” in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 2001, pp. 559–562.

[6] T.-H. Chen, J.-L. Tsai, C. C.-P. Chen, and T. Karnik, “HiSIM: Hierarchical interconnect-centric circuit simulator,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 489–496.

[7] 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. Comput.-Aided Design Integr. Circuits Syst.*, vol. 8, no. 10, pp. 1051–1064, Oct. 1989.

[8] J. E. Dennis and J. J. Moré, “Quasi-Newton methods, motivation and theory,” *SIAM Rev.*, vol. 19, no. 1, pp. 46–89, Jan. 1977.

[9] P. Feldmann, B. Melville, and D. Long, “Efficient frequency domain analysis of large nonlinear analog circuits,” in *Proc. IEEE Custom Integr. Circuit Conf.*, May 1996, pp. 461–464.

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

[11] K. R. Jackson and R. Sacks-Davis, “An alternative implementation of variable step-size multistep formulas for stiff ODEs,” *ACM Trans. Math. Soft.*, vol. 6, no. 3, pp. 295–318, Sep. 1980.

[12] B. Hu and C.-J. R. Shi, “Fast-yet-accurate PVT simulation by combing direct and iterative methods,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 495–501.

[13] K. J. Kerns, I. L. Wemple, and A. T. Yang, “Stable and efficient reduction of substrate model networks using congruence transforms,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1995, pp. 207–214.

[14] D. A. Knoll and D. E. Keyes, “Jacobian-free Newton–Krylov methods: A survey of approaches and applications,” *J. Comput. Phys.*, vol. 193, no. 2, pp. 297–357, Jan. 2004.

[15] K. S. Kundert and A. Sangiovanni-Vincentelli, *Sparse User’s Guide—A Sparse Linear Equation Solver Version 1.3a*. Berkeley: Univ. California, Apr. 1988.

[16] Y.-M. Lee and C. C.-P. Chen, “Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit method,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 75–80.

[17] E. Lelarasmee and A. Sangiovanni-Vincentelli, “RELAX: A new circuit simulator for large scale MOS integrated circuits,” in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 1982, pp. 682–690.

[18] P. Li and L. T. Pileggi, “Efficient harmonic balance simulation using multi-level frequency decomposition,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 677–682.

[19] Z. Li and C.-J. R. Shi, “SILCA: Fast-yet-accurate time-domain simulation of VLSI circuits with strong parasitic coupling effects,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2003, pp. 793–799.

[20] ——, “An efficiently preconditioned GMRES method for fast parasitic-sensitive deep-submicron VLSI circuit simulation,” in *Proc. Des., Autom. and Test Eur. Conf.*, Mar. 2005, pp. 752–757.

[21] ——, “A quasi-Newton preconditioned Newton–Krylov method for robust and efficient time domain simulation of integrated circuits with strong parasitic couplings,” in *Proc. IEEE/ACM Asian and South Pacific Des. Autom. Conf.*, Jan. 2006, pp. 402–407.

[22] ——, “SILCA: SPICE-accurate iterative linear-centric analysis for efficient time-domain simulation of VLSI circuits with strong parasitic couplings,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1087–1103, Jun. 2006.

[23] W. J. McCalla, *Fundamentals of Computer-Aided Circuit Simulation*. Norwell, MA: Kluwer, 1988.

[24] L. W. Nagel, “SPICE: A Computer Program to Simulate Semiconductor Circuits,” Univ. California, Berkeley, Tech. Rep. UCB/ERL M520, May 1975.

[25] A. Odabasioglu, M. Celik, and L. T. Pillegi, “PRIMA: Passive reduced-order interconnect macromodeling algorithm,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 645–654, Aug. 1998.

[26] J. R. Phillips and L. M. Silveira, “Simulation approaches for strongly coupled interconnect systems,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 430–437.

[27] A. R. Newton and A. Sangiovanni-Vincentelli, “Relaxation-based electrical simulation,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. CAD-3, no. 4, pp. 308–331, Oct. 1984.

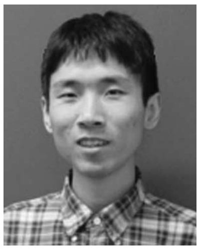
[28] Y. Saad, “A flexible inner-outer preconditioned GMRES algorithm,” *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 461–469, Mar. 1993.

[29] ——, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM, 2003.

[30] R. A. Saleh, J. E. Kleckner, and A. R. Newton, “Iterated timing analysis and SPLICE1,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Sep. 1983, pp. 139–140.

[31] C.-J. Shi and K. Zhang, “Tree relaxation: A new iterative solution method for linear equations,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 1988, pp. 2355–2359.

[32] V. Simoncini and D. B. Szyld, “Flexible inner-outer Krylov subspace methods,” *SIAM J. Numer. Anal.*, vol. 40, no. 6, pp. 2219–2239, Jan. 2003.
[33] *Star-Hspice Manual*, *Release 1998.2,* Avanti! Corp., Freemont, CA, 1998.
[34] R. Telichevesky, K. Kundert, and J. White, “Fast simulation algorithms for RF circuits,” in *Proc. IEEE Custom Integr. Circuit Conf.*, May 1996, pp. 437–444.
[35] F. Veersé, “Efficient iterative time preconditioners for harmonic balance RF circuit simulation,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2003, pp. 251–254.
[36] B. Wan, B. Hu, L. Zhou, and C.-J. R. Shi, “MCAST: An abstract-syntax-tree based model compiler for circuit simulation,” in *Proc. IEEE Custom Integr. Circuits Conf.*, San Jose, CA, Sep. 2003, pp. 249–252.
[37] J. K. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*. Norwell, MA: Kluwer, 1987.
[38] A. Yajima, F. Yamamoto, and T. Morioka, “Complete LU decomposition conjugate residual method and its performance for large-scale circuit simulation,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 1988, pp. 619–622.



Zhao Li received the B.S. degree in electronics and the M.S. degree in microelectronics and solid-state electronics from Tsinghua University, Beijing, China, in 1998 and 2000, respectively, and the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 2005.

He is currently with Cadence Design Systems, Inc., San Jose, CA. His research interests include mixed-signal and deep-submicrometer circuit simulation, symbolic analysis, behavioral modeling for analog/RF-circuit applications, device modeling, and optimization algorithms.

C.-J. Richard Shi (M’91–SM’99–F’06) received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1994.

From 1994 to 1998, he was with Analogy (now part of Synopsys), the University of Iowa, and Rockwell Semiconductor Systems (now Conexant Systems). In 1998, he joined the faculty of the University of Washington, Seattle, where he is currently a Professor of electrical engineering, working in the area of computer-aided design and test of mixed-signal integrated circuits, as well as VLSI implementation of communication systems.

Dr. Shi is a contributor to the IEEE Std 1076.1-1999 (VHDL-AMS) standard for the description and simulation of mixed-signal circuits and systems. He founded the IEEE International Workshop on Behavioral Modeling and Simulation in 1997. He has been a recipient of several awards for his research including a Doctoral Prize from the Natural Science and Engineering Research Council of Canada in 1995, a Best Paper Award from the 1998 IEEE VLSI Test Symposium, a Best Paper Award from the 1999 IEEE/ACM Design Automation Conference, a National Science Foundation CAREER Award in 2000, and an SRC-TECHCON Best Paper Award in 2003. He was an Associate Editor, as well as a Guest Editor, of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING. He is currently serving as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS.