SILCA: SPICE-Accurate Iterative Linear-Centric Analysis for Efficient Time-Domain Simulation of VLSI Circuits with Strong Parasitic Couplings

Zhao Li, Student Member, IEEE, and C.-J. Richard Shi, Senior Member, IEEE
{lz2000, cjshi}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500
SILCA: SPICE-Accurate Iterative Linear-Centric Analysis for Efficient Time-Domain Simulation of VLSI Circuits with Strong Parasitic Couplings *

Zhao Li, Student Member, IEEE, and C.-J. Richard Shi, Senior Member, IEEE

Mixed-Signal CAD Research Laboratory
Department of Electrical Engineering, University of Washington, Seattle WA 98195
{lz2000, cjshi}@ee.washington.edu

Abstract: A new circuit analysis method, namely SPICE-accurate Iterative Linear-Centric Analysis (SILCA), is proposed for efficient and accurate time-domain simulation of deep-submicron VLSI circuits with strong parasitic couplings introduced by interconnect wires, common substrate, power/ground networks, etc. SILCA consists of several linear-centric techniques applied to classical time-domain nonlinear circuit simulation, and can be easily implemented to any SPICE-like simulator. For time-domain numerical integration, explicit-formula substitution and iterative-formula transformation are presented to convert implicit variable time-step integration to fixed leading coefficient time-step integration, so that equivalent conductance in companion models of charge storage devices is kept constant for time-domain simulation. We characterize both the convergence and stability properties of the resulting fixed leading coefficient integration formulae and describe techniques for adaptive step-size control. For nonlinear iteration, a successive variable chord method is introduced as an alternative to the Newton-Raphson method to achieve constant linearized conductance for nonlinear devices during nonlinear iteration. Furthermore, the low-rank update technique is implemented for fast LU factorization. With these techniques, the number and cost of required LU factorizations is reduced dramatically with SILCA. Experimental results on circuits coupled with substrate and power/ground networks have demonstrated that SILCA achieves SPICE-like accuracy. Further, more than an order of magnitude speedup over SPICE3 in terms of both the cost of LU factorization and overall CPU time has been observed for circuits with tens of thousands devices, and the efficiency increases further with the size of a circuit. SILCA is suitable for the accurate time-domain simulation of parasitic-sensitive very large-scale integrated circuits, where the number of linear parasitic devices dominates the number of nonlinear devices.

Keywords: Time-domain (Transient) Circuit Simulation, Strong Parasitic Coupling Effects, Fixed Leading Coefficient Numerical Integration Scheme, Successive Variable Chord Method

* 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 (SRC Task 921).
1. Introduction

With the increasing operation frequency, lower supply voltage and smaller device feature size, parasitic coupling effects are becoming more and more important for modern deep-submicron VLSI circuit designs [23]. The increasing demand to integrate digital, analog and radio frequency (RF) circuits into one single chip requires accurate analysis of very large scale integrated (VLSI) circuits together with extracted parasitic elements arising from interconnect lines, common substrate, power/ground networks, etc. [1][19][23][29][31]. Meanwhile on-chip and packaging inductances are no longer ignorable for accurate circuit analysis [8]. For such purposes as well as high fidelity coupled circuit and electromagnetic modeling [32], SPICE-like simulators are desirable for accurate transistor-level time-domain simulation.

However, efficient simulation of such systems presents a complexity challenge to SPICE [20]. For time-domain circuit simulation, SPICE uses numerical integration formulae [2][18] to form companion models for capacitors and inductors at each time point, and applies the Newton-Raphson method [18] to linearize nonlinear devices. Then the circuit is simulated at each time point by iteratively solving a system of linearized equations in the form of $Ax = b$, where $A$ is typically the so-called modified nodal analysis (MNA) circuit matrix [18][20], or the Jacobian matrix*. It is known that the cost of device evaluation dominates simulation of small to medium size circuits, and can be reduced with device bypass [14][20], table lookup [1], parallel computation techniques [13], etc. However, for a system with strong parasitic couplings, the per-iteration cost of SPICE time-domain simulation is dominated by LU factorization [18] of the circuit matrix $A$. In practice, the cost for LU factorization by sparse matrix solvers [12] is $O(n^{1.1-1.5})$ for sparse circuits, where $n$ is the circuit matrix size. However, strong parasitic couplings present in deep sub-micron circuits can cause the circuit matrix to become much denser and larger, even with model order reduction [21][27], the cost for LU factorization can approach its worst case $O(n^3)$ [23].

One key idea to improve the efficiency of SPICE-like circuit simulation is to keep the circuit matrix as constant as possible and therefore decrease the number of LU factorizations required during the entire time-domain simulation. This has been implemented in both the numerical integration and nonlinear iteration stages. For numerical integration, several strategies have been proposed on re-formulating the backward differentiation formulae (BDF) [2][18] to keep the leading coefficient constant since it is the leading coefficient that contributes to a Jacobian matrix. These include fixed coefficient methods [9] (i.e., in LSODE [24]), fixed leading coefficient methods [10] (i.e., in DASSL [22]), overdetermined polynomial methods (ODPM) [5], etc. All these methods have been shown to be effective in bypassing Jacobian

* Quasi-Newton iteration matrix is a terminology that is mathematically more precise to use throughout the paper. However, for simplicity and with a slight abuse of terminology, the terminology of Jacobian matrix is used.
matrix factorization. However, the stability of fixed coefficient and fixed leading coefficient methods is worse than that of variable coefficient methods [10]. Furthermore, for fixed coefficient and fixed leading coefficient methods, interpolation must be performed at each time point to approximate the solutions at a few evenly spaced previous time points, which will unfortunately introduce extra errors and increase simulation cost. The overdetermined polynomial method overcomes the interpolation problem by introducing an extra coefficient in the backward differentiation formulae. Although the overdetermined polynomial method was claimed to be more efficient, its stability is becoming worse and the ODPM-1 formula [5] has been shown to be stable only when the present time step-size $h_n$ is less than or equal to the predefined basis time step-size $h$. Therefore, a large basis time step-size $h$ has been adopted in [5] to have a $h_n/h$ ratio less than 1. The efficiency of the overdetermined polynomial method is thus limited.

To reduce the number of Jacobian matrix factorizations during the nonlinear iteration process, quasi-Newton methods [6][28] have been studied extensively and applied in circuit simulation [1] and mixed-mode circuit and device simulation [34]. Recently [1] explored the successive chord method [18], where each nonlinear device is modeled as a fixed linear resistor (called chord) combined with a variable nonlinear current source. Since a fixed chord is used, the circuit matrix will not change during nonlinear iteration and only one LU factorization is required overall if a fixed time step-size is used. Coupled with model order reduction and table lookup MOSFET models, this idea has been demonstrated to be effective for timing analysis of single-stage digital logic gates driving large-size parasitic networks [1]. Unfortunately, there are two principal difficulties that restrict the success use of this linear-centric idea to the simulation of general VLSI circuits: 1) Most VLSI circuits have widely distributed time constants, and require variable time step-size control for the simulation efficiency and accuracy. With variable step-sizes, the circuit matrix is no longer constant across time points unless a numerical integration formula as discussed before is used to keep the leading coefficient constant. 2) The successive chord method has a linear convergence rate. It may need excessive amount of iterations to converge. This implies that a huge number of forward/backward substitutions (FBSs) [18] may be required even for a single time point simulation, and thus offsets the gain from the reduction of LU factorizations.

Recently in the contexts of power grid analysis [4], substrate analysis [23], and parasitic extraction [11], krylov-subspace based iterative methods such as the conjugate gradient algorithm, generalized minimum residual (GMRES) algorithm, etc., have been shown to be more efficient than the method of LU factorization and forward/backward substitution (named the direct method). However, there is no report of successful applications of iterative methods to the classical problem of time-domain nonlinear circuit simulation in the literature, and industrial circuit simulators are virtually all based on direct methods.
This paper presents SILCA — SPICE-accurate Iterative Linear Centric Analysis — a new direct method capable of analyzing VLSI circuits containing strong parasitic coupling effects with SPICE-like accuracy yet orders of magnitude faster. SILCA consists of applying the linear-centric principle to both numerical integration and nonlinear iteration to keep circuit matrices as constant as possible during variable step-size time-domain nonlinear circuit simulation.

1) Two general techniques, namely explicit-formula substitution and iterative-formula transformation, are presented to convert implicit integration formulae in SPICE-like simulators to fixed leading coefficient integration formulae; these formulae lead to constant equivalent conductance of capacitor/inductor companion models. We note that most industrial SPICE simulators have used their own versions of implicit integration formulae that have been proven over the years by circuit designers.

2) Successive variable chord (SVC) method, a variant of the successive chord method, is introduced to keep linearized conductance of nonlinear devices constant for a larger voltage/current range by incorporating device-related behavioral knowledge. Together with the SVC method, a piecewise weakly nonlinear (PWNL) MOSFET model is introduced for the calculation of Jacobian matrices. The low-rank update technique is further applied for fast LU factorization by noting the fact that the number of nonlinear devices, switching operating regions defined by the PWNL MOSFET model, is only a few at a single time point.

With these, the required LU factorizations can be reduced by orders of magnitude with a small increase of iterations. As the result, rather than solving a newly linearized system by another costly LU factorization, we are able to achieve the same accurate results by several efficient forward/backward substitutions on a previously linearized system. The entire method is robust, accurate, and has been implemented into SPICE3. Further, the proposed method is compatible with other circuit analysis methods, such as model order reduction [21][27], to achieve even greater simulation speedup.

Some preliminary results of this paper were presented in [16]. This paper is organized as follows. Section 2 presents new fixed-leading-coefficient integration schemes, the analysis of their stability and convergence, and methods for adaptive step-size control. Section 3 presents the SVC method and the low-rank update technique. The SILCA algorithm is described in Section 4. Section 5 shows experimental results on substrate and power/ground coupling analyses. Section 6 concludes the paper.

2. Fixed leading coefficient integration schemes

In this section, we present and characterize two general techniques of taking any implicit integration formula to derive such an integration formula that yields a constant circuit matrix for variable step-size
time domain circuit simulation. Mathematically, let \( x_n, x_{n-1}, \ldots, x_0 \) and \( \dot{x}_n, \dot{x}_{n-1}, \ldots, \dot{x}_0 \) be the values and time derivatives of variable \( x \) at time points \( t_n, t_{n-1}, \ldots, t_0 \), then any linear multi-step numerical integration formula implemented in SPICE-like simulators can be written in the following general form,

\[
\dot{x}_n = \sum_{i=0}^{k} a_i x_{n-i} + \sum_{j=1}^{l} b_j x_{n-j}
\]

where \( a_i, i = 0, 1, \ldots, k \), and \( b_j, j = 1, 2, \ldots, l \), are coefficients of the integration formula, the leading coefficient \( a_0 \) is non-zero (hence implicit), and \( t_n - t_{n-1} \) is the current time step. Let \( h \) be some kind of basis time step-size, the current time step-size can be re-written as \( \alpha h \), and \( \alpha \) is a positive real number.

Only the leading coefficient \( a_0 \) contributes to the circuit matrix. In general, \( a_0 \) is a function of \( \alpha h \). Since \( \alpha \) changes with time points, the circuit matrix would change. To keep the circuit matrix constant, we re-write the integration formula above as follows:

\[
\dot{x}_n = a_0(h)x_n + a_0(\alpha h)x_n + \sum_{i=1}^{k} a_i x_{n-i} + \sum_{j=1}^{l} b_j x_{n-j}
\]

where \( a_0(h) \) is independent of \( \alpha \). The first technique is to replace \( x_n \) in the second term using an explicit integration formula of \( x_n \). This is called explicit-formula substitution. The second technique is to replace \( x_n \) in the second term using an initial guess, and then iterate to converge. This is called iterative-formula transformation. With these, the resulting formulae have a fixed leading coefficient, are referred to as fixed leading coefficient integration formulae, following the convention of Jackson and Sacks-Davis [10].

In Sections 2.1 and 2.2, we use the standard trapezoid formula as an example to derive fixed leading coefficient integration formulae based on explicit-formula substitution and iterative-formula transformation. We provide a characterization of the stability and local truncation error of the resulting integration formulae. We note that these derivation and analyses can apply to any implicit integration formula used in a circuit simulator. In Sections 2.3, 2.4 and 2.5, we present how the resulting formulae can be used in a way similar to the traditional predictor-corrector integration method, how to select the basis time step-size, and how to adaptively control the step-size for stability.

### 2.1 Fixed leading coefficient integration by explicit-formula substitution

Let \( h \) be a basis time step-size. The time step-size \( h_n \) for the present time point \( t_n \) can be represented by \( h_n = \alpha h \), where \( \alpha \) is a positive scalar. In the following, we derive the explicit-formula substitution based fixed leading coefficient integration formula in the context of the standard trapezoid formula as follows.
\[
\begin{align*}
\dot{x}_n & \approx \frac{2}{h} (x_n - x_{n-1}) - x_{n-1} \\
& = \frac{2}{ah} (x_n - x_{n-1}) - x_{n-1} \\
& = \frac{2\alpha - (2\alpha - 2)}{ah} (x_n - x_{n-1}) - x_{n-1} \\
& = \frac{2}{h} x_n - \frac{2}{h} x_{n-1} - \frac{2\alpha - 2}{ah} (x_n - x_{n-1}) - x_{n-1}
\end{align*}
\] (1)

where \(x_n = x(t_n)\), \(x_{n-1} = x(t_{n-1})\), \(t_n = t_{n-1} + \alpha h\), \(\dot{x}_n\) and \(\dot{x}_{n-1}\) are first-order time derivatives at \(t_n\) and \(t_{n-1}\), respectively. \textit{We would like to substitute the} \(x_n\) \textit{in the third term by all the known values from the previous time points.} This can be done using any explicit Adams-Bashforth formula [18]. The simplest is the forward Euler (FE) formula with a step-size \(ah\) defined as follows:

\[
x_n \approx x_{n-1} + ah \dot{x}_{n-1} \tag{2}
\]

After the \(x_n\) the third term in Eq. (1) is approximated by Eq. (2), the following \textit{mixed trapezoid-FE} formula with the time step-size \(ah\) is obtained:

\[
\dot{x}_n = \frac{2}{h} x_n - \frac{2}{h} x_{n-1} - \frac{2\alpha - 2}{h} (x_n - x_{n-1}) - x_{n-1} \tag{3}
\]

The mixed trapezoid-FE formula is an implicit integration formula. When \(\alpha = 1\), it reduces to the standard trapezoid formula. When \(\alpha = 1/2\), it represents the backward Euler (BE) formula with a step-size \(h/2\).

To see the circuit interpolation of the mixed trapezoid-FE formula Eq. (3), the companion model of a linear capacitor is shown as Fig. 1. Note that even the actual time step-size is \(ah\), the equivalent conductance of the companion model is \textit{constant} as long as the basis time step-size \(h\) is a constant. Similarly, a constant conductance companion model for an inductor can be constructed.

\[\text{Figure 1. The capacitor companion model using the mixed trapezoid-FE formula.}\]

The local truncation error (LTE) measures how closely a numerical integration formula approximates the differential operator. We can prove the following result:
**Theorem 1:** The local truncation error $\varepsilon$ of the mixed trapezoid-FE formula (3) with a time step-size $\alpha h$ is given by

$$
\varepsilon = \left(1 - \frac{1}{\alpha} \left(\frac{x_{n-1}}{2}\right)\right)(\alpha h)^2 + \left(1 - \frac{1.5}{\alpha} \left(\frac{x_{n-1}}{6}\right)\right)(\alpha h)^3
$$

where $t_{n-1}$ is between $t_n$ and $t_{n-1}$.

**Proof:** The proof is a straightforward application of the local truncation error estimation for the standard trapezoid formula [18], which is provided in Appendix I.

According to Theorem 1, when $\alpha = 1$, the LTE of the mixed trapezoid-FE formula reduces to that of the standard trapezoid formula. When $\alpha = 1/2$, it represents the LTE of the backward Euler formula using time step-size $h/2$. When $\alpha \to +\infty$, it approaches the LTE of the forward Euler formula. The mixed trapezoid-FE formula is a 2\textsuperscript{nd} order integration formula only if $\alpha = 1$ and degenerates to a 1\textsuperscript{st} order formula if $\alpha \neq 1$.

In contrast to local truncation error, stability is a global property related to the growth or decay of the local error introduced at each time point and propagated to the following time points. "Absolute stability" requires that $|x_n| < |x_{n-1}|$. It is often studied with the use of a simple test circuit as shown in Fig. 2. The stability property of the mixed trapezoid-FE formula can be proved as below:

**Theorem 2:** The absolute stability region of the mixed trapezoid-FE formula (3) with time step-size $\alpha h$ is defined by

$$
\frac{1 + (2\alpha - 1)z}{1 - z} < 1
$$

where $z = -h/(2 \tau)$ and $\tau$ is the time constant of a test circuit as shown in Fig. 2.

**Proof:** The proof is provided in Appendix II.

![Figure 2. A RC test circuit example.](image-url)
The absolute stability regions for $\alpha = 0.625$ and $\alpha = 2.5$ are shown in Figs. 3(a) and 3(b), respectively. From Theorem 2, several observations can be made on the stability of the mixed trapezoid-FE formula:

1) When $\alpha > 1$, the absolute stability region moves closer to that of the forward Euler formula. The mixed trapezoid-FE formula is not A-stable [18] or stiff stable [9][18], and cannot be used as a variable time step-size control scheme alone.

2) When $\alpha < 1$, the absolute stability region includes the open left half plane of the complex $z$-plane, and the mixed trapezoid-FE formula is A-stable. When $\alpha$ approaches $1/2$, the absolute stability region approaches that of the backward Euler formula. Further the smaller $\alpha$, the better the stability. However, according to Theorem 1, for a fixed time step-size $h_n = \alpha h$, a small $\alpha$ will unfortunately result in a large LTE. Therefore, there exists a tradeoff between the stability and the LTE.

We further note that when $\alpha < 1$, the mixed trapezoid-FE formula has a better “stiff decay” property [2] than the standard trapezoid formula. An integration formula is said to have the stiff decay property if $|x_n| \to 0$ when $z \to -\infty$. It means that a decent description of the solution in rapidly switching moments could be maintained in highly stiff cases, whereas the standard trapezoid formula generally encounters numerical oscillation phenomena.

### 2.2 Fixed leading coefficient integration by iterative-formula transformation

The LTE and stability problems of the mixed trapezoid-FE formula come from the replacement of the $x_n$ in the third term of Eq. (1) by an approximate $x_n$ defined in the explicit forward Euler formula Eq. (2). In this subsection, rather than using explicit integration formulae, the $x_n$ in the third term of Eq. (1) is substituted by the $(k-1)$-th iteration solution $x_n^{(k-1)}$ at the present time point and a new $k$-th iteration
solution $x_n^{(k)}$ is achieved by solving Eq. (1), where $k$ is the iteration number. This leads to the iterative-formula transformation of Eq. (1), called the *iterative trapezoid formula*, written as follows:

$$
x_n^{(k)} = \frac{2}{h} x_n^{(k)} - \frac{2}{h} x_{n-1} - \frac{2(\alpha - 1)}{ah} (x_n^{(k-1)} - x_{n-1}) - x_{n-1}
$$

$$
= \frac{2}{h} x_n^{(k)} - \frac{2}{h} x_n^{(k-1)} + 2 \frac{x_n^{(k-1)} - x_{n-1}}{ah} - x_{n-1}
$$

(4)

where $x_n^{(k)}$ and $x_n^{(k-1)}$ are the solution at the present time point for iteration $k$ and $k-1$ respectively.

For the implementation convenience, the companion model for a linear capacitor with the iterative trapezoid formula is shown in Fig. 4; Note that its equivalent conductance is a constant across time points if the basis time step-size $h$ is fixed.

![Figure 4. The capacitor companion model using the iterative trapezoid formula.](image)

The final solution is said to be converged if $|x_n^{(k)} - x_n^{(k-1)}|$ is smaller than a predefined error tolerance. If the iterative trapezoid formula (4) converges, the LTE requirement will be satisfied since the final converged solution is the same as that of the standard trapezoid formula.

In the following, we characterize both the convergence and stability properties of the iterative integration formula Eq. (4). To study the convergence property, let us write the linear(ized) circuit equation as [21]:

$$
Gx + C x = b
$$

where $G$ and $C$ represent the conductance and susceptance matrices, and $b$ is the vector of input sources. Replace time derivatives by the iterative trapezoid formula Eq. (4), we have

$$
G x_n^{(k)} + C \left( \frac{2}{h} x_n^{(k)} - \frac{2}{h} x_n^{(k-1)} + 2 \frac{x_n^{(k-1)} - x_{n-1}}{ah} - x_{n-1} \right) = b
$$

$$
\left( G + \frac{2C}{h} \right) x_n^{(k)} = \left( 1 - \frac{1}{\alpha} \right) \frac{2C}{h} x_n^{(k-1)} + 2 \frac{C}{ah} x_{n-1} + C x_{n-1} + b
$$

Clearly the trapezoid formula converges if
\[
\left\| \left( G + \frac{2C}{h} \right)^{-1} \left( 1 - \frac{1}{\alpha} \right) \frac{2C}{h} \right\| < 1
\]

where \( \| \| \) represents the spectral radius of the iteration matrix. The above inequality can be re-written as:

\[
\left| \frac{1 - 1/\alpha}{1 - z} \right| < 1
\]

where \( z = -h/(2\tau) \) and \( \tau \) is an eigenvalue of the matrix \( G^{-1}C \). For a test circuit as shown in Fig. 2, \( \tau \) represents the time constant. With this, we can show the following generalized convergence property:

**Theorem 3:** The convergence region of the iterative trapezoid formula Eq. (4) with a time step-size \( \alpha h \) is defined by Eq. (5).

**Figure 5. Convergence region of the iterative trapezoid formula for \( \alpha = 0.625 \) and \( \alpha = 2.5 \).**

From Eq. (5), the iterative trapezoid formula does not converge in the region of \( |z - 1| \leq |1 - 1/\alpha| \). To ensure the iterative trapezoid formula to converge for any decaying or stable oscillating system (Re(\( z \)) \( \leq 0 \)); i.e., to have the convergence region to includes all of the left half of the complex \( z \)-plane, we must choose \( \alpha > 0.5 \). In this case, \( |1 - 1/\alpha| < 1 \), the iterative integration formulae will not converge for a system with increasing components or unstable oscillation (Re(\( z \)) \( > 0 \)) in the region of \( |z - 1| \leq |1 - 1/\alpha| < 1 \). In practice, to speed up the convergence, \( 0.625 < \alpha < 2.5 \) is used in our experiments. More discussions on this can be found in Section 2.4. The convergence region for \( \alpha = 0.625 \) and \( \alpha = 2.5 \) is shown in Fig. 5. It represents the worst-case convergence region for \( 0.625 < \alpha < 2.5 \). In practice, a maximum iteration number limit for each iteration step is set. In case that the iteration number exceeds the maximum limit (due to either slow convergence or non-convergence), the iteration process with the same time step-size will be attempted one more time with the standard trapezoid formula before the time step-size is decreased. Therefore, even for a system with Re(\( z \)) > 0, the iterative integration scheme will bypass
the non-convergence region and finally achieve the correct solution. The penalty is that more iteration steps may be required.

Theoretically, an iterative implicit integration formula shall have the same stability as the corresponding original implicit formula if the iterative implicit integration formula is solved exactly (iterated to the infinity). However, in practice, the iterative implicit formula may be terminated after a few iterations. Very often, an error tolerance will be set to test the convergence, which leads to solutions within a finite number of iterations. In such case, the stability of the iterative formula can deteriorate. The stability of iterative trapezoid formula Eq. (3) can be characterized in the following theorem:

**Theorem 4:** The absolute stability region of the iterative trapezoid integration formula (4) with time stepsize $\alpha h$ is defined by

\[
\left\{ \frac{1 - 1/\alpha}{1 - z} k \left( \frac{2z}{z - 1/\alpha} + \frac{1/\alpha + z}{1/\alpha - z} \right) < 1 \right. 
\]

where $z = -h/(2\tau)$ and $\tau$ is the time constant for a test circuit as shown in Fig. 2.

**Proof:** The proof is provided in Appendix III.

![Figure 6](image_url)  
Figure 6. Absolute stability regions of the iterative trapezoid formula with $k=2$ for (a) $\alpha=0.625$ and (b) $\alpha=2.5$.

The absolute stability regions for $\alpha = 0.625$ and $\alpha = 2.5$ with $k = 2$ are shown in Fig. 6(a) and Fig. 6(b), respectively, which can be proved to satisfy the “stiff stability” requirements suggested by Gear [9]. For a fixed iteration number $k$, the absolute stability region of the iterative trapezoid formula will approach that of the standard trapezoid integration formula with $\alpha \to 1$. Furthermore, we prove the following stability property of the iterative trapezoid formula:

**Theorem 5:** When $k \to +\infty$ the absolute stability region of the iterative trapezoid formula (4) includes the entire open left half of the complex $z$-plane and excludes the entire right half of the complex $z$-plane.
Theorem 5 can be interpreted by noting the following fact: Eq. (4) is mathematically equivalent to applying the standard trapezoidal method with basis time step-size $h$ to the differential equation, and then using a quasi-Newton method [6][28] to solve the resulting equation. Therefore, when Eq. (4) is solved exactly ($k \to +\infty$), the absolute stability region of the iterative trapezoid formula will be the same as that of the standard trapezoid formula. This can also be verified by setting $k \to +\infty$ in the stability region definition of the iterative trapezoid formula in Theorem 4.

### 2.3 The predictor-corrector use of fixed-leading coefficient integration in SILCA

In SILCA, we first apply Eq (3) and then apply Eq (4) in a way similar to the predictor and corrector procedure. Using Eq. (3) to predict an initial guess for iterative trapezoidal formula Eq. (4) can lead to the fast convergence than using the previous time-point value as the initial guess for Eq. (4). However, it does not change the convergence region of Eq. (4) as defined by Theorem 3. Very often we may choose to carry Eq. (4) for one or a finite number of iterations, and then use the LTE to adjust the step size, similar to what is done in the classical predictor and corrector procedure [9]. In this case, applying Eq. (3) before Eq. (4) will lead to the stability region that is worse than that of applying Eq. (4) alone. We can prove the following result.

**Theorem 6:** The absolute stability region of applying Eq. (3) as a predictor and Eq. (4) as a corrector with time step-size $\alpha h$ is defined by

$$\left| \frac{1 - 1/\alpha}{1 - z} \right|^k \left( \frac{2\alpha z^2}{z - 1/\alpha} \right) + \frac{1/\alpha + z}{1/\alpha - z} < 1$$

where $z = -h/(2\tau)$ and $\tau$ is the time constant for a test circuit as shown in Fig. 2.

**Proof:** The proof is provided in Appendix III.

The absolute stability regions for $\alpha = 0.625$ and $\alpha = 2.5$ with $k = 2$ are shown in Fig. 7(a) and Fig. 7(b), respectively. It can be seen that when $\alpha = 0.625$, the absolute stability region of applying Eq. (3) followed by Eq. (4) is larger than that of applying Eq. (4) alone. However, when $\alpha = 2.5$, applying Eq. (3) followed by Eq. (4) becomes less stable than applying Eq. (4) alone, and it is no longer stiff stable. Theoretically, $k \to +\infty$, the absolute stability region of applying Eq. (3) followed by Eq. (4) will eventually reach that of the standard trapezoid formula. In practice, due to the finite iterations, applying Eq (3) followed by Eq (4) is less stable than the standard trapezoid formula especially if $\alpha > 1$. Therefore, in SILCA, to ensure the stability, Eq. (3) is applied as a predictor only if $\alpha < 1$. 

We note that applying Eq. (3) followed by Eq. (4) differs from the classical predictor and corrector procedure [9] since both Eq. (3) and Eq. (4) are \textit{implicit} integration formulae (leading coefficients are not zero), and thus has the better stability than the classical explicit predictor followed by implicit corrector (solved by fixed-point iterations). Although explicit integration formulae have been suggested as predictors in the classical predictor-corrector integration method [9], its effectiveness as a predictor is limited [18]. The reason is that due to the finite number of iterations used in the corrector, a good initial guess could be achieved with an explicit formula only if the present time step-size is small enough such that the stability is satisfied with the explicit formula. Otherwise, an initial guess with an unstable predictor would be even worse than the solution directly coming from the previous time point. This can be seen by comparing Fig. 6(b) to Fig. 7(b).

2.4 Illustration of basis time step-size (h) control

As discussed in Section 2.2, to satisfy the convergence property defined by Theorem 3, \(0.5 < \alpha < +\infty\) is required. The limited \(\alpha\) range means that it is impossible to use only one single basis time step-size during transient simulation in our framework. When \(h_n/h\) is out of the \(\alpha\) range, a new basis time step-size has to be chosen (i.e., the present time step-size \(h_n\)), which means the circuit matrix has to be updated and a new LU factorization is required. In this sense, a large \(\alpha\) range is preferred to decrease the total number of LU factorizations. However, according to Theorem 3, the linear convergence rate of the iterative trapezoid
formula is related to $|1-1/\alpha|$ and a smaller $|1-1/\alpha|$ means a faster convergence rate. Obviously, a small $\alpha$ range is ideal to reduce the total number of iterations. Therefore, in practice, $0.625 < \alpha < 2.5$ ($|1-1/\alpha| < 0.6$) is chosen to achieve a balance between the number of LU factorizations and the number of iterations, which will reduce the error to less than 5% of the original error after 6 iteration steps. Considering that SPICE3 needs at least 2 iteration steps to converge at a time point, the number of iteration steps with SILCA is approximately $3\times$ over that with SPICE3 for general circuits. The detailed basis time step-size control scheme will be described in Algorithm II of Section 4 and a linear circuit example is shown in Section 5.1 to illustrate the efficiency and validity of the iterative trapezoid formula.

![Figure 8. Convergence rate factor $|1-1/\alpha|$ vs. $\alpha$.](image)

It should be noted that SILCA could be combined with fixed time step-size methods to enlarge the range of $\alpha$. As shown in Appendix III, when $\alpha > 1$, the first iteration with the iterative trapezoid formula is equivalent to applying the standard trapezoid formula with the basis time step-size $h$. Then, one idea is to apply the standard trapezoid formula with the basis time step-size $h$ for multiple iterations if $\alpha$ is large. For example, if $\alpha = 3.5$, we could apply the standard trapezoid formula with the basis time step-size $h$ for the first two iterations and then the iterative trapezoid formula for the rest iterations with $\alpha = 3.5-1 = 2.5$. By this way, the convergence and stability properties will not be affected since the range of $\alpha$ for the iterative trapezoid formula is kept unchanged ($\alpha = 2.5$ for the previous example). Figure 8 shows the convergence rate factor $|1-1/\alpha|$ variation with the range of $\alpha$ by the iterative trapezoid method alone and its combination with a fixed time step-size method (Restart=$n$ means applying the fixed time step-size method for $n$ step-sizes). The extra cost is that more iterations will be required with more step-sizes performed by a fixed time step-size method.
2.5 Illustration of stability control

The iterative trapezoid formula satisfies the stiff stability [9][18], and is applicable to stiff circuits [18][33] as long as the circuit poles are close to the real axis of the complex-z plane, such as RC circuits. For oscillating circuits with poles close to the imaginary axis in the complex-z plane, according to Theorem 5, the absolute stability region of the iterative trapezoid formula will be the same as that of the standard trapezoid formula when the iteration number \( k \) goes to the infinity. However, in practice, the iteration number \( k \) is finite, the stability of the iterative trapezoid formula will become worse than that of the standard trapezoid formula. This can be illustrated using the linear RCL circuit example in Fig. 9.

![Figure 9. A linear RCL circuit example.](image)

The transfer function for both half RCL circuits in Fig. 9 is as below:

\[
H(s) = \frac{V_{out}}{V_{in}} = \frac{1}{RC^2 L \cdot s^3 + CL \cdot s^2 + 2RC \cdot s + 1}
\]

There are three poles for the right half circuit \(-0.5689, -0.2151 \pm j1.3071\), and three poles for the left half circuit \(-999999, -0.5 \pm j1000\). Noting that \( z = (h \lambda)/2 \), among these six poles, \(-0.5689, -999999\) and \(-0.2151 \pm j1.3071\) are on or close to the negative real axis of the complex \( z \)-plane, therefore they will not cause unstable problems since the iterative trapezoid formula has the stiff stability property. However, the rest two poles \(-0.5 \pm j1000\) are far away from the negative real axis and close to the imaginary axis of the complex \( z \)-plane, which may not be covered by the absolute stability region when the convergence is achieved in a small iteration number \( k \) (i.e., the blank region of the left half of the complex \( z \)-plane as shown in Fig. 6(a) and Fig. 6(b) for \( k = 2 \)). The simulation results with SPICE3 and SILCA (without the stability control) are shown in Fig. 10. Unstable simulation results are observed with SILCA and the number of iteration steps with SILCA is increased to \( 2X \) of that with SPICE3. It should be noted that backward differentiation formulae [18] with the order larger than 2 have the same stability problem for oscillating circuits.
This can be explained by comparing Fig. 5 and Fig. 6, which shows that the stability region is smaller than the convergence region. In other words, the stability requirement is much stricter than the convergence requirement. Despite the fact that the above oscillating case (such as circuits with poles close to the imaginary axis in the complex-$z$ plane) is rare in general circuit simulation, SILCA incorporates two practical measures to ensure the A-stability for all cases at the cost of more iterations and/or more LU factorizations: 1) Since the absolute stability region of an iterative trapezoid formula is related to the iteration number $k$ and the larger $k$ ensures the better stability, we set a minimum iteration number that the iterative trapezoidal formula should iterate. 2) A tighter $\alpha$ range is helpful since the absolute stability region will approach that of the standard trapezoid formula when $\alpha$ approaches 1.

To verify whether the absolute stability is ensured after a few iterations for the above case, we define the stability as the left part of the inequality in Theorem 4. Figure 11 shows the stability variation with the number of iterations for $\alpha=0.667$ and $\alpha=2.0$, respectively, when the basis time step-size $h$ is varied with $\lambda = -0.5+j1000$. The stability curves approach 1 after about 8 iterations for both cases. Therefore, in practice, a minimum iteration number limit is set to $(2+10*|1-1/\alpha|)$ so that enough iterations are performed if $\alpha$ is far away from 1.
3. Successive variable chord method

SPICE-like circuit simulators use the Newton-Raphson method for solving a set of nonlinear equations. Typically for each Newton-Raphson iteration, a new LU factorization is required. This can be extremely costly for a circuit with strong parasitic coupling effects or coupled with reduced dense linear networks. The successive chord method [18] always uses a fixed chord as the first order derivative during nonlinear iteration. Hence, at each time point, only one LU factorization is needed for nonlinear iteration. But it is often hard to choose a single fixed chord for a (strongly) nonlinear curve to always ensure a good convergence rate. In general, a chord that ensures global convergence will unfortunately lead to a very slow convergence rate.

To achieve a good balance between the number of LU factorizations and that of iterations, we propose the successive variable chord (SVC) method. The basic idea is to divide a nonlinear curve into different segments, each of which represents a weakly nonlinear curve and the same (local) chord is used for the same segment during nonlinear iteration – so-called Piecewise Weakly Nonlinear (PWNL) analysis. As shown in Fig. 12, the nonlinear curve is divided into three PWNL segments, three local chords are introduced, each representing the maximum derivative for the corresponding segment to ensure convergence. A new LU factorization is performed only if the nonlinear curve enters a different PWNL segment, and a new local chord is then used. By doing this, a similar convergence speed and accuracy can be achieved as the Newton-Raphson method while the number of LU factorizations can be decreased dramatically. We emphasize that the PWNL model of a nonlinear device is used only for the calculation of

![Figure 11. Stability variation with the number of iterations for (a) $\alpha=0.667$ and (b) $\alpha=2.0$.](image)
first-order derivatives, while the nonlinear function is still evaluated using the original nonlinear device model.

![Image](image.png)

**Figure 12. A PWNL example implemented with the SVC method.**

The PWNL idea implemented with the SVC method can be very effective due to the following facts: 1) MOSFETs in analog applications generally operate linearly around their operating points, only weakly nonlinearity properties may be present. A fixed chord representing the $g_m$, $g_{mbs}$, and $g_{ds}$ of MOSFETs at operating points is generally sufficient. A linear-centric harmonic balance analysis method has been proposed in [15]. 2) MOSFETs in digital applications reside in two regions at most time points—cutoff region and well-conducted linear region with a very small source-to-drain voltage, both regions have a relatively steady $g_m$, $g_{mbs}$, and $g_{ds}$. The only situation where $g_m$, $g_{mbs}$, and $g_{ds}$ change a lot is the time when MOSFETs switch from the cut-off region through the saturation region to the linear region (or vice versa). This process only occupies a small fraction of total simulation time for a MOSFET in a large-scale digital circuit. Furthermore, only a small fraction of MOSFETs switch their regions at one particular time point. Hence a fixed chord for these situations will not significantly affect the total iteration process.

With the considerations above, five MOSFET operating regions for digital circuit applications are defined as shown in Fig. 13, and $g_m$, $g_{mbs}$, and $g_{ds}$ for different operating regions are listed in Table I. In Table I, Reg#0 represents the cut-off region, Reg#1 and Reg#3 are saturation regions, and Reg#2 and Reg#4 are linear regions. $g_{m-max}$ and $g_{mbs-max}$ are the maximum values in all the regions (defined by $V_{dd}$), and $g_{ds-i}$ are defined (generally the maximum values) for different regions to ensure the convergence.

Notice that for Reg#1 and Reg#2 we set both $g_m$ and $g_{mbs}$ to zero. This can avoid frequent MOSFET switching between normal and reversed modes due to numerical errors, and thus further reduces the
number of required LU factorizations. On the other hand, setting $g_m$ and $g_{mbs}$ to zero has little effect on the convergence rate, since either $g_m$ and $g_{mbs}$ are small (Reg#2) or their contributions to the KCL equations are small (since $|V_{gs} - V|$ is small in Reg#1), and the contribution of $g_{ds}$ is dominant.

![Figure 13. Operating regions of MOSFET for digital applications.](image)

| Table 1. $g_m$, $g_{mbs}$, and $g_{ds}$ for different MOSFET operating regions. |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Reg#0 | Reg#1 | Reg#2 | Reg#3 | Reg#4 |
| $g_m$ | 0 | 0 | 0 | $g_{m,\text{max}}$ | $g_{m,\text{max}}$ |
| $g_{mbs}$ | 0 | 0 | 0 | $g_{mbs,\text{max}}$ | $g_{mbs,\text{max}}$ |
| $g_{ds}$ | 0 | $g_{ds,1}$ | $g_{ds,2}$ | $g_{ds,3}$ | $g_{ds,4}$ |

Another advantage of the SVC method is that chords can be pre-calculated before simulation, no derivative calculation is required during the iteration as in the Newton-Raphson method. This can lead to a significant saving in device loading time. Furthermore, table lookup models can be easily implemented in SILCA than in SPICE since there is no need of lookup tables for derivative calculation.

The proposed SVC method is more accurate and effective than ad hoc device bypass techniques utilized in modern circuit simulators [14][20], where device evaluations are bypassed when terminal voltages of a nonlinear device are kept almost constant for a few continuous nonlinear iteration steps. It has been reported that the voltage/current range of device bypass has to be kept small enough to avoid incorrect simulation results [20]. However, for high-frequency deep-submicron circuit applications, the efficiency is limited, since terminal voltages of most nonlinear devices are not completely constant but are changing slowly. The SVC method defines PWNL segments and local chords based on the behaviors of specific nonlinear devices under study, therefore it can keep the circuit matrix constant for a larger voltage/current range and requires much less LU factorizations.

By the above MOSFET operating region definition, only five sets of $g_m$, $g_{mbs}$, and $g_{ds}$ are used during time-domain simulation for digital systems. We further have the following observations: 1) At one time
point, most MOSFETs in a large digital system will stay in their operating regions as defined above, while only a few may switch from one region to another region. 2) For a switching MOSFET, the update of $g_m$, $g_{mbs}$, and $g_{ds}$ is region-wise. In other words, the change of $g_m$, $g_{mbs}$, and $g_{ds}$ from Reg#$i$ to Reg#$j$ is fixed. Therefore, in the case that a small amount of MOSFETs change operating regions, we can compute the new L and U matrices directly from the old L and U matrices using the low-rank update technique [7][30], rather than performing costly LU factorization of the entire updated circuit matrix.

Suppose that the present circuit matrix is $Y$, and one MOSFET is now switching from Reg#$1$ to Reg#$2$. The circuit matrix for the next iteration can be expressed by:

$$Y' = Y + cr^T$$

where $c$ and $r$ are sparse column vectors representing values of updated elements. In this case, $c = r = [0...0 e 0...0 –e 0...]^T$, and $e = \sqrt{(g_{ds-2} - g_{ds-1})}$. Noting that there are only four different elements between the matrix $Y$ and $Y'$, the new L and U matrices for $Y'$ can be updated from the previous ones for $Y$ efficiently with the low-rank update technique. The worst case complexity of $m$ low-rank update for a dense matrix is $O(m*n^2)$, where $m$ is the number of updated elements, $n$ is the matrix size. If $m$ is much less than $n$, the low-rank update will perform much faster than a regular LU factorization, whose worst case cost is $O(n^3)$ for a dense matrix. With the introduced PWNL MOSFET definition, $m$ will be kept small enough at a time point since the number of MOSFETs, whose terminal voltages change so violently that the operating region is switched, is generally small.

Figure 14. Matrix sparsity exploited by the low-rank update algorithm.

Furthermore, the low-rank update cost can be decreased dramatically by exploiting sparse matrix techniques [7][30] and nonlinear/linear circuit partitioning to place matrix elements due to nonlinear devices at the right-bottom corner of a circuit matrix [7]. In this way, only matrix elements whose values need to be updated are recomputed, while all other matrix elements are kept the same as before. For example, the circuit matrix $Y$ in Fig. 14 is partitioned into the linear part $Y_{lin}$, the nonlinear/linear coupling part $Y_{coup}$, and the nonlinear part $Y_{non}$. Whenever a nonlinear device changes its operating region (affecting four matrix elements of $Y_{non}$ in this example, marked by $\otimes$), $Y_{lin}$ and $Y_{coup}$ are kept the same. For the sparse matrix $Y_{non}$, only nine matrix elements need to be updated (marked by $\otimes$ and $\oplus$) and the other eight are...
kept unchanged (marked by \(\times\)). Therefore, the matrix sparsity can be fully exploited by the low-rank update technique.

Algorithm I. Rank-one update algorithm.

\[
Y = LU, \quad Y' = Y + cr^T
\]

\[
d = 1;
\]

\[
\text{for } i = 1, n \text{ continue; end}
\]

\[
\text{oldL} = L_{ii};
\]

\[
q = r_i \times d;
\]

\[
L_{ii} = L_{ii} + q \times c_i;
\]

\[
fact = L_{ii} / oldL;
\]

\[
p = d \times c_i / L_{ii};
\]

\[
c_i = c_i / oldL;
\]

\[
d = d - q \times p;
\]

\[
\text{for all non-empty } L_{ji} \text{ in column } i \text{ of } L
\]

\[
c_j = c_j - L_{ji} \times c_i;
\]

\[
L_{ji} = fact \times L_{ji} + q \times c_j;
\]

end

\[
\text{for all non-empty } U_{ij} \text{ in row } i \text{ of } U
\]

\[
r_j = r_j - r_i \times U_{ij};
\]

\[
U_{ij} = U_{ij} + r_j \times p;
\]

end

For the completeness of the paper, Algorithm I includes the sparse rank-one update algorithm. Note that all diagonal matrix elements are stored in the lower triangular matrix \(L\) and the upper triangular matrix \(U\) has unit elements on its diagonal. \textit{SILCA} utilizes sparse matrix solver package SPARSE1.3 [12] and a sparse low-rank update algorithm has been implemented successfully in SPARSE1.3. In our implementation, we also order the \(m\) low-rank updates heuristically based on the order of LU factorization to keep the condition number of the circuit matrix from becoming worse after low-rank updates. In addition, if the value of a diagonal element \((L_{ii})\) during low-rank updates becomes smaller than the predefined threshold value, the diagonal element will not be suitable for the following steps since it acts as a divider as shown in Algorithm I. In this case, a regular LU factorization is restored.

4. The \textit{SILCA} algorithm

The basic flow for \textit{SILCA} time-domain simulation is shown in Algorithm II. Practical considerations, such as breakpoints [20], are not included for clarity. A new LU factorization is only required if the
standard implicit integration scheme is used. In case that only local chords of nonlinear devices change, rank-one update is performed for fast LU factorization. No LU factorization is needed in any other case.

Nonlinear capacitors can be handled in SILCA by combining the proposed iterative trapezoid integration formula and the proposed successive variable chord, as illustrated as follows.

\[ Q_n^{(k)} = \frac{2}{h_n} (Q^{(k)} - Q_{n+1}) - \dot{Q}_{n+1} \]

\[ = \frac{2}{h_n} [Q^{(k-1)} + C_n^{(k-1)} (V_n^{(k)} - V_n^{(k-1)}) - Q_{n+1}] - \dot{Q}_{n+1} \]

\[ = \frac{2C_n^{(k-1)}}{h_n} V_n^{(k)} - \frac{2C_n^{(k-1)}}{h_n} V_n^{(k-1)} + \frac{2}{h_n} (Q^{(k-1)} - Q_{n+1}) - \dot{Q}_{n+1} \]

In the above derivation, a linearized capacitance \( C \) is introduced to represent the PWNL definition of the nonlinear capacitor. The basis step-size \( h \) is used as previously. For the clarity, the nonlinear charge is assumed to be the function of a single voltage. In practical MOSFET models, nonlinear charges are generally affected by three voltages \(-V_{gs}, V_{ds}, \) and \( V_{bs} \). For example, the nonlinear charge between the drain and the source of a MOSFET is \( Q_{ds} = Q(V_{gs}, V_{ds}, V_{bs}) \). Suppose that both linearized capacitors of \( Q_{ds} (C_{ds}, C_m \) and \( C_{mbs}) \) and linearized conductors of \( I_{ds} (g_{ds}, g_m \) and \( g_{mbs}) \) in a MOSFET need to be updated due to the switch of PWNL regions, the contribution of the MOSFET to the circuit matrix is as follows.

<table>
<thead>
<tr>
<th>S</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>( \Delta G_{ds} - \Delta G_{m}, \Delta G_{m} - \Delta G_{mbs}, \Delta G_{mbs} )</td>
</tr>
</tbody>
</table>

\[ \Delta G_{ds} = \frac{2 \cdot \Delta C_{ds}}{h} + \Delta g_{ds}, \quad \Delta G_{m} = \frac{2 \cdot \Delta C_{gs}}{h} + \Delta g_{m}, \quad \Delta G_{mbs} = \frac{2 \cdot \Delta C_{bs}}{h} + \Delta g_{mbs}, \quad a = \Delta G_{ds} + \Delta G_{m} + \Delta G_{mbs} \]

There are a total of 8 matrix entries to be updated. With the above representation, the rank-one update algorithm described in Section 3 can be used to realize fast LU factorization. In case that more matrix entries (at most 16 for a MOSFET) are affected by the switch of PWNL regions, a series of rank-one or rank-\( m \) updates \([3]\) are required to perform fast LU factorization; In this case, the efficiency of low-rank updates may be reduced.
Algorithm II. Transient simulation flow in SILCA.

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 {
    $\alpha = h_n/h$, iter_no = 0
    INNER LOOP: do {
      IF ($0.625 < \alpha < 2.5$) {
        IF ($\alpha < 1$ && iter_no == 0) {
          Apply Mixed Trap-FE Integration Predictor Eq. (3)
        }
        Apply Iterative Integration Corrector Eq. (4)
      } ELSE {
        IF (iter_no == 0) {
          $h = h_n$
        }
        Apply Standard Implicit Integration Scheme
      }
      Apply the SVC method on nonlinear devices
      IF ($0.625 < \alpha < 2.5$) {
        IF (chord is changed) { Apply Low-rank update & FBS }
        ELSE { Apply FBS }
      } ELSE { Apply LU factorization & FBS }
      iter_no = iter_no + 1
    } while (not converged)
    Choose a new $h_n$ based on LTE requirement
  } while (LTE greater than predefined error limit)
  $t = t + h_n$
}

5. Experimental results

Four sets of experiments are reported to demonstrate the validity and efficiency of the two introduced linear centric techniques. The first test uses a simple linear RLC circuit to demonstrate the proposed iterative integration scheme. The second test uses a variety of analog, digital and RF circuits with relatively small sizes to evaluate the effectiveness of the successive variable chord method implemented with the low-rank update technique. The last two examples are circuits coupled with substrate and power/ground networks, which are used to demonstrate the scalability of SILCA on larger circuits where the substantial portion of the circuits are linear parasitic devices. Level 1 model of MOSFETs is implemented with the proposed PWNL idea in SILCA, and nonlinear capacitors in MOSFETs are simplified as linear ones in both SILCA and SPICE3. To make a fair evaluation of the benefits of the proposed linear-centric techniques, no table lookup models of MOSFETs are used, and no RC(L) model order reduction algorithm is utilized. Some comparison results with a fast SPICE simulator HSIM and using an iterative linear solver are also included.
5.1 Evaluation of iterative integration scheme

The efficiency of the iterative integration scheme can be illustrated with a simple linear circuit example shown in Fig. 15. It includes two RCL circuits with different time constants. The input is a pulse signal (initially in the low voltage level 0V) with 50% duty ratio and 80 sec period. The simulation length is set to 160 sec. Since the minimum time constant is 0.01 sec for the left half RCL circuit, at least 16000 time points are required for a fixed time step-size simulation. Simulation results are shown in Table II, where \#Total points represents the number of total simulated time points and \#Accepted points represents the number of accepted time points. The rejected time points are those violating the LTE requirement or exceeding the maximum iteration limit. Since in SILCA a similar adaptive time step control scheme as that in SPICE3 is applied based on the LTE requirement, it can be seen from Table I that SILCA and SPICE3 achieve similar \#Total points and \#Accepted points, which are much less than that required by a fixed time step-size method. Furthermore, the number of LU factorizations used by SILCA decreases to 1.14% of that of SPICE3 (or 87.63X LU factorization cost saving). The number of iterations increases to about 2.5X. Figure 16 shows the histogram of the number of iteration steps, in which it can be seen that most of iteration converges in two to six steps. The simulation results with the GMRES algorithm (\(\varepsilon=1e^{-8}\)) are also included in Table II. The average number of GMRES iterations per GMRES solving process is about 2.

![Figure 15. A linear RCL circuit example.](image)

<table>
<thead>
<tr>
<th></th>
<th># Total points</th>
<th># Accepted points</th>
<th># Iteration</th>
<th># LU</th>
<th>#GMRES Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPICE3</td>
<td>2630</td>
<td>1965</td>
<td>5258</td>
<td>5258</td>
<td>–</td>
</tr>
<tr>
<td>SILCA</td>
<td>2645</td>
<td>1970</td>
<td>12831</td>
<td>55</td>
<td>–</td>
</tr>
<tr>
<td>GMRES</td>
<td>2640</td>
<td>1964</td>
<td>5278</td>
<td>59</td>
<td>9175</td>
</tr>
</tbody>
</table>
Figure 16. The histogram of the number of iteration steps.

Figure 17 shows the distribution of simulated time step-sizes ($h_n = \alpha h$) and the output waveform of $V_{out1}$ (a decaying oscillation waveform) during SILCA simulation. As shown in Fig. 17, the distribution of simulated time step-sizes is much denser when $V_{out}$ is close to 0. The reason is that the relative LTE for a low voltage level is small, so time step-sizes are restricted by the relative LTE and cannot change too much when $V_{out}$ is close to 0. It can be seen that most simulated time step-sizes are between 0.05 sec and 0.2 sec, centering around 0.08 sec. Recall that the iterative integration formulae can make a relaxation of $0.625 < \alpha < 2.5$, it is possible that fewer basis time step-sizes are required. Figure 18 shows the distribution of basis time step-sizes ($h$) during SILCA simulation. It can be seen that most basis time step-sizes are the same and near 0.08 sec.

Figure 17. Distribution of simulated time step-sizes and time-domain output waveform of $V_{out1}$. 

26
The histogram of basis time step-sizes with **SILCA** is shown in Fig. 19. First, it can be seen that only a few basis time step-sizes are used during transient simulation. Second, compared to Fig. 17, it can be concluded that most basis time step-sizes are near 0.08 sec and constant during the following time intervals: 10–40 sec, 45–80 sec, 80–120 sec, and 120–160 sec. Recall that in **SILCA** it is the basis time step-size that is used for MNA circuit matrix construction, the circuit matrix is now kept constant in these longer time intervals with the iterative integration scheme.

It should be noted that **SILCA** is mainly designed for speeding up circuit simulation in case that most of time step-sizes $h_n$ are close to the basis time step-size $h$, i.e., $0.625 < (h_n/h) < 2.5$ in our experiments. In general, for transient simulation of parasitic-sensitive circuits, most of time step-sizes are close to the basis time step-size.
time step-size for a relatively long time interval when the transient behavior of circuits does not change significantly, i.e., staying either in the logical “0” state or logical “1” state. In case that time step-sizes $h_n$ changes violently, a new basis time step-size $h$ will be chosen. However, based on our experiences, such chances are only a few (i.e., near break points). Further, SILCA can be combined with fixed time step-size methods to enlarge the range of $(h_n/h)$, as discussed in Section 2.4.

5.2 Evaluation of SVC and low-rank update

To illustrate the efficiency of the SVC and low-rank update techniques, simulations on several analog, digital, and RF circuits have been performed, and results are shown in Table III. It can be seen that the number of iterations generally increases to 1.5–2.5X of that with SPICE3. But the number of LU factorizations used by the SVC method with low-rank update decreases to 3%–20% of that used by SPICE3. We can see more saving in LU factorization with low-rank update for larger circuits, such as a 20-stage inverter chain, a ring oscillator and a VCO. In general, low-rank update technique will be more efficient for simulating a nonlinear circuit with a large-scale (potentially dense) network of linear elements used to model strong parasitic coupling effects, since only the L and U matrices for the sparse nonlinear part need to be updated during nonlinear iteration and the dense linear part remains unchanged. The average number of GMRES iterations per GMRES solving process is about 1 to 3.

It should be pointed out that although the number of LU factorizations is reduced dramatically with the SVC and low-rank update techniques, the speedup for circuits in Table III is not great since the simulation cost is dominated by device evaluation. As a relaxed direct method, SILCA has to take more device evaluations than SPICE3 since more iteration steps are required. For the Opamp follower example (including 32 MOSFETs, 8 capacitors, and 4 current sources), SPICE3 runs for 17.87 sec in which 12.06 sec is spent on device loading, while SILCA requires 18.65 sec with 13.89 sec on device loading. In this case, SILCA is more costly than SPICE3 since the simulation time is dominated by device loading. Therefore, although SILCA could be used for general-purpose circuit simulation, it is highly recommended to use SILCA for strongly coupled systems to achieve speed-up. For the same Opamp follower example, after a simple power/ground network (57 RCL devices) and a simple substrate network (164 resistors) are incorporated, SPICE3 needs 87.66 sec with 21.73 sec’s device load, while SILCA runs for 36.98 sec with 16.98 sec’s device load and achieves 2.37X speedup over SPICE3. It is expected that more speedup will be achieved if the surrounding environment is modeled by larger RCL networks and parasitic coupling effects are carefully considered, which is shown in Section 5.3 and Section 5.4. Furthermore, table lookup device models [1] or parallel computation techniques [13] can be used to reduce the cost of device evaluation, which have been widely applied in modern circuit simulators.
### Table III. Simulation results on test circuits.

<table>
<thead>
<tr>
<th>Test Circuits</th>
<th>#Total points</th>
<th>#Accept points</th>
<th>#Iter</th>
<th>#LU</th>
<th>#GMRES Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv</td>
<td>142</td>
<td>127</td>
<td>344</td>
<td>344</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>145</td>
<td>129</td>
<td>538</td>
<td>58</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>141</td>
<td>127</td>
<td>380</td>
<td>63</td>
<td>253</td>
</tr>
<tr>
<td>20-stage inverter chain</td>
<td>369</td>
<td>266</td>
<td>1193</td>
<td>1193</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>355</td>
<td>259</td>
<td>2384</td>
<td>56</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>357</td>
<td>259</td>
<td>2029</td>
<td>60</td>
<td>5275</td>
</tr>
<tr>
<td>Nand2</td>
<td>132</td>
<td>123</td>
<td>306</td>
<td>306</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>114</td>
<td>531</td>
<td>51</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>112</td>
<td>421</td>
<td>54</td>
<td>324</td>
</tr>
<tr>
<td>One-shot trigger</td>
<td>501</td>
<td>421</td>
<td>1525</td>
<td>1525</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>466</td>
<td>401</td>
<td>3511</td>
<td>181</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>505</td>
<td>421</td>
<td>2650</td>
<td>198</td>
<td>5971</td>
</tr>
<tr>
<td>Comparator</td>
<td>145</td>
<td>127</td>
<td>444</td>
<td>444</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>138</td>
<td>1251</td>
<td>58</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>138</td>
<td>1071</td>
<td>60</td>
<td>1354</td>
</tr>
<tr>
<td>Opamp follower</td>
<td>19812</td>
<td>13816</td>
<td>74216</td>
<td>74216</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>19686</td>
<td>13797</td>
<td>98318</td>
<td>11</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>19723</td>
<td>13785</td>
<td>91808</td>
<td>11</td>
<td>219717</td>
</tr>
<tr>
<td>Ring oscillator</td>
<td>243</td>
<td>173</td>
<td>1022</td>
<td>1022</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>251</td>
<td>180</td>
<td>2403</td>
<td>32</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>260</td>
<td>192</td>
<td>2250</td>
<td>38</td>
<td>5180</td>
</tr>
<tr>
<td>VCO</td>
<td>1506</td>
<td>1045</td>
<td>7621</td>
<td>7621</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1574</td>
<td>1118</td>
<td>18935</td>
<td>485</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>1137</td>
<td>16096</td>
<td>399</td>
<td>47609</td>
</tr>
</tbody>
</table>

*Note: For each circuit, the 1st row is the SPICE3 result, the 2nd row is the SILCA result, the 3rd row is the GMRES result.

#### 5.3 Coupled circuit and substrate parasitics analysis

The first example is a simple substrate coupling network as shown in Fig. 20, coupled with two inverters with pulse inputs in different operating frequencies – the first inverter operates at a low frequency and the second inverter operates at a high frequency. The bulk contacts of nMOSFETs are directly connected to P-substrate ports, and those of pMOSFETs are connected to P-substrate ports.
through a capacitor between the N-well and the P-substrate [25]. There are four other P-substrate ports connecting to the ground and the backplane of the substrate is also connected to the ground. RCL loads are added at the output of each inverter (not shown in Fig. 20). The substrate is modeled as a dense resistor network [31] consisting of a 3-dimensional dense resistor mesh with multiple layers. In Fig. 18, a one-layer resistor network is illustrated to model the substrate part among four inverter bulk contacts.

Although simplified truncated substrate models have been proposed to capture dominant coupling conductance [19][25], they are likely to underestimate coupling effects in circuit systems designed to be noise mature [23]. Furthermore, the accuracy with simplified substrate models may not be sufficient. Therefore, accurate analysis of a circuit with a fully modeled substrate is desirable for high fidelity circuit design and verification. Figure 21 shows the time-domain output waveform of the first inverter when the output signal is digital “1” (the high voltage level). First, the result from SILCA matches that from SPICE3. Second, it can be seen that high frequency feed-through signals from the second inverter are present in Fig. 21. This is an important first-pass design failure reason in deep-submicron digital and analog circuit designs, which may often not be captured by simplified substrate analysis.

Figure 21. Transient output waveform of the first inverter for the substrate coupling example.

Table IV is the statistics of running SILCA on a number of substrate analysis examples with varying circuit substrate network complexity compared to SPICE3. In our experiments, the number of layers and the number of resistors per layer are changed to vary the total number of circuit elements. A maximum 38.69X LU factorization cost saving and 17.30X overall speedup (with about 35 thousand elements) are achieved for this simple substrate coupling analysis example, and the cost of forward/backward substitution is increased to 2.5~2.75X. Figure 22 shows the run time comparison between SILCA and SPICE3 with the number of total circuit elements varied. No rank-one update technique is used here.
Several observations are as follows: 1) The larger the circuit is (therefore the larger LU/FBS cost ratio), the more overall speedup can be achieved with SILCA. SILCA is very suitable for deep-submicron circuit systems with strong parasitic coupling effects; 2) Device load cost with SILCA is decreased, which is proportional to the LU factorization cost saving. The reason is that in SILCA, MNA reloads are only performed when MNA elements need to be updated due to nonlinear devices and/or capacitors/inductors. For the substrate coupling examples, since most devices are resistors, their device loads are only performed when a new LU is required. 3) The more savings on LU factorization, the more iterations are required, which means more cost on forward/backward substitution and device evaluation. Therefore, there exists a tradeoff between the cost of the LU factorization and that of forward/backward substitution and device evaluation. The maximum overall speedup will approach the LU factorization cost saving (around 25~40X for this example) for large strongly coupled systems.

We also compare SILCA with a fast SPICE-like circuit simulator HSIM 1.3 [35] and the results are collected in Table V. HSIM 1.3 uses the backward Euler integration formula, a table lookup MOS level 2 model and device bypass techniques. Further, HSIMSPED=1 and HSIMALLOWEDDV=0.005 are set in HSIM 1.3 so that the number of total simulated time points is close to that of SPICE3 and SILCA to achieve the same accuracy. It can be seen from Table V that the larger the circuit is, the more speedup can be achieved with SILCA. Note that SILCA does not use table lookup MOSFET models.

![Figure 22. Run time comparison of the substrate coupling example.](image)
Table IV. Simulation results for the substrate coupling analysis example.

<table>
<thead>
<tr>
<th>#Layer x Res_Per_Layer</th>
<th>SPICE3</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>SILCA</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Iter</td>
<td>LU</td>
<td>FBS</td>
<td>Load</td>
<td>LU/FBS</td>
<td>#Iter</td>
<td>#LU</td>
<td>LU</td>
<td>FBS</td>
<td>Load</td>
<td>LU</td>
<td>Overall</td>
</tr>
<tr>
<td></td>
<td>(LU)</td>
<td>(sec)</td>
<td>(sec)</td>
<td>(sec)</td>
<td></td>
<td>(sec)</td>
<td>(sec)</td>
<td>(sec)</td>
<td>(sec)</td>
<td>(sec)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1x1281</td>
<td>12453</td>
<td>22.31</td>
<td>2.80</td>
<td>6.06</td>
<td>7.97</td>
<td>33157</td>
<td>375</td>
<td>0.62</td>
<td>7.85</td>
<td>0.64</td>
<td>35.98</td>
<td>3.42</td>
</tr>
<tr>
<td>2x1281</td>
<td>8522</td>
<td>131.94</td>
<td>7.58</td>
<td>12.16</td>
<td>17.41</td>
<td>21332</td>
<td>217</td>
<td>3.41</td>
<td>18.70</td>
<td>0.66</td>
<td>38.69</td>
<td>6.66</td>
</tr>
<tr>
<td>3x1281</td>
<td>9680</td>
<td>504.31</td>
<td>17.64</td>
<td>28.09</td>
<td>28.59</td>
<td>26623</td>
<td>273</td>
<td>14.28</td>
<td>49.64</td>
<td>1.52</td>
<td>35.32</td>
<td>8.41</td>
</tr>
<tr>
<td>4x1281</td>
<td>10280</td>
<td>1.374e3</td>
<td>32.07</td>
<td>50.96</td>
<td>42.84</td>
<td>28370</td>
<td>285</td>
<td>38.07</td>
<td>86.32</td>
<td>1.97</td>
<td>36.09</td>
<td>11.53</td>
</tr>
<tr>
<td>5x1281</td>
<td>8764</td>
<td>2.326e3</td>
<td>46.79</td>
<td>68.21</td>
<td>49.71</td>
<td>23666</td>
<td>251</td>
<td>118.08</td>
<td>3.10</td>
<td>35.11</td>
<td>13.02</td>
<td></td>
</tr>
<tr>
<td>6x4961</td>
<td>7020</td>
<td>1.231e5</td>
<td>1.492e3</td>
<td>1.490e3</td>
<td>82.50</td>
<td>17835</td>
<td>271</td>
<td>4.724e3</td>
<td>3.838e3</td>
<td>63.69</td>
<td>26.06</td>
<td>14.62</td>
</tr>
<tr>
<td>7x4961</td>
<td>9256</td>
<td>2.721e5</td>
<td>2.860e3</td>
<td>2.685e3</td>
<td>95.14</td>
<td>23147</td>
<td>312</td>
<td>8.933e3</td>
<td>7.014e3</td>
<td>99.96</td>
<td>30.46</td>
<td>17.30</td>
</tr>
</tbody>
</table>

Table V. Simulation results compared to HSIM 1.3

<table>
<thead>
<tr>
<th>#Layer x Res_Per_Layer</th>
<th>SPICE3 (sec)</th>
<th>HSIM 1.3 (sec)</th>
<th>SILCA (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1281</td>
<td>31.17</td>
<td>8.91</td>
<td>9.11</td>
<td>0.98</td>
</tr>
<tr>
<td>2x1281</td>
<td>151.68</td>
<td>43.55</td>
<td>22.77</td>
<td>1.91</td>
</tr>
<tr>
<td>3x1281</td>
<td>550.04</td>
<td>166.97</td>
<td>65.44</td>
<td>2.55</td>
</tr>
<tr>
<td>4x1281</td>
<td>1457.03</td>
<td>383.79</td>
<td>126.36</td>
<td>3.04</td>
</tr>
<tr>
<td>5x1281</td>
<td>2441.00</td>
<td>759.54</td>
<td>187.42</td>
<td>4.05</td>
</tr>
</tbody>
</table>

5.4 Coupled circuit and power/ground network analysis

![Figure 23. The power/ground analysis example.](image)

The second example is a power/ground network as shown in Fig. 23. The power and ground supply networks are modeled as two RCL mesh layers (parasitic coupling capacitors are not shown in Fig. 23). Between these two layers is a 20-stage inverter chain, different inverters of which are connected to different power/ground nodes. Furthermore, RCL loads are added to each inverter to model interconnect lines between adjacent stages.
Figure 24. Transient output waveform of the inverter chain for power/ground analysis example.

Figure 24 shows the time-domain output waveform of the inverter chain when the output signal is digital “1” (the high voltage level). The “1” signal has been disturbed due to the $IR$-drop and $L*dI/dt$ effects of the power/ground network (the input $V_{dd}$ is 3.3v). Table VI shows the simulation results with varied numbers of elements modeling the power/ground network. In our experiments, the sizes of two RCL meshes are changed to vary the number of elements. We can see that SILCA achieves more speedup for larger circuits. The number of iterations increases to 3.5~4.2X with SILCA. It is worthy noticing that, the maximum LU factorization cost saving and overall speedup reach 88.50X and 14.00X (with about 60 thousand elements), respectively, with the rank-one update technique, which are 19.82X and 8.86X, respectively, with only the SVC method.

Table VI. Simulation results for the power/ground analysis example.

<table>
<thead>
<tr>
<th>#Elems</th>
<th>SPICE3</th>
<th>SILCA</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o low-rank</td>
<td>with low-rank</td>
<td>w/o low-rank</td>
</tr>
<tr>
<td></td>
<td>#Iter</td>
<td>#LU</td>
<td>Overall</td>
</tr>
<tr>
<td>4002</td>
<td>14614</td>
<td>215</td>
<td>19.73</td>
</tr>
<tr>
<td>34802</td>
<td>16887</td>
<td>213</td>
<td>2.263e3</td>
</tr>
<tr>
<td>61602</td>
<td>18027</td>
<td>218</td>
<td>8.894e3</td>
</tr>
</tbody>
</table>
Figure 25. Circuit matrix structure (a) before and (b) after reordering for the power/ground example

Figure 25 (a) and Figure 25 (b) show the circuit matrix structure before and after LU factorization for a power/ground analysis example. It can be seen that the original circuit matrix before factorization is very regular and sparse (9618 elements in a 1177x1177 matrix, which means the sparsity is 0.70%), while the matrix after factorization becomes irregular and much denser (89733 elements, the number of which is increased 9.33X and the sparsity becomes 6.68%). The cost of LU factorization for the relatively dense matrix is much closer to its worst-case cost. Therefore, SILCA achieves great speedup over SPICE3 through saving a lot of LU factorizations.

For the comparison purpose, we have implemented a coupled iterative/direct solver for nonlinear circuits with large-scale power/ground networks [17]. In this coupled solver, power/ground networks are formulated with a nodal analysis (NA) circuit matrix [18], which is symmetric positive definitive, and solved by the conjugate gradient method with an incomplete Cholesky decomposition preconditioner [4]. Nonlinear circuits are formulated with a MNA circuit matrix and solved by the direct method based on LU factorization and Newton-Raphson iteration as in SPICE. The iterative method and direct method are coupled together by a Gauss-Seidel relaxation scheme [33]. Experimental results on the above power/ground analysis examples show that the coupled iterative/director solver achieved about the same speedup over SPICE3 as SILCA. However, it should be noticed that the coupled iterative/direct solver is efficient only if there exists a good partition with few boundary nodes between linear circuit parts and nonlinear circuit parts.

Very recently, we developed a new GMRES solver with a new LU factorization preconditioning scheme for time-domain simulation of nonlinear circuits with large-scale power/ground networks. The basic idea is to apply the same time step-size controlling scheme as that used in SILCA. Whenever time
step-sizes change violently, a new basis time step-size is chosen and a regular LU factorization is performed. If time step-sizes changes in the range of $0.625 < \alpha < 2.5$, rather than using linear centric analysis methods in SILCA, a GMRES solver is applied with the previous factorized L and U matrices as the preconditioner. Meanwhile, to make a fair comparison with SILCA, low-rank update will be applied to the preconditioning L and U matrices whenever a nonlinear device switches its operating region. The GMRES solver is implemented following the left-preconditioned GMRES algorithm in [25].

Simulation results with the new GMRES solver ($\epsilon=1\text{e-}8$) is shown in Table VII. It is seen that the average number of GMRES iterations ($\text{(#GMRES Iter)/(#GMRES)}$) with the LU factorization preconditioner is about 3–3.5 for a GMRES solving process, which shows that the preconditioner is very efficient. It is shown in Table VII that the speedup over SPICE3 with the GMRES solver is less than that with SILCA. The main reason is that the number of forward/backward substitutions with the GMRES solver ($\text{#Precond in Table VII}$) is generally larger than that with SILCA ($\text{#Iter in Table VI}$). Furthermore, extra costs due to matrix-vector product operations have to be taken during the GMRES solving process. It can be expected that the simulation cost will be increased if the error tolerance of the GMRES solver is made tighter. It should be noticed that the number of nonlinear iterations ($\text{#Tran Iter}$) is less than that with SILCA since there is no iterative integration scheme required for capacitors/inductors. However, the number of nonlinear iterations is larger than that with SPICE3 due to piecewise weakly nonlinear definition of MOSFETs.

### Table VII. GMRES simulation results for the power/ground analysis example ($\epsilon=1\text{e-}8$).

<table>
<thead>
<tr>
<th>#Elem</th>
<th>#Tran Iter</th>
<th>#Tran LU</th>
<th>#GMRES</th>
<th>#GMRES Iter</th>
<th>#Precond Tran LU (sec)</th>
<th>GMRES (sec)</th>
<th>Tot Tran (sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4002</td>
<td>6086</td>
<td>49</td>
<td>5945</td>
<td>19872</td>
<td>24766</td>
<td>4.42</td>
<td>114.12</td>
<td>132.48</td>
</tr>
<tr>
<td>34802</td>
<td>7083</td>
<td>51</td>
<td>6922</td>
<td>22140</td>
<td>27696</td>
<td>661.66</td>
<td>9572.16</td>
<td>10648.52</td>
</tr>
<tr>
<td>61602</td>
<td>7207</td>
<td>63</td>
<td>7000</td>
<td>22476</td>
<td>28019</td>
<td>2848.69</td>
<td>20549.69</td>
<td>24275.38</td>
</tr>
</tbody>
</table>

### 6. Conclusions

In this paper, a new nonlinear time-domain circuit simulation method called SILCA has been proposed for deep-submicron VLSI circuit design and verification, which requires accurate modeling of parasitic coupling effects or coupled circuit and electromagnetic modeling. A new variable time step-size iterative numerical integration scheme is developed to ensure constant equivalent conductance for capacitor/inductor companion models. We have characterized the convergence and stability properties of the new introduced integration formulae. As an alternative to the Newton-Raphson method, a successive variable chord method is proposed for nonlinear circuit simulation, and the low-rank update technique has been implemented for efficient LU factorization. With these techniques, SILCA can reduce dramatically
the number of costly LU factorizations for time-domain simulation. Experimental results on coupled circuit, substrate and power/ground network analysis have demonstrated that SILCA can achieve SPICE-like accuracy yet with orders of magnitude speedup over SPICE. Future research includes nonlinear capacitor handling, automatic generation of optimum PWNL models from nonlinear device models, exploiting incomplete LU preconditioners [25] for GMRES solving, and applications of SILCA to coupled electrical and thermal simulation where thermal effects are modeled as large linear RC networks.

Acknowledgements

The authors wish to thank Drs. Guoyong Shi, Pavel Nikitin and Alicia Manthe for their proof readings of this paper, Prof. Kartikeya Mayaram from Oregon State University, Dr. John Rockway from Naval Space and Warfare System Center, San Diego, for several helpful discussions on this research, and the anonymous reviewers for their valuable comments that greatly enhanced this paper.

Appendix I Proof of Theorem 1

Proof:

First, \( x_n \) and \( \dot{x}_n \) can be expressed by a Taylor series at \( t_n \) as follows (\( t_n = t_{n+1} + \alpha h \)):

\[
x_n = x_{n-1} + x_{n-1} \cdot (\alpha h) + \left( \frac{x_{n-1}}{2} \right) (\alpha h)^2 + \left( \frac{x_{n-1}}{6} \right) (\alpha h)^3 + O((\alpha h)^4)
\]

\[
\dot{x}_n = \ddot{x}_{n-1} + x_{n-1} \cdot (\alpha h) + \left( \frac{x_{n-1}}{2} \right) (\alpha h)^2 + O((\alpha h)^3)
\]

If the above first order derivative is used in the mixed trapezoid-FE formula (3), the approximated value of \( x_n \) can be expressed as:

\[
\hat{x}_n = x_{n-1} + (2\alpha - 1) \frac{h}{2} \dot{x}_{n-1} + \frac{h}{2} x_n
\]

\[
= x_{n-1} + (2\alpha - 1) \frac{h}{2} x_{n-1} + \frac{h}{2} \left[ \ddot{x}_{n-1} + x_{n-1} \cdot (\alpha h) + \left( \frac{x_{n-1}}{2} \right) (\alpha h)^2 + O((\alpha h)^3) \right] 
\]

\[
= x_{n-1} + x_{n-1} \cdot (\alpha h) + \left( \frac{\alpha x_{n-1}}{2} \right) (h)^2 + \left( \frac{\alpha^2 x_{n-1}}{4} \right) (h)^3 + O((h)^4)
\]

The error introduced at \( t_n \) is given by the difference between the exact value \( x_n \) and the approximate value \( \hat{x}_n \) computed by the mixed trapezoid-FE formula such that:
\[ e = x_n - \hat{x}_n = \left(1 - \frac{1}{\alpha} \right) \left( \frac{x_{n-1}}{2} \right) (\alpha h)^2 + \left(1 - \frac{1.5}{\alpha} \right) \left( \frac{x_{n-1}}{6} \right) (\alpha h)^3 + O((\alpha h)^4) \]

\[ = \left(1 - \frac{1}{\alpha} \right) \left( \frac{x_{n-1}}{2} \right) (\alpha h)^2 + \left(1 - \frac{1.5}{\alpha} \right) \left( \frac{x_{n-1}}{6} \right) (\alpha h)^3 \]

where \( t_\xi \) is between \( t_n \) and \( t_{n-1} \).

This proves Theorem 1. □

**Appendix II  Proof of Theorem 2**

*Proof:*

Applying the mixed trapezoid-FE formula (3) to a RC filter example as shown in Fig. 2, the following relationship can be derived (\( \tau = RC \)):

\[ \alpha x_n + x_n = 0 \]

\[ \frac{2\tau}{h} x_n - \frac{2\tau}{h} x_{n-1} - (2\alpha - 1) \alpha x_{n-1} + x_n = 0 \]

\[ \frac{2\tau}{h} x_n - \frac{2\tau}{h} x_{n-1} + (2\alpha - 1)x_{n-1} + x_n = 0 \]

\[ x_n = \frac{1 + (2\alpha - 1)z}{1 - z} x_{n-1}, \quad z = -\frac{h}{2\tau} \]

So, the absolute stability region of the mixed trapezoid-FE formula is defined by the following inequality:

\[ \left| \frac{1 + (2\alpha - 1)z}{1 - z} \right| < 1 \]

The proof of Theorem 2 is completed. □

**Appendix III  Proof of Theorem 4 and Theorem 6**

*Proof:*

Applying the iterative trapezoid formula (4) to a RC filter example as shown in Fig. 2, the following iterative relationship can be derived (\( \tau = RC \)):
\[ \alpha_n^{(k)} + x_n^{(k)} = 0 \]
\[
\frac{2\tau}{h} x_n^{(k)} - \frac{2\tau}{h} x_n^{(k-1)} + \frac{2\tau}{c h} (x_n^{(k-1)} - x_{n-1}) - \alpha_{n-1} + x_n^{(k)} = 0
\]
\[
\frac{2\tau}{h} x_n^{(k)} - \frac{2\tau}{h} x_n^{(k-1)} + \frac{2\tau}{c h} (x_n^{(k-1)} - x_{n-1}) + x_n + x_n^{(k)} = 0
\]
\[
x_n^{(k)} = \frac{1 - \frac{1}{\alpha}}{1 - z} x_{n-1}^{(k)} + \frac{1}{\alpha + z} x_{n-1}^{(k-1)} + \frac{1}{\alpha + z} x_{n-1}, \quad z = -\frac{h}{2\tau}
\]

Since the proof is for the stability property of the iterative trapezoid formula, the initial guess \( x_n^{(0)} \) of \( x_n \) is the solution of the previous time point \( x_n^{(0)} = x_{n-1} \). Then, the following derivation can be done:

\( k = 1 \)
\[
x_n^{(1)} = \frac{1 - \frac{1}{\alpha}}{1 - z} x_{n-1} + \frac{1}{\alpha + z} x_{n-1} = \frac{1 + z}{1 - z} x_{n-1}
\]
\( k = 2 \)
\[
x_n^{(2)} = \frac{1 - \frac{1}{\alpha}}{1 - z} x_n^{(1)} + \frac{1}{\alpha + z} x_{n-1} = \left( \frac{1 - \frac{1}{\alpha}}{1 - z} x_n^{(1)} + \frac{1}{\alpha + z} \right) x_{n-1}
\]
\[ ... \]
\( k = m \)
\[
x_n^{(m)} = \left( \frac{1 - \frac{1}{\alpha}}{1 - z} x_n^{(m-1)} + \frac{1}{\alpha + z} \right) x_{n-1}
\]
\[
= \left[ \left( 1 - \frac{1}{\alpha} \right) \frac{1 + z}{1 - z} + \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1}{1 - z} + \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1}{1 - z} + \ldots + \frac{1}{\alpha + z} \right] x_{n-1}
\]
\[ \quad (6) \]

Noting that the terms in the square bracket of Eq. (6) are a geometric series except the first term, it can be written further in the following format if \( \frac{1 - \frac{1}{\alpha}}{1 - z} \neq 1 \):

\( k = m \)
\[
x_n^{(m)} = \left[ \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1 + z}{1 - z} + \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1}{1 - z} \right] x_{n-1}
\]

According to Eq. (6), it is easy to check that the absolute stability condition cannot be satisfied if \( \frac{1 - \frac{1}{\alpha}}{1 - z} = 1 \). Therefore, the absolute stability region of the iterative trapezoid formula is then expressed by the following inequality:

\[
\left| \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1 + z}{1 - z} + \frac{1 - \frac{1}{\alpha}}{1 - z} \frac{1}{1 - z} \right| \frac{1}{1 - z} < 1
\]

38
Finally, we have the following result:

$$\left( \frac{1-1/\alpha}{1-z} \right)^n \left( \frac{2z}{z-1/\alpha} \right) + \frac{1/\alpha + z}{1/\alpha - z} < 1$$

This completes the proof of Theorem 4.

If the mixed trapezoid-FE formula is applied as an integration predictor when $\alpha < 1$, the absolute stability region of the iterative trapezoid formula is derived as follows.

$k = 1$

$$x_n^{(1)} = \frac{1 + (2\alpha - 1)z}{1 - z} x_{n-1}$$

... 

$k = m$

$$x_n^{(m)} = \frac{1-1/\alpha}{1-z} x_{n-1}^{(m-1)} + \frac{1/\alpha + z}{1-z} x_{n-1}^{(m-2)}$$

$$= \left[ \frac{1}{1-z} \left( 1 + (2\alpha - 1)z \right) \frac{1}{1-z} \left( 1 + (2\alpha - 1)z \right) + \frac{1}{1-z} \left( 1 + (2\alpha - 1)z \right) \right] x_{n-1}$$

Therefore, the absolute stability region of the iterative trapezoid formula is then expressed by the following inequality:

$$\left( \frac{1-1/\alpha}{1-z} \right)^n \left( \frac{2z}{z-1/\alpha} \right) + \frac{1/\alpha + z}{1/\alpha - z} < 1$$

This completes the proof of Theorem 6. □

References


