UNIVERSITY OF CALIFORNIA, SAN DIEGO

Efficient Large-Scale Transistor-Level Transient Analysis

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Zhengyong Zhu

Committee in charge:

Professor Chung-Kuan Cheng, Chair
Professor Alex Orailoglu
Professor J. Benjamin Rosen
Professor Fan Chung Graham
Professor Ernest Kuh

2005

.

The dissertation of Zhengyong Zhu is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2005

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Chung-Kuan Cheng for his support, supervision and guidance. In addition, I wish to express my appreciation to Professor Ernest Kuh of Berkeley. His supervision and advice on my research work is also extremely important to me.

I wish to thank my dissertation committee members, Professor Fan Chung Graham, Professor J. Benjamin Rosen and Professor Alex Orailoglu for technical discussions in numerical analysis algorithms and their advices and reviews of this dissertation.

I would like to acknowledge the support from Moazzem Hossain, Manjit Borah and Khosro Rouz of Fastrack Design. They provide tremendous help with the experimental environment.

I am grateful to all the graduate students in the UCSD VLSI CAD group for making the group a friendly and fun place to work. Among them, special thanks to Zhanhai Qin, Bo Yao, Hongyu Chen, Shuo Zhou, Jianghua Liu, Rui Shi, He Peng, Haikun Zhu, Yi Zhu, Ling Zhang and many others. Additional thanks goes to Rui Shi for her contribution in the splitting algorithm included in this dissertation.

I am deeply thankful to Barry Rubin and Alina Deutsch of IBM T.J. Watson Research Center. They provided me valuable intern opportunities and exposed me to a different field that complements my research work in school.

I would like to thank my parents and sister for their support and care ever since my childhood. I am also grateful to my little nephew for all the fun and laughter he brings to me. My final thanks to my wife, Dan Huang. My Ph.D. study could not have been completed without her endless support, patience, care and love.

VITA

2000	B.E. in Computer Science and Technology Tsinghua University, Beijing, China
2002	M.S. in Computer Science and Technology University of California, San Diego
2005	Doctor of Philosophy in Computer Science University of California, San Diego

PUBLICATIONS

Z.Zhu, M.Borah, K.Rouz, C.K.Cheng, E.S.Kuh, Efficient Transient Simulation for Transistor-Level Analysis, ADP-DAC 2005, pp. 240-243

Z.Zhu, B.Yao, C.K.Cheng, Power Network Analysis Using an Adaptive Algebraic Multigrid Approach, DAC 2003, pp. 105-108

H.Chen, C.K.Cheng, N.Chou, A.Kahng, J.MacDonald, P.Suaris, B.Yao, Z.Zhu, An algebraic Multigrid solver for analytical placement with layout based clustering, DAC 2003, pp. 794-799

Z.Qin, Z.Zhu, C.K.Cheng, Efficient Transient Analysis for Large Linear Networks", The 10th Workshop on Synthesis and System Integration of Mixed Technologies, Oct. 2001, pp. 293-300

FIELDS OF STUDY

Major Field: Computer Science
Studies in VLSI CAD
Professor Chung-Kuan Cheng

ABSTRACT OF THE DISSERTATION

Efficient Large-Scale Transistor-Level Transient Analysis

by

Zhengyong Zhu
Doctor of Philosophy in Computer Science
University of California, San Diego, 2005
Professor Chung-Kuan Cheng, Chair

With increasing design complexity, huge size of extracted interconnect data is pushing the capacity of transistor level simulation tools to the limits. Direct methods, such as LU decomposition used in Berkeley SPICE and its variations, are prohibitive due to the super linear complexity. In last decade, various numerical methodologies, such as circuit partitioning, fast linear solver, model order reduction, approximated device model, simplified numerical integration or linearization, piecewise linear waveform approximation and so on, have been introduced to improve the performance of simulation under the rising demand from advanced technologies. Although with significant runtime improvement, those methods usually trade off accuracy for speed. Inaccurate simulation results could lead to over-design that increases the product cost especially for nanometer high performance integrated circuit designs.

Inspired by the increasing gap between the design complexity and post-layout transistor-level simulation tools, we propose two efficient transistor-level analysis approaches with spice accuracy for deep-submicron and nanometer VLSI circuits:

1) Efficient technique to solve linearized circuit equation: A novel two-stage Newton-Raphson approach is implemented to dynamically model the linear network and nonlinear devices interfaces. Coupled linear networks are solved by the

adaptive algebraic multigrid method. The circuit latency and activity variations are well captured by adaptive strategies to greatly avoid the unnecessary repeated calculation. The proposed approach employs extra iterations between linear and nonlinear circuits inside the linearization process to ensure the global convergence.

2) New Numerical Integration Procedure: We propose a generalized operator splitting method for transistor-level transient analysis and demonstrate that the generalized method is unconditionally stable. Following the generalized approach, we partition the circuits and alternate the explicit and implicit numerical integrations between the partitions. The splitting algorithm is derived to significantly reduce the overhead during LU factorization. Thus the robust direct method still remains efficient for large-scale circuits.

Unlike the existing fast transistor-level simulation methods, both approaches proposed in the dissertation offer guaranteed simulation accuracy as well as significant runtime advantage. They can be used in post-layout transistor-level analysis of large-scale digital and mix-signal circuits.

Chapter I

Introduction

I.A Introduction

Electronic circuits require an accurate method to test circuit performance. Before computer simulation is available, electronic circuits were assessed by designing and fabricating discrete hardware breadboard circuits. Engineers then probed these breadboard circuits to locate unacceptable design causes and modified the circuits for design improvements. However, breadboard testing is impractical for large-scale integrated circuits due to the excessive design complexity and layout-specific parasitic effects. In addition, extensively probing integrated circuits to determine unacceptable design is unfeasible as circuit size keeps on shrinking. Computer simulation is necessary for integrated circuits before the expensive manufacturing.

The most well known circuit simulator is SPICE, which stands for "Simulation Program with Integrated Circuit Emphasis". The dawn of SPICE dated back to 1968, when Professor Ron Rohrer started a circuit simulation course project at Berkeley, which came out a software package named CANCER (Computer Analysis of Nonlinear Circuits Excluding Radiation). Later, Larry Nagel extended CANCER to SPICE as his Ph.D dissertation. Since then, several versions of SPICE (SPICE1, SPICE2, SPICE3 etc) were delivered. The latest version is SPICE3f5,

which is released in 1995. However, most numerical algorithms were finalized at SPICE2 in 1975. SPICE3 is just a version rewritten in C. Meanwhile, many industry variants are introduced and HSPICE is the leading product.

SPICE has capacity limitations due to its super linear complexity. It cannot perform analysis of large-scale circuits. With increasing design complexity, huge size of extracted interconnect data is pushing the capacity of transistor level simulation tools to the limits. Several fast SPICE simulation tools are already available. Although with significant runtime improvement, those tools usually trade off accuracy for speed. SPICE still remains the golden standard for circuit simulation.

I.B Spectrum of Simulation

Circuit simulation can be categorized in terms of level of detail offered, ranging from the detailed transistor level all the way to the Register Transfer Level (RTL). As illustrated in Figure I.1, simulation at different level has different accuracy requirement, which leads to different runtime performance and capacity. Each level has its constrains on the size of circuit it can simulate. The coarser the level, the larger the capacity and the better the performance. Usually circuit simulators are only targeting certain level of simulation. There is no universal simulator that can satisfy the requirements of every level.

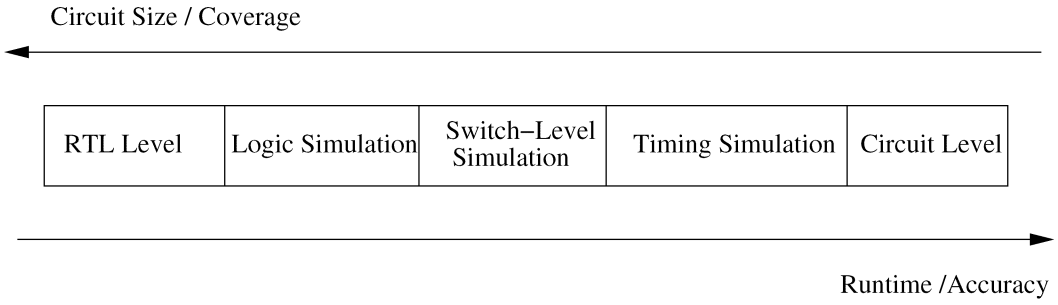


Figure I.1: Spectrum of Simulation

I.B.1 Transistor-Level Circuit Simulation

Circuit simulation at the transistor level is the most accurate analysis. Detailed device models especially those of transistors are provided in the simulation. The circuit behavior is derived from complicated numerical analysis techniques of significant runtime complexity. The size of circuits that typical transistor-level circuit simulation tools such as SPICE can handle is usually limited. Numerical algorithms involved in the transistor-level circuit simulation are introduced in Chapter II.

I.B.2 Timing Simulation

Timing simulation is used for verifying the timing behavior of circuits and usually acts as sign-off tool when full-chip transistor level verification is prohibitive. In general, there are two kinds of timing simulations: dynamic and static.

Dynamic timing simulation is similar to the transistor-level circuit simulation, except that there are many simplified assumptions on the device model, numerical integration and linearization process. Piecewise linear waveform approximation and event-driven techniques are adopted in many dynamic timing analysis tools.

Unlike dynamic analysis, the static timing analysis (STA) is a vectorless method. It finds the worst or best-case scenarios in terms of timing. The circuit is mapped to a timing graph, where each node represents a gate and each edge represents an interconnect wire. Delay model is associated with nodes and edges in the timing graph. The delay model for gates is usually a pre-calculated table, which is a function of load capacitance and input slope. The algorithm computes the arrival time and required time of signals through the timing graph. The difference between arrival time and required time at each node in the timing graph is called slack, which indicates the magnitude of timing violation. When logic correlation is ignored, the extreme timing violation case detected by static timing analysis may never happen in real world. The false timing violation path

is referred as false-path. In order to avoid over-design, static timing analysis tools have to remove those false paths and find the worse or best-case scenarios as close to the real situation as possible. Another consideration is the cross-talk impact on wire delay, which is usually captured by iterative timing window refinements.

I.B.3 Switch-Level Simulation

In switch-level simulation, each transistor is modelled as an ideal switch. Switch-level simulation can be very efficient for large-scale circuits. However, it provides limited timing information because transistors are approximated ideal switches. Signals on wires are only represented as 0, 1, X (unknown), and at a variety of drive strengths corresponding to the W/L of the transistor driving the net. Switch-Level simulation may have trouble with analog circuits such as sense amplifiers.

I.B.4 Logic Simulation

Logic simulation verifies the correct logical operation of a design. Gates instead of transistors are used for simulation. Logic simulator can only simulate Boolean gates, which exclude transmission gates, switch logic, pseudo-NMOS and dynamic logic.

I.B.5 Register Transfer Level(RTL) Simulation

Register transfer level simulation allows the user to verify or simulate a description at the system or chip level. Higher-level functions realized by a collection of gates are represented as the function itself in the register transfer level simulation. There is no transistor or gate information; only functions are evaluated. The primary interest of RTL simulation is the state transitions.

I.C Types of Circuit Level Simulation

I.C.1 DC Operating Point Analysis

DC operating point analysis is the starting point of all other types of analysis. It establishes the DC bias point of the circuit. The solution at DC point is the initial condition for all other analysis.

In DC analysis, only DC source is considered. Capacitors are modelled as open circuits and inductors are viewed as ideal short circuits. The circuit equation is solved at every Newton-Raphson iteration. Since nonlinear convergence may be difficult to achieve for some circuits at the DC stage, SPICE provides various DC convergence aid options to help locate the DC bias point.

I.C.2 DC Transfer Analysis

DC transfer analysis is actually a series of DC operating point analysis. The DC source (current or voltage) is changed gradually and DC operating point analysis is performed at every stepped DC source values. The solution at the current source value can act as the initial guess for the next DC value. DC transfer curves are useful to determine the noise margins and large-signal characteristics of analog circuits.

I.C.3 Frequency Domain AC Analysis

AC analysis is used for analog circuit simulations. The circuits are simulated at a series of frequency values using phaser method to determine the frequency response of circuit transfer functions. Small-signal models for transistors and diodes are used in AC analysis, which means nonlinear effects such as distortion and saturation are ignored during frequency sweep. The circuit equations use complex variables instead of real variables in other types of analysis.

I.C.4 Time Domain Transient Analysis

Transient analysis is the most widely used analysis. The circuits are repeatedly solved over a period of time, which is divided into discrete time points. Storage elements like capacitors and inductors are replaced with equivalent circuits using numerical integration. The solution at the current time point acts as the initial guess for the next time point. Transient analysis provides current and voltage waveforms as output. Dynamic time step control and nonlinear convergence check ensure that the final result is accurate within the error tolerance.

I.C.5 Transmission Line Simulation

The circuit elements in analysis types introduced above are all lumped elements. However, at relatively higher frequencies, lumped models become inaccurate and distributed quasi-TEM models based on Telegrapher's equations become necessary. This is because the interconnect time-of-flight (delay) has become comparable to signal transition times for longer interconnect at higher frequency. Different transmission line models are available. The simulation of transmission line requires different numerical techniques such as convolution for distributed element model evaluation.

I.C.6 RF Simulation

Unlike digital circuits, electronic components operating at radio frequencies such as amplifiers, mixers, oscillators, voltage-controlled oscillators (VCOs), and phase-locked loops (PLL's) or Radio frequency integrated circuit (RFIC) requires different kinds of simulation features. Because of the signal frequencies, the circuit time-constants, and the types of analysis required, simulators such as SPICE are neither adequate nor appropriate [26]. Though the time-domain simulation in SPICE can still simulate the steady state analysis, the runtime is not acceptable because it has to wait until the circuit reaches the steady state. Special-purpose simulation tools have been developed to address the needs of RF simulation. Differ-

ent techniques are used for RF simulation, such as harmonic balance [18], Shooting method [42] and Envelop method [27].

I.D Review of Fast Spice Techniques

With increasing design complexity, huge size of extracted interconnect data is pushing the capacity of transistor level simulation tools to the limits. Direct methods, such as LU decomposition used in Berkeley SPICE [28] and its variations, are prohibitive because of the super linear complexity $O(n^{1.5})$ where n is the number of circuit nodes.

In last decade, there is rich literature in the field of circuit analysis to improve the performance of simulation under the rising demand from advanced technologies. In general, those fast simulation techniques can be classified into the following categories:

Parasitic Reduction Huge volume of the parasitic data generated from the post-layout parasitic extraction raises challenges for the memory consumption as well as the complexity of numerical analysis algorithms. Fast spice simulators usually apply parasitic reduction to reduce the problem size with a negligible loss in accuracy. Though RC reduction is quite common, reduction of inductance is rarely observed in commercial tools. This is because the on-chip inductance effect is still not an issue among the top priorities and RLC reduction is far more complicated than RC reduction. Recent work in the model order reduction has been focused on generating stable and passive macromodels [11, 17, 24, 32, 33, 35, 37].

Circuit Partitioning Most of the contemporary commercial tools are based on circuit partitioning, such as HSIM of Nassda [54], Power Mill/Time Mill/Rail Mill and NanoSim [55–58] of Synopsys, RedHawk [59] of Apache, UltraSim/VoltageStorm of Cadence [60, 61] and so on. SAMSON2 [36] further saves unnecessary computation by applying different integration method (ex-

plicit or implicit) on subcircuits according to their activities. Waveform Relaxation (WR) [22, 31, 47] is an iterative method that attempts to partition the circuit into subcircuits and repeatedly solve subcircuits by the relaxation of the nodal voltage waveform over time intervals.

Circuit partitioning can also reuse computation for identical subcircuits. One typical example is the isomorphism technique used in Nassda's HSIM for memory cells.

Though subcircuits have smaller size and can be simulated with different time step to explore latency, one practical issue of the partition-based methods is the potential convergence problem due to the coupling effect or strong feedback. The convergence rate is also sensitive to the partition algorithm and propagation order.

Fast Matrix Solver Fast matrix solvers such as preconditioned conjugate gradient method [14] and multigrid method [4] have been applied to the large linear network analysis [5, 19, 30, 49]. GMRES [46] can be used to solve medium size nonlinear circuits such as RF and analog components [42].

Piecewise Linear Waveform Approximation and Event Driven Some timing simulators [10, 13, 25, 44] employ piecewise approximation of nodal voltages, branch currents or I-V curves. Among them, SPECS [13, 44] uses piecewise constant I-V curves. In SWEC [25], branch conductance is approximated by piecewise constant functions. In ACES [10], I-V curve is approximated by piecewise linear functions. Moreover, ACES and SPECS employ explicit integration method; hence avoid the expensive matrix decomposition operation. Event Driven techniques often appear with piecewise linear approximation methods.

Simplified Linearization Process Successive chord method is used in [1, 23] as an alternate to Newton-Raphson method. The advantage of successive chord method is that the Jacobian matrix is kept unchanged in a range to

reduce the number and cost of LU decompositions. It works well for low-level device models, but will introduce significant errors for advanced detailed device models such as BSIM3 and BSIM4.

Fast Transistor Device Model Evaluation Transistor device model evaluation is time-consuming for transient analysis when detailed device models such as BSIM3 and BSIM3 are used. It can easily take up more than 30% of the total runtime. Many commercial fast transistor level simulation tools offer option of device table model, which saves pre-calculated device evaluation data for later use. However, table model is not accurate enough for nanometer designs. Another option is to simplify the MOSFET gate capacitance model only and keep the detailed MOSFET dynamic conductance model because the MOSFET capacitance evaluation takes most of the device evaluation time. One possible approximation is to replace the voltage-dependent MOSFET capacitances by its average and ignore the Miller Effect between gate-to-drain and gate-to-source terminals.

I.E Dissertation Overview

Inspired by the increasing gap between the design complexity and post-layout transistor-level simulation tools, we propose two efficient transistor-level analysis approaches with SPICE accuracy for deep-submicron and nanometer VLSI circuits in this dissertation.

Chapter II introduces numerical analysis techniques evolved in the SPICE transient analysis flow, including matrix formulation, solving ordinary differential equations, solving nonlinear equations and systems of linear equations. Besides LU decomposition used in SPICE, iterative matrix solvers such as preconditioned conjugate gradient(PCG), GMRES and multigrid are also discussed.

In Chapter III, we present an efficient technique to solve linearized circuit equation. A novel two-stage Newton-Raphson approach is implemented to decou-

ple the linear and nonlinear portions of circuits inside the inner most Newton-Raphson iteration and different portion is solved by different matrix solving techniques according to its matrix properties. This chapter presents the linear-nonlinear interfaces modelling as well as the modified transient analysis flow. After a brief review of algebraic multigrid theory, special grid construction and error smoothing for general circuits with inductors and adaptive strategies are discussed in detail. The runtime performance is then demonstrated by experimental results.

Chapter IV introduces a new numerical integration procedure. We present a generalized operator splitting method and demonstrate that the generalized method is unconditionally stable. The local truncation error is derived to dynamically estimate the time step size. A splitting algorithm for general circuits is proposed to significantly reduce the overhead of LU factorization. Hence, the robust direct method still remains efficient for large-scale circuits.

The final chapter concludes with a brief summary of the contributions made in this dissertation.

Chapter II

Background

This chapter briefly introduces the numerical analysis techniques evolved in the SPICE transient analysis flow, including matrix formulation, numerical integration, solving nonlinear system and sparse matrix techniques. Those fundamental numerical algorithms are quite essential for better understanding of the new transient analysis approaches proposed in the following chapters.

II.A Circuit Transient Analysis System Matrix Formulation

Circuits are collections of various kinds of basic circuit elements such as resistor, capacitor, inductor, MOSFET device, voltage source, current source and so on. All those elements are stamped into the system equations according to their constitutive equations and the conservation laws (Kirchhoff's Current Law and Kirchhoff's Voltage Law). Table II.1 lists the constitutive equations of some basic circuit elements.

Table II.1: Basic Element Constitutive Equation

Elements	Constitutive Equation
Resistor R	$v = Ri$
Capacitor C	$i = C\frac{dv}{dt}$
Inductor L	$v = L\frac{di}{dt}$

Among several matrix formulation methods, Sparse Tableau Analysis (STA) and Nodal Analysis (NA)/Modified Nodal Analysis (MNA) are most widely used. The dissertation work formulates the matrix using Modified Nodal Analysis; hence, only Nodal Analysis and Modified Nodal Analysis are introduced here.

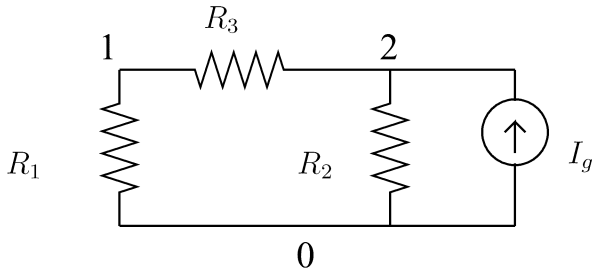


Figure II.1: Matrix Formulation by Nodal Analysis

Nodal Analysis The Nodal Analysis lists KCL equation at every node. The system matrix dimension equals the number of nodes in the circuits. Though the formulation of nonlinear devices and energy storage elements require linearization or numerical integration discussed later, linear resistor is sufficient to explain the Nodal Analysis. Usually, those elements are stamped into matrix one by one by inspection. Each element has a sub-matrix template to fill in values. The template of resistor can be found in Table II.2. The Matrix formulation for a resistor circuit in Figure II.1 is derived as:

$$\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_3} & -\frac{1}{R_3} \\ -\frac{1}{R_3} & \frac{1}{R_2}+\frac{1}{R_3} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ I_g \end{bmatrix} \tag{II.1}$$

Modified Nodal Analysis The limitation of Nodal Analysis is that elements without admittance form such as voltage source, voltage control voltage source (VCVS), current control current source (CCCS), current control voltage source (CCVS) and inductor cannot be represented in the Nodal Analysis formulation. This is resolved by introducing extra branch current unknowns in addition to the nodal voltage variables in the Nodal Analysis. The new formulation is called Modified Nodal Analysis (MNA) [16]. Detailed MNA stamping templates for various circuit elements are given in Table II.2.

II.B Numerical Integration

In transient analysis, the circuit equation, which is actually an ordinary differential equation (ODE), is solved at discrete time points selected over the entire simulation interval. The final waveforms are actually the interpolation of those discrete time points.

Unlike resistors, the energy storage elements such as capacitor and inductor cannot be stamped into the matrix directly due to their differential constitutive equations. Instead, those differential constitutive equations are integrated numerically at every discrete time point. The resulting equivalent circuit models for energy storage elements contain only resistors and current/voltage sources, which can be stamped into system matrices directly.

II.B.1 Numerical Integration of Capacitor

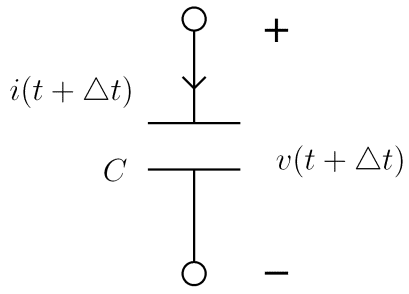


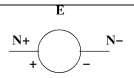
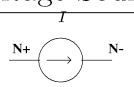
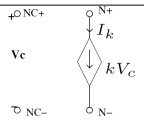
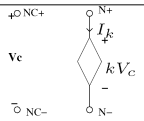
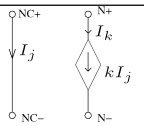
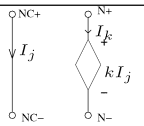
Figure II.2: Circuit Model of Capacitor

In order to derive a linear relation of current and voltage across the capacitor shown in Figure II.2 from the differential constitutive equation (II.2), the current is integrated over a time interval $\triangle t$ in Equation (II.3).

$$i = C\frac{dv}{dt} \tag{II.2}$$

$$v(t+\triangle t) = v(t) + \frac{1}{C}\int_{t}^{t+\triangle t} i(\tau)d\tau \tag{II.3}$$

Table II.2: Basic Circuit Element Stamping Templates

Circuit Element	Matrix Stamp	Equation
 Resistor	$\begin{array}{c\|cc} & V_{N+} & V_{N-} \\ \hline N+ & \frac{1}{R} & -\frac{1}{R} \\ N- & -\frac{1}{R} & \frac{1}{R} \end{array}$	$i = (v_{N+} - v_{N-})/R$
 Voltage Source	$\begin{array}{c\|ccccc} & V_{NC+} & V_{NC-} & V_{N+} & V_{N-} & I_k \\ \hline NC+ & & & & & 0 \\ NC- & & & & & 0 \\ N+ & & & & & 1 \\ N- & & & & & -1 \\ \hline & -k & k & 1 & -1 & \end{array}$	$v_{N+} - v_{N-} = E$
 Current Source	$\begin{array}{c\|c} & RHS \\ \hline N+ & -I \\ N- & I \end{array}$	$i_{N+} = -i_{N-} = I$
 VCCS	$\begin{array}{c\|cc} & V_{NC+} & V_{NC-} \\ \hline N+ & k & -k \\ N- & -k & k \end{array}$	$i_j = k(V_{NC+} - V_{NC-})$
 VCVS	$\begin{array}{c\|ccccc} & V_{NC+} & V_{NC-} & V_{N+} & V_{N-} & I_k \\ \hline NC+ & & & & & 0 \\ NC- & & & & & 0 \\ N+ & & & & & 1 \\ N- & & & & & -1 \\ \hline & -k & k & 1 & -1 & \end{array}$	$-k(V_{NC+} - V_{NC-})$ $+V_{N+} - V_{N-} = 0$
 CCCS	$\begin{array}{c\|ccccc} & V_{NC+} & V_{NC-} & V_{N+} & V_{N-} & I_j \\ \hline NC+ & & & & & 1 \\ NC- & & & & & -1 \\ N+ & & & & & k \\ N- & & & & & -k \\ \hline & 1 & -1 & & & \end{array}$	$V_{NC+} - V_{NC-} = 0$ $i_j = ki_k$
 CCVS	$\begin{array}{c\|cccccc} & V_{NC+} & V_{NC-} & V_{N+} & V_{N-} & I_j & I_k \\ \hline NC+ & & & & & 1 & \\ NC- & & & & & -1 & \\ N+ & & & & & & 1 \\ N- & & & & & & -1 \\ \hline & 1 & -1 & & & & \\ & & & 1 & -1 & -k & \end{array}$	$V_{NC+} - V_{NC-} = 0$ $V_{N+} - V_{N-} - kI_j = 0$

For transient analysis, the current waveform inside the integration is usually approximated by Backward Euler as:

$$\int_{t}^{t+\triangle t} i(\tau)d\tau = i(t+\triangle t)\triangle t \tag{II.4}$$

or using Trapezoidal Rule (TR) as:

$$\int_{t}^{t+\triangle t} i(\tau)d\tau = \frac{i(t+\triangle t)+i(t)}{2}\triangle t \tag{II.5}$$

The corresponding equivalent circuit models are shown in Figure II.3 and Figure II.4 respectively.

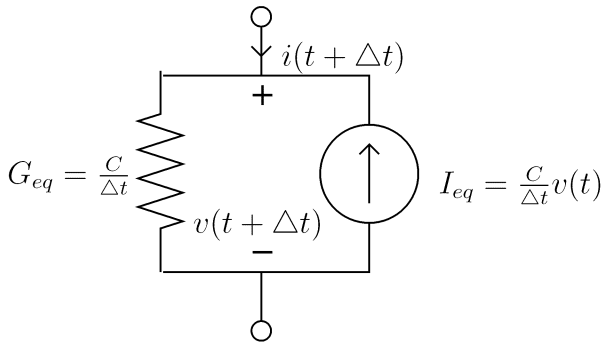


Figure II.3: Equivalent Circuit Model of Capacitor using Backward Euler Approximation

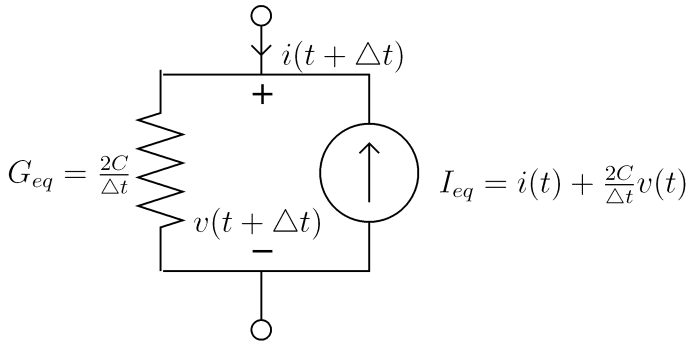


Figure II.4: Equivalent Circuit Model of Capacitor using Trapezoidal Rule Approximation

II.B.2 Numerical Integration of Inductor

In order to derive a linear relation of current and voltage across an inductor shown in Figure II.5 from the differential constitutive equation (II.6), the voltage waveform is integrated over a time interval $\triangle t$ in Equation (II.7).

$$v = L\frac{di}{dt} \tag{II.6}$$

$$i(t+\triangle t) = i(t) + \frac{1}{L}\int_{t}^{t+\triangle t} v(\tau)d\tau \tag{II.7}$$

Similar to the capacitor model, the voltage waveform inside the integration is usually approximated by Backward Euler (BE) as:

$$\int_{t}^{t+\triangle t} v(\tau)d\tau = v(t+\triangle t)\triangle t \tag{II.8}$$

or using Trapezoidal Rule (TR) as:

$$\int_{t}^{t+\triangle t} v(\tau)d\tau = \frac{v(t+\triangle t)+v(t)}{2}\triangle t \tag{II.9}$$

The corresponding equivalent circuit models are shown in Figure II.6 and Figure

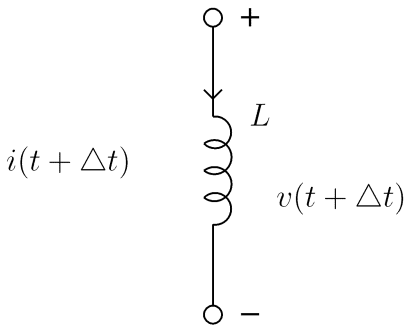


Figure II.5: Circuit Model of Inductor

II.7 respectively.

Since inductor has unknown branch current variable, Modified Nodal Analysis instead of Nodal Analysis is applied.

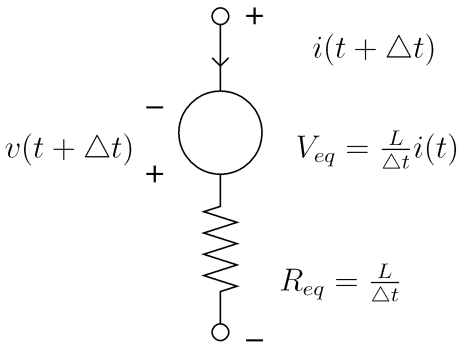


Figure II.6: Equivalent Circuit Model of Inductor using Backward Euler Approximation

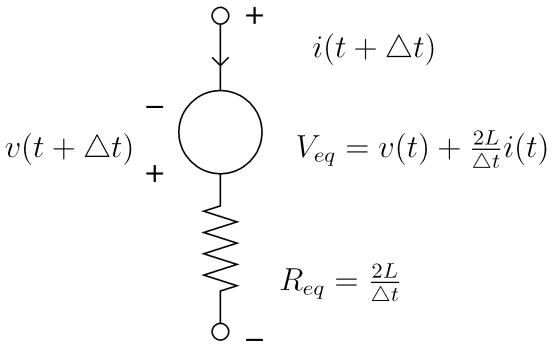


Figure II.7: Equivalent Circuit Model of Inductor using Trapezoidal Rule Approximation

II.B.3 Matrix Formulation

In terms of matrices, the system matrix of RLC circuits after numerical integration is formulated as:

$$\begin{bmatrix} \frac{C}{h}+G & -A^T \\ A & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{C}{h} & 0 \\ 0 & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t+h) \qquad \text{(II.10)}$$

for Backward Euler approximation and

$$\begin{bmatrix} \frac{2C}{h}+G & -A^T \\ A & \frac{2L}{h} \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{2C}{h} & A^T \\ -A & \frac{2L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t+h) + U(t) \qquad \text{(II.11)}$$

for Trapezoidal Rule approximation where C, L, G are the matrices of capacitance, inductance and conductance respectively. Matrix A is an incidence matrix linking between the topology of capacitance nodes and inductance branches. Vectors V, I, and U describe the vector of nodal voltages, branch currents and system inputs respectively. Scalar h is the time step from time t to $t+h$.

II.B.4 Convergence of Numerical Integration Method

The solution derived from numerical integration methods for circuit transient analysis is correct only if the maximum error over the entire time interval is smaller than the error tolerance. Stability and consistency are two key issues to ensure the global accuracy.

Stability

Stability property ensures that the error generated at current time point will not enlarge later in the following time points. An integration method is A-stable if it is stable for any time step size. The A-stable property is especially essential in the circuit simulation because circuits may be a stiff system with quite different time step size requirements. It can be proved that all high order (> 2) numerical integration methods are not A-stable. That also explain why circuit

simulators mainly adopt Backward Euler (First Order) and Trapezoidal Rule approximation (Second Order) for numerical integrations.

Consistency

A numerical integration method is A-stable does not mean it will generate correct result; it only ensures that the error generated at each time point will not increase in the following time points. For correctness, the local truncation error (LTE), error generated at each time point, should not exceed the error tolerance. The local truncation error is a function of time step size and can be reduced by choosing a smaller time step size. A numerical integration method should have reasonable time step size in order to be practical in circuit simulation. Tiny time step produces accurate result at each time point, but the waveform has to be solved at significant amount of time points, which leads to prohibitive runtime.

II.C Nonlinear System

The system equation for circuits with nonlinear devices is actually a multi-dimensional nonlinear equation $F(x) = 0$. Usually, there is no analytical solution for those nonlinear equations; instead, they are solved numerically.

Starting from an initial guess x^0, the nonlinear system solution is approximated iteratively by a sequence of solutions x^1,x^2,$\cdots$, x^n until it converges. The sequence of approximated solution is generated based on the Taylor expansion at the previous approximated solution point, as shown in Equation (II.12).

$$F(x^{(k+1)}) = F(x^{(k)}) + \frac{\partial F(x^{(k)})}{\partial x}(x^{(k+1)} - x^{(k)}) \tag{II.12}$$

For a multi-dimensional problem, the $\frac{\partial F}{\partial x}$ becomes a matrix, which is referred as Jacobian Matrix J. The definition of Jacobian matrix is derived as

below:

$$J(x) = \begin{bmatrix} \frac{\partial F_1(x)}{\partial x_1} & \cdots & \frac{\partial F_1(x)}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_N(x)}{\partial x_1} & \cdots & \frac{\partial F_N(x)}{\partial x_N} \end{bmatrix} \tag{II.13}$$

The approximated solution $x^{(k+1)}$ is solved by setting $F(x^{(k+1)}) \approx 0$, hence we derive a system of linear equations:

$$J(x^{(k)})x^{(k+1)} = -F(x^{(k)}) + J(x^{(k)})x^{(k)} \tag{II.14}$$

It can be proved that Newton's method converges quadratically as long as a sufficiently close initial guess is given.

II.D Transient Simulation Flow

The overall SPICE transient analysis flow is depicted in Figure II.8. The matrix is set up using Modified Nodal Analysis once and reloaded at the beginning of every Newton-Raphson iteration. First, all energy storage devices are replaced by equivalent circuit models using numerical integration. Then, all nonlinear elements are linearized at current solution point and the circuit equation becomes a system of linear equations, which is solved by LU factorization. At the end of each Newton-Raphson iteration, convergence is checked for every device. Once the Newton-Raphson iteration converges for all nonlinear devices, SPICE estimates the next time step size and moves to the next time point.

II.E Solving Systems of Linear Equations

Solving the system of linear equations is the most time consuming operation in the circuit analysis. With increasing design complexity, direct methods, such as LU decomposition used in Berkeley SPICE and its variations, are prohibitive due to the super linear complexity. In this section, we will briefly introduce the direct method and some efficient sparse matrix solving techniques, along with their advantages and limitations.

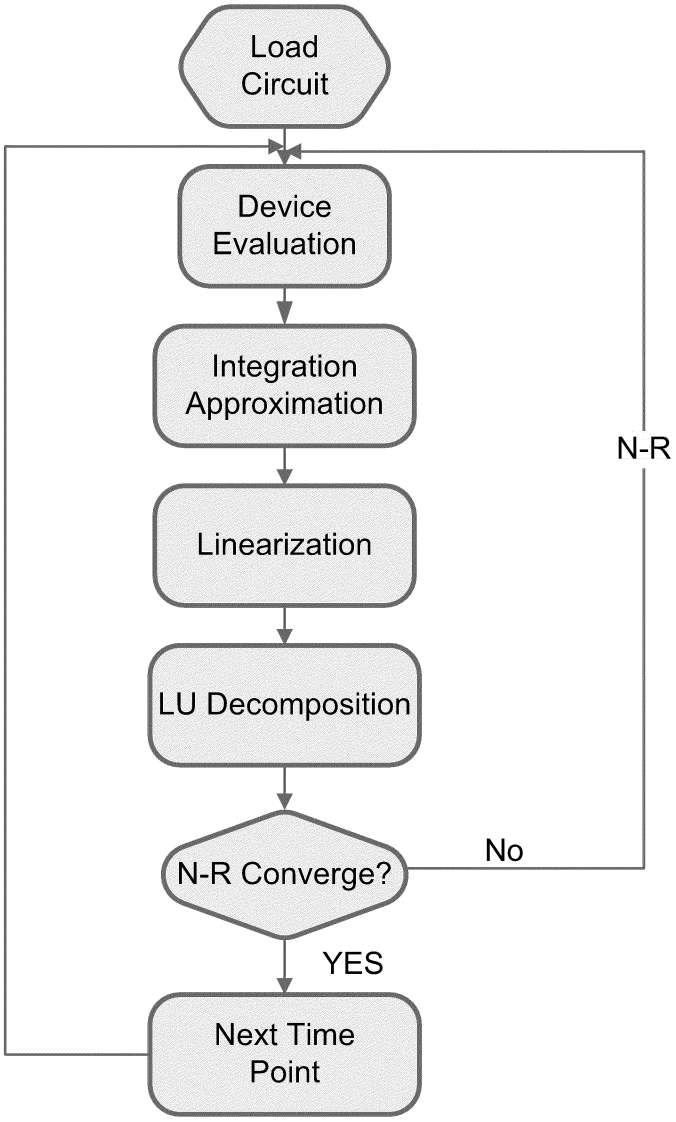


Figure II.8: SPICE Transient Simulation Flow

II.E.1 Direct Method (LU Decomposition)

For a linear system $Mx = b$, LU decomposition method factorizes the matrix M into a lower triangular matrix L and an upper triangular matrix U such that $A = LU$, and solves the matrix by forward and backward substitutions. The complexity is $O(n^3)$ for dense matrix and $O(n^{1.5})$ for sparse matrices generated in circuit simulation using Nodal Analysis or Modified Nodal Analysis.

In order to reduce the number of non-zero fill-ins generated during factorization, matrices have to be reordered according to their structure. However, the optimized ordering is difficult to achieve. In practice, various heuristic algorithms are used. Among them, minimum degree based ordering algorithm works well for circuit simulation applications.

Numerical stability issues such as division by tiny or zero diagonal terms should be considered during factorization. Pivoting is necessary to ensure the numerical stability.

Direct method cannot handle large matrices due to the high complexity, which is the main cause that limits the SPICE capacity because direct method is used to solve matrix equation in SPICE.

II.E.2 Basic Iterative Solver

Fortunately, the matrices generated in circuit simulation using Nodal Analysis (NA) or Modified Nodal Analysis (MNA) are sparse, which may be solved efficiently by iterative methods.

In general, there are two kinds of iterative methods, stationary and non-stationary. Stationary methods include some basic iterative methods such as Jacob, Gauss-Seidel and SOR. The update operators in stationary methods do not depend on the iteration count. The multigrid method can also be viewed as a stationary method. On the contrary, update operators in non-stationary methods differ at every iteration. Krylov methods, such as GMRES and conjugate gradient method, as well as their variants, are all non-stationary methods.

Given a system of linear equations,

$$\sum_{j=1}^{n} m_{ij} x_j = b_i \tag{II.15}$$

where $i = 1 \cdots n$. Its matrix formulation is

$$Mx = b \tag{II.16}$$

where M is a n by n non-singular square matrix, x and b are the unknown vector and the right-hand-side vector.

To simplify the notation of the following basic iterative methods, let $M = D - L - U$, where $D = \begin{bmatrix} m_{11} & & & \\ & m_{22} & & \\ & & \ddots & \\ & & & m_{nn} \end{bmatrix}$, $L = -\begin{bmatrix} 0 & & & & \\ m_{21} & 0 & & & \\ m_{31} & m_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1} & m_{n2} & m_{n3} & \cdots & 0 \end{bmatrix}$ and $U = -\begin{bmatrix} 0 & m_{12} & m_{13} & \cdots & m_{1n} \\ & 0 & m_{23} & \cdots & m_{2n} \\ & & 0 & \cdots & m_{3n} \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}$

D is the diagonal terms of matrix M. L is the lower triangular matrix and U is the upper triangular matrix of M. The meanings of matrices L and U are different from those in LU decomposition method.

Jacobi Method

The Jacobi method updates entries of the unknown vector x one by one at every iteration. As illustrated in Equation (II.18), the i^{th} entry of x is updated according to the i^{th} equation (II.17) with other entries of x fixed at the value of previous iteration. Here, the superscript denotes the iteration count.

$$\sum_{j=1}^{n} m_{ij} x_j = b_i \tag{II.17}$$

$$x_i^{(k)} = \frac{b_i - \sum_{j \neq i} m_{ij} x_j^{(k-1)}}{m_{ii}} \tag{II.18}$$

The Jacobi method in matrix term is derived as:

$$x^{(k)} = D^{-1}\left(L + U\right) x^{(k-1)} + D^{-1} b \tag{II.19}$$

Gauss-Seidel Method

The Gauss-Seidel method is similar to the Jacobi method, except that the newly calculated unknown entries are used to update other unknowns as soon as they are available. Thus, the updating of i^{th} entry of x at iteration k becomes:

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} m_{ij} x_j^{(k)} - \sum_{j>i} m_{ij} x_j^{(k-1)}}{m_{ii}} \tag{II.20}$$

The Gauss-Seidel method in matrix term is derived as:

$$x^{(k)} = \left(D - L\right)^{-1}\left(U x^{(k-1)} + b\right) \tag{II.21}$$

Better convergence is expected for Gauss-Seidel method, because the newly calculated unknown entries are used at the current iteration as soon as they are available instead of waiting until the next iteration.

Successive Over Relaxation Method (SOR)

The SOR method is an enhanced version of Gauss-Seidel method. It is actually a weighted average of Gauss-Seidel solution $\overline{x}$ at current iteration and the solution of previous iteration:

$$x_i^{(k)} = \omega \overline{x}_i^{(k)} + (1 - \omega) x_i^{(k-1)} \tag{II.22}$$

where ω is a scalar chosen to accelerate the convergence and $0 < \omega < 2$. The selection of ω relates to the matrix eigenvalues and is not an easy problem.

The SOR method in matrix term is given as:

$$x^{(k)} = (D - wL)^{-1} (wU + (1 - w)D)x^{(k-1)} + w(D - wL)^{-1}b \tag{II.23}$$

Gauss-Seidel method can be viewed as a special case of SOR method with $\omega = 1$. The SOR method is called Over-Relaxation when $2 > \omega > 1$ and Under-Relaxation when $\omega < 1$.

Convergence Study of Basic Iterative Methods

Iterative methods converge for matrices that satisfy convergence conditions. For a system of linear equations $Mx = b$, a sufficient condition for the convergence of Gauss-Seidel and Jacobi methods is that the matrix A is diagonally dominant:

$$|m_{ii}| \geq \sum_{j=1\&j\neq i}^{n} |m_{ij}\ | \tag{II.24}$$

When matrix M is symmetric positive definite, which means all eigenvalues are positive, the SOR method converges for any ω in the range of $(0, 2)$.

In general, all stationary iterative methods have the following formulation:

$$x^{(k)} = Gx^{(k-1)} + f \tag{II.25}$$

where G is the iterative matrix and the superscription represents the iteration count.

A necessary and sufficient condition for the convergence of stationary iterative methods is that the magnitude of the largest eigenvalue of the iterative matrix G is smaller than 1.

We can derive the iterative matrices for Jacobi, Gauss-Seidel and SOR methods respectively:

- Jacobi Method

$$G = D^{-1} (L + U) \tag{II.26}$$

- Gauss-Seidel Method

$$G = (D - L)^{-1}U \tag{II.27}$$

- SOR Method

$$G = (D - \omega L)^{-1}(\omega U + (1 - \omega)D) \tag{II.28}$$

II.E.3 Conjugate Gradient Method

When the matrix in a system of linear equations $Mx = b$ is symmetric and positive definite, the solution of $Mx = b$ is actually equivalent to the minimization problem of the quadratic function $f(x)$ of the unknown vector x:

$$f(x) = \tfrac{1}{2}x^T Mx - b^T x + c \tag{II.29}$$

The basic idea of conjugate gradient method is to find a set of minimization directions and step sizes along those directions to minimize the quadratic function $f(x)$ in a n dimensional space where n is the number of unknowns. At each iteration, the solution is updated as Equation (II.30) until it is close enough to the exact solution.

$$x^{(i+1)} = x^{(i)} + a^{(i)}d^{(i)} \tag{II.30}$$

where $d^{(i)}$, $a^{(i)}$ are update direction and the associated step size at the iteration i respectively.

The algorithm of conjugate gradient method is given below:

$$\begin{aligned} d^{(0)} &= r^{(0)} = b - Mx^{(0)} \\ a^{(i)} &= \frac{{r^{(i)}}^T r^{(i)}}{{d^{(i)}}^T M d^{(i)}} \\ x^{(i+1)} &= x^{(i)} + a^{(i)}d^{(i)} \\ r^{(i+1)} &= r^{(i+1)} - a^{(i)}Md^{(i)} \\ \beta^{(i+1)} &= \frac{{r^{(i+1)}}^T r^{(i+1)}}{{r^{(i)}}^T r^{(i)}} \\ d^{(i+1)} &= r^{(i+1)} + \beta^{(i+1)}d^{(i)} \end{aligned} \tag{II.31}$$

At each iteration, the conjugate gradient method eliminates error components in one orthogonal direction. It can be proved that the conjugate gradient

method gets the exact solution in n iterations. However, if the matrix dimension n is large, the conjugate gradient method will involve too many iterations. The update directions relate to the eigenvectors. If the eigenvectors are clustered, then few iterations may get sufficient accurate result. Since not all matrices have this property, a preconditioner N is used to improve the matrix property.

The idea of preconditioning is to solve the Equation (II.32) instead of the original problem $Mx = b$.

$$N^{-1}Mx = N^{-1}b \tag{II.32}$$

The preconditioning matrix N is selected to decrease the matrix condition number such that the eigenvectors of $N^{-1}M$ are more clustered.

The algorithm of preconditioned conjugate gradient method is given below:

$$\begin{aligned}
r^{(0)} &= b - Mx^{(0)} \\
d^{(0)} &= N^{-1}r^{(0)} \\
a^{(i)} &= \frac{{r^{(i)}}^T N^{-1} r^{(i)}}{{d^{(i)}}^T M d^{(i)}} \\
x^{(i+1)} &= x^{(i)} + a^{(i)} d^{(i)} \\
r^{(i+1)} &= r^{(i+1)} - a^{(i)} M d^{(i)} \\
\beta^{(i+1)} &= \frac{{r^{(i+1)}}^T N^{-1} r^{(i+1)}}{{r^{(i)}}^T N^{-1} r^{(i)}} \\
d^{(i+1)} &= N^{-1}r^{(i+1)} + \beta^{(i+1)} d^{(i)}
\end{aligned} \tag{II.33}$$

Matrix N is the preconditioner. At each iteration, the vector $N^{-1}r^{(i+1)}$ is derived from the solution of $Nz^{(i+1)} = r^{(i+1)}$, which is solved only once.

Conjugate gradient method is always used with some preconditioner and the choice of preconditioner is essential to the performance of the conjugate gradient method. There is no universal preconditioner for all matrices. Every preconditioner has its advantages and disadvantages. It only works well for certain kinds of matrices.

Conjugate gradient method converges for symmetric and positive definite matrices. There are other variants of CG-like methods. Among them, GMRES [46] can handle unsymmetrical matrices at a cost of extra memory consumption because

it has to store all intermediate vectors for direction constructions. Other variants of CG-like methods include Bi-CGSTAB [43], which can also handle unsymmetrical matrices but is less robust.

II.E.4 Multigrid Method

The multigrid method is an efficient technique first widely used for solving partial differential equations [4]. The basic idea of multigrid method is to map the hard-to-damp low frequency error at fine level to the easy-to-damp high frequency error at coarse level, solve the mapped problem at coarse level, and then map the error correction of coarse level back to the fine level. The mapping operator from fine to coarse level is called the restriction operator P_h^{2h}, and the mapping operator from coarse to fine level is called the interpolation operator P_{2h}^h. Here the subscription and superscription represent the grid level. The construction of hierarchical grid structure stops when the coarse level matrix can be quickly solved by LU decomposition. At each level, the high frequency error is erased by a smoothing operator, usually some basic iterative methods such as Gauss-Seidel. The residue is passed to the coarse level and the solution at coarse level is actually the error correction to be passed to the fine level. An iteration of restrictions from the finest level to the coarsest level and interpolations from the coarsest level back to the finest level is called a single multigrid V-cycle, which iterates until converge. A single multigrid V-cycle [4] is illustrated below.

Multigrid V-Cycle

- Relax on $M^h u^h = f^h$ ν_1 times with initial guess v^h, calculate residue $r^h = f^h - M^h v^h$
- Compute $f^{2h} = P_h^{2h} r^h$
  - Relax on $M^{2h} u^{2h} = f^{2h}$ ν_1 times with initial guess $v^{2h} = 0$, calculate residue $r^{2h} = f^{2h} - M^{2h} v^{2h}$

- Compute $f^{4h} = P_{2h}^{4h} r^{2h}$
  - Relax on $M^{4h}u^{4h} = f^{4h}$ ν_1 times with initial guess $v^{4h} = 0$, calculate residue $r^{4h} = f^{4h} - M^{4h}v^{4h}$
  - Compute $f^{8h} = P_{4h}^{8h} r^{4h}$
    .
    . to coarsest level
    .
  - Correct $v^{4h} \leftarrow v^{4h} + P_{8h}^{4h} v^{8h}$
  - Relax on $M^{4h}u^{4h} = f^{4h}$ ν_2 times with initial guess v^{4h}
- Correct $v^{2h} \leftarrow v^{2h} + P_{4h}^{2h} v^{4h}$
- Relax on $M^{2h}u^{2h} = f^{2h}$ ν_2 times with initial guess v^{2h}

- Correct $v^h \leftarrow v^h + P_{2h}^h v^{2h}$
- Relax on $M^h u^h = f^h$ ν_2 times with initial guess v^h

Geometric Multigrid

There are two kinds of multigrid methods: geometric multigrid and algebraic multigrid (AMG). Geometric multigrid method requires regular mesh structure. Its coarse grid reduction algorithm is defined geometrically, e.g. coarse node is selected every other fine node. If we want to handle problems with irregular structure, AMG is a good alternative of geometric multigrid method. The coarsening and interpolation operation of AMG are based on the matrix itself. This overhead makes AMG less efficient than geometric multigrid if the problem analyzed has regular mesh structure. In geometric multigrid, the smoothing operation at each level is important to the convergence. The error after smoothing should be geometrically smooth, in other words, the local error among adjacent nodes should be small. While in AMG, the interpolation plays the crucial role. The convergence rate actually depends on the coarse grid construction and inter-level mapping operators [38].

Algebraic Multigrid

Since AMG has no grid concept in mind, the inter-level mapping operators have to be determined based on the matrix. In AMG, smooth error means error components with relatively small residuals (II.34) [4], which means the residue is small but the error decreases very slowly after several iterations of smoothing operation.

$$Me \approx 0 \tag{II.34}$$

Equation (II.34) can be rewritten as (II.35). Hence, the error of fine node can be well represented by a linear combination of its neighbors' error.

$$m_{ii}e_i \approx -\sum_{j \neq i} m_{ij}e_j \tag{II.35}$$

If coarse and fine nodes have already been defined, we can further approximate the error of fine node by errors of only its coarse node neighbors. From Equation (II.35), we can construct the interpolation operator P_{2h}^{h}. Because of the symmetry of the mapping between levels in two directions, the restriction operator P_{h}^{2h} is the transposition of the interpolation operator, i.e. ${P_{2h}^{h}}^{T} = P_{h}^{2h}$. The coarse level matrix $M^{(2h)}$ can be derived as $M^{2h} = P_{2h}^{h} M^{h} P_{h}^{2h}$, where $M^{(h)}$ is the fine level matrix.

If the matrix is symmetric and positive definite, convergence of AMG can be guaranteed as long as the smoothing operation converges at each level. A detailed proof can be found in [39].

Chapter III

Two-Stage Newton-Raphson Method for Transistor Level Simulation

III.A Introduction

In this chapter, we introduce an efficient transistor level simulation approach with SPICE-accuracy for deep-submicron (DSM) and nanometer VLSI circuits with strong coupling effects.

Without any simplified assumptions on device model, linearization process and numerical integration, the proposed approach targets fast-yet-accurate solutions for the entire system at every single time point. On the contrary, circuit partitioning is now widely used for the time-domain transistor-level simulation. Most commercial fast spice tools are based on partitioning. Though resolving the memory and complexity issues, partition based methods have difficulties to guarantee the accuracy especially when coupling effect exists.

Different circuit components have different physical natures and the resulting system matrices have various properties. While fast linear solvers are good candidates for linear circuits such as power distribution networks, direct methods

are still needed for nonlinear circuits. Even though some fast linear solvers, such as GMRES [46] and Bi-CGSTAB [43] that can handle non-S.P.D. (symmetric positive and definite) matrices, have been used in the simulation of RF and analog components, the extra overhead and convergence issue prevent them from being applied to large-scale transistor level analysis.

Since there is no universal efficient matrix solver for general large-scale circuits, the proposed approach uses fast linear solvers for linear portions of circuits and direct method for nonlinear components. The separation is realized inside the innermost Newton-Raphson iteration of the transient simulation flow. The linear and nonlinear interfaces are dynamically abstracted as equivalent circuit models and there are iterations between linear and nonlinear portions of circuits to ensure the convergence. The same convergence check and dynamic time step control methodologies as SPICE are adopted to guarantee the SPICE-level accuracy.

Algebraic multigrid method is used to solve the large linear networks especially power and ground distribution networks. We extend the standard algebraic multigrid method (AMG) to handle general linear RLC circuits with self and mutual inductors. Since multigrid methods require matrices to be symmetric and positive definite, circuit system matrix formulated by Modified Nodal Analysis (MNA) is converted to the symmetric and positive definite form. Special grid construction and error smoothing are proposed for circuits with self and mutual inductors.

In order to exploit circuit spatial and temporal latencies, incremental multigrid hierarchical grid construction and adaptive error smoothing are introduced to avoid non-necessary computation overhead.

This chapter is organized as follows. The two-stage Newton-Raphson linear-nonlinear interface modelling and overall transient analysis flow are presented in section III.B. Section III.C discusses the algebraic multigrid theory, application on general RLC circuits and adaptive methodologies. Experimental results are then demonstrated in section III.D.

III.B Two-Stage Discrete Newton-Raphson Method

The proposed approach decouples nonlinear components and linear networks inside the Newton-Raphson iteration and dynamically models the linear-nonlinear interface using a two-stage Newton-Raphson procedure. Figure III.1(a) shows an example of an NAND gate surrounded by coupled power, ground and signal nets. Decoupling and interface modelling of nonlinear components and linear networks are shown in Figure III.1(b) and III.1(c) respectively. Though not drawn in those figures, the transistor bulk nodes are modelled in the same way.

The linear and nonlinear portions are then solved separately and iteratively, provided that the linear-nonlinear interfaces are dynamically modelled as Norton equivalent circuits when solving each portion. The detailed derivation of interface models is explained in the next subsection.

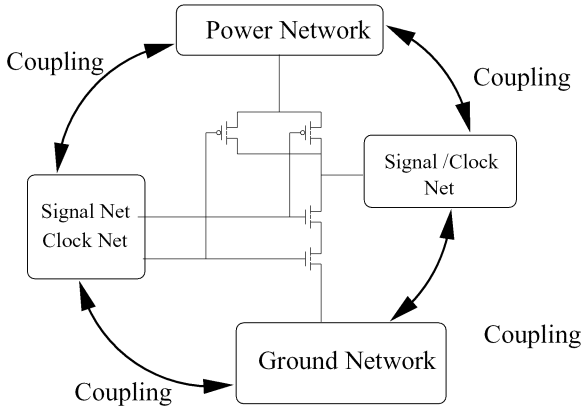
III.B.1 Norton Equivalent Linear-Nonlinear Interface Model

Accurate modelling of linear-nonlinear interfaces inside Newton-Raphson iterations is critical to the convergence of the proposed approach. If we consider the interface boundary condition as an $I - V$ function $i = f(v)$, it is intuitive to adopt the Newton-Raphson concept when modelling the interface. Inspired by this idea, we implement a discrete Newton-Raphson method, which includes two stages: nonlinear stage for nonlinear transistor components and linear stage for linear networks. The interfaces are linearized as Norton equivalent circuits, hence, the linear networks and nonlinear components can be solved separately inside Newton-Raphson iterations as shown in Figure III.2.

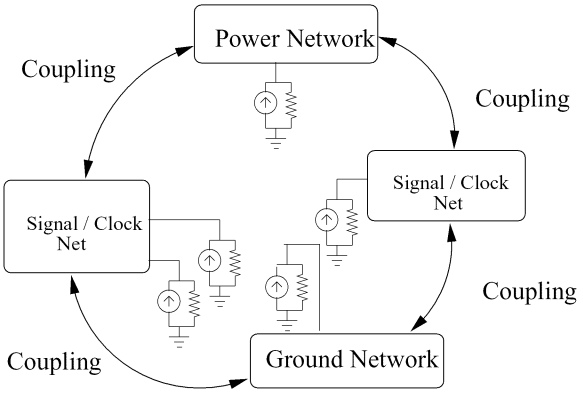
For the $I - V$ curves in Figure III.3 (a) and (b), the linearized equation using Newton-Raphson method is:

$$i = i^k + \frac{di}{dv}(v - v^k) \tag{III.1}$$

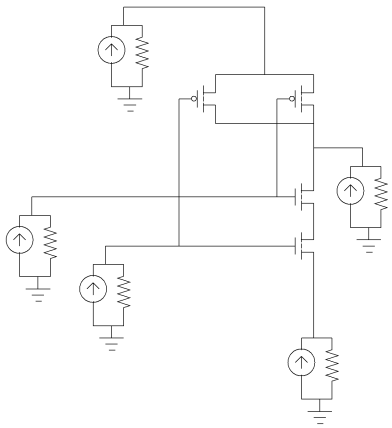
The dynamic conductance G_{eq} and equivalent current source I_{eq} in Figure III.3 (c) can be obtained from the following equations:



(a) Original Circuit

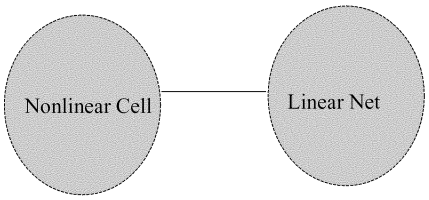


(b) Circuit at Linear Stage

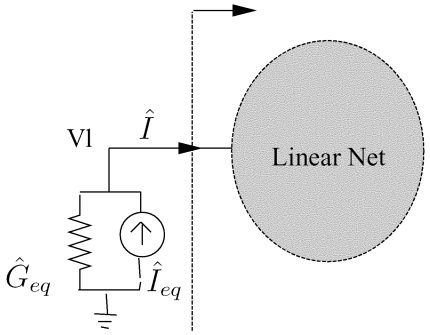


(c) Circuit at Nonlinear Stage

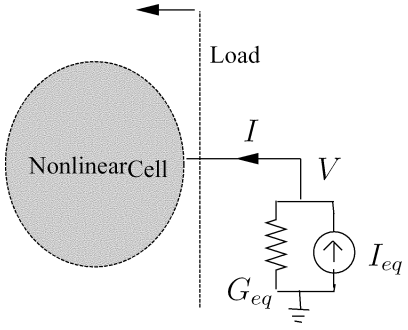
Figure III.1: Example of NAND Gate Surrounded by Power/Clock/ Signal Net



(a) Linear network and nonlinear component interface



(b) Equivalent Circuit Model of Nonlinear Component



(c) Equivalent Circuit Model of Linear network

Figure III.2: Example of Interface Modelling. G_{eq} and $\hat{G}_{eq}$ are equivalent conductance. I_{eq} and $\hat{I}_{eq}$ denote the equivalent current. V, $\hat{V}$, I and $\hat{I}$ are the nodal voltage and branch current respectively.

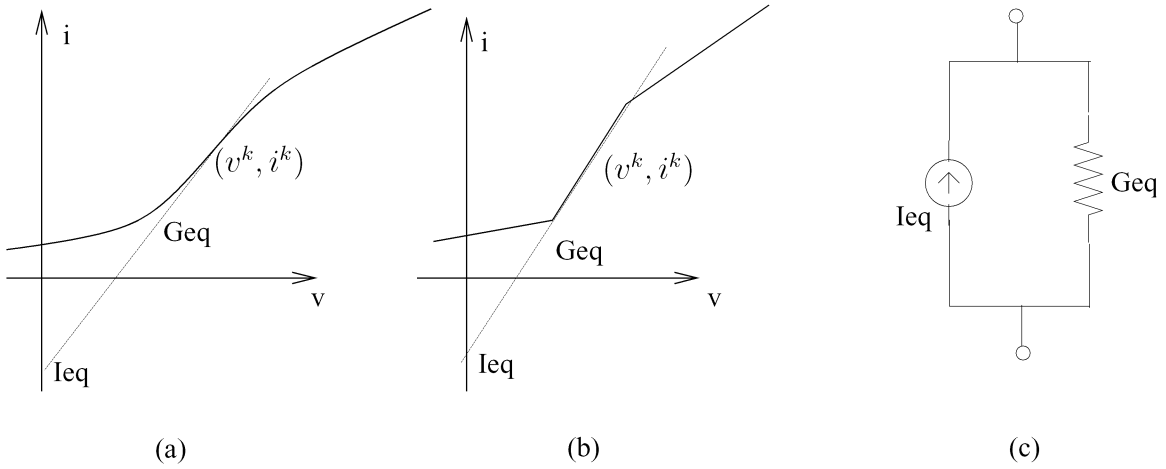


Figure III.3: Newton-Raphson Linearization (a) Nonlinear Curve (b) Piecewise Linear Curve (c) Norton Equivalent Circuit Model

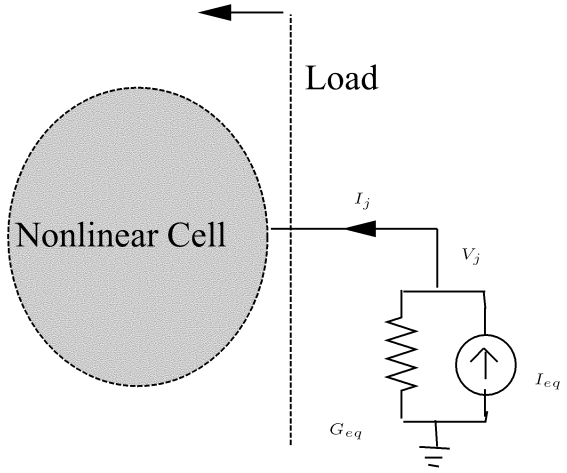
$$\begin{cases} G_{eq} &= \frac{di}{dv}|_k \\ I_{eq} &= G_{eq}v^k - i^k \end{cases} \tag{III.2}$$

Unlike the nonlinear devices which have given nonlinear $I-V$ functions and analytic G_{eq} formulation, the interface boundary $I-V$ function $f(v)$ is not determined until the circuit is simulated. Different discrete differential approximations are implemented for nonlinear components and linear networks according to their linearity properties and problem sizes.

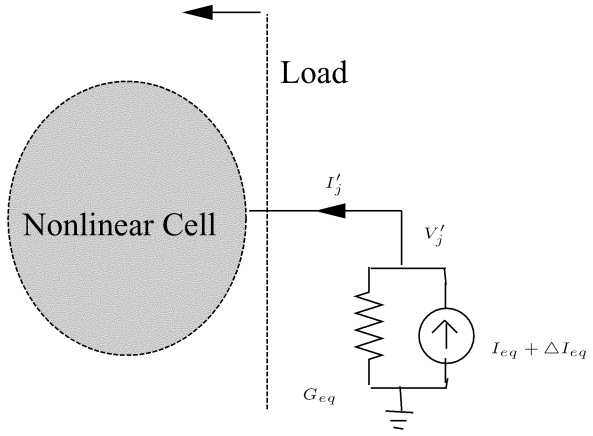
Norton Equivalent Circuit of Nonlinear Component Interface

Discrete Approximation Figure III.4 illustrates the derivation of the Norton equivalent circuit of a nonlinear component at boundary j. The load current I_{eq} and conductance G_{eq} are the Norton equivalent model of the neighboring linear network. We perturb the load current I_{eq} at boundary j with a small variation $\triangle I_{eq}$. The equivalent conductance of the nonlinear component is approximated from the alterations of nodal voltage and branch current at boundary j:

$$\hat{G}_{eq} = \frac{I'_j - I_j}{V'_j - V_j} \tag{III.3}$$



(a) Original Boundary Condition



(b) Perturbed Boundary Condition

Figure III.4: Nonlinear Stage Newton-Raphson Procedure at Port J

where V_j,V'_j, I_j, I'_j are nodal voltages and branch currents at boundary j with different load currents as can be seen in Figure III.4.

Output Conductance The Norton conductance is actually equivalent to the output conductance observed at the boundary node. When applying a unit current source at the boundary and leaving all other boundaries grounded, the output conductance G_{eqj} at boundary node j is the reciprocal of the nodal voltage. The voltage is solved from the LU factorized Jacobian matrix J of

the nonlinear component as:

$$G_{eqj} = \frac{1}{e_j^T (J)^{-1} e_j} = \frac{1}{e_j^T U^{-1} L^{-1} e_j} \tag{III.4}$$

where $e_j[i] = 1$ iff $i = j$.

Since the Jacobian matrix J is already factorized for the analysis of the nonlinear component itself, the boundary modelling only involves backward and forward substitutions with different unit current vector on the right hand side. The overhead to derive the equivalent conductance has a complexity of $O(K)$ where K is the total number of nonzero elements of the factorized Jacobian matrix J. If a nonlinear component has m boundaries, the overall complexity becomes $O(mK)$. However, the overhead can be reduced by optimizing the LU solving procedures if the boundaries nodes are reordered to the bottom of matrix during factorization.

The solving procedure consists of backward substitution which solves U matrix and forward substitution which solves L matrix as below:

$$\begin{aligned} Lz &= e_j \\ Ux &= z \\ G_{eqj} &= \frac{1}{e_j^T x} \end{aligned} \tag{III.5}$$

For a right hand side e_j that is nonzero only at j^{th} entry, there is no need to calculate all elements of vector z and x, since we are only interested in the j^{th} entry of x. As depicted in Figure III.5 and III.6, the complexity of backward and forward substitutions becomes $O(K_j)$, where K_j is the number of nonzero elements of submatrices L_j and U_j. Hence, the complexity for m boundaries is $O(mK')$ where K' is the number of nonzero elements of submatrices L' and U' which includes all m boundary nodes. If all boundaries are reordered to the bottom, the dimension of matrices L' and U' is only m. One thing to keep in mind is that if there are too many boundary nodes, the forced boundary nodes reordering may conflict with the original ordering in LU

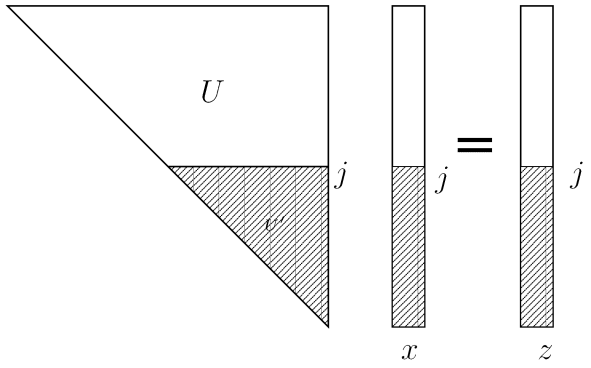


Figure III.5: Solving $Ux = z$. Shaded area in vector z means nonzero elements. Shaded area in vector x and matrix U means elements needed in derivation.

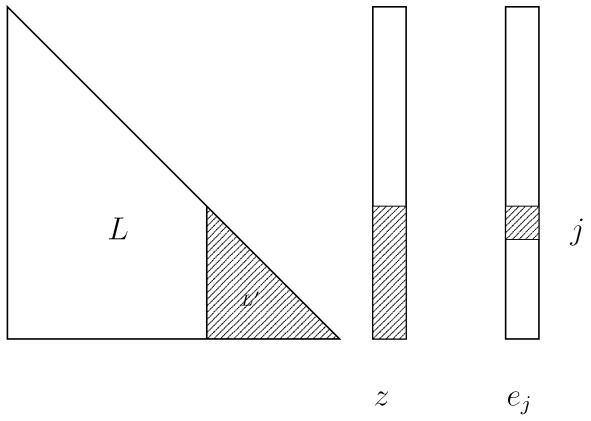


Figure III.6: Solving $Lz = e_j$. Shaded area in vector z and e means nonzero elements. Shaded area in matrix L means elements needed in derivation

decomposition and cause the increase of the number nonzero fill-ins during LU factorization.

Comparison Unlike the discrete approximation, the output conductance is the exact value of Norton equivalent conductance. However, the result of discrete approximation is quite close to the exact value, as can be concluded from Figure III.7, which shows the error distribution for 1500 boundary samples of an ASIC design.

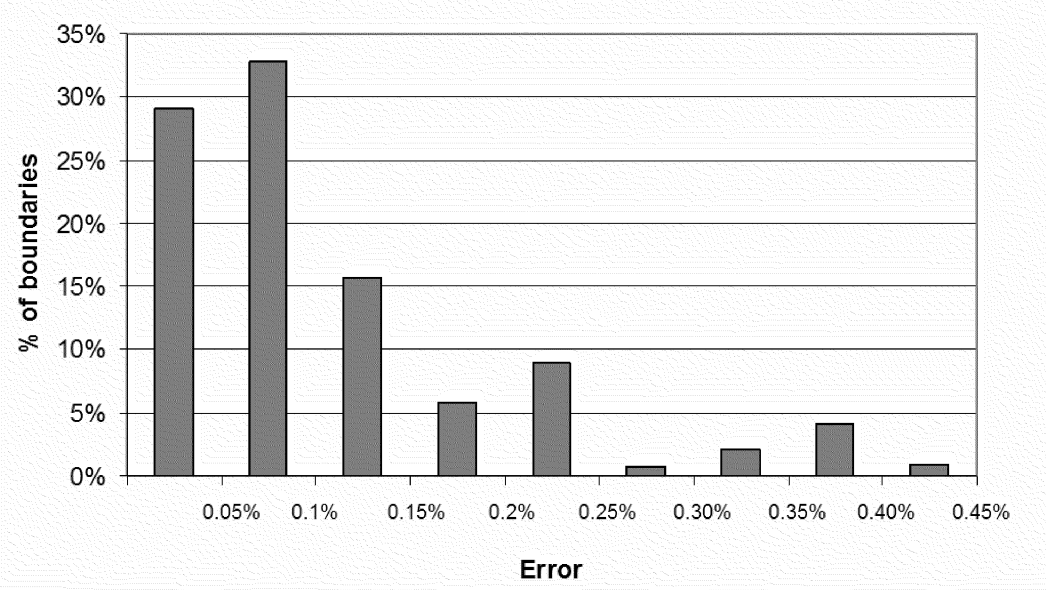


Figure III.7: Error Distribution of Discrete Approximation

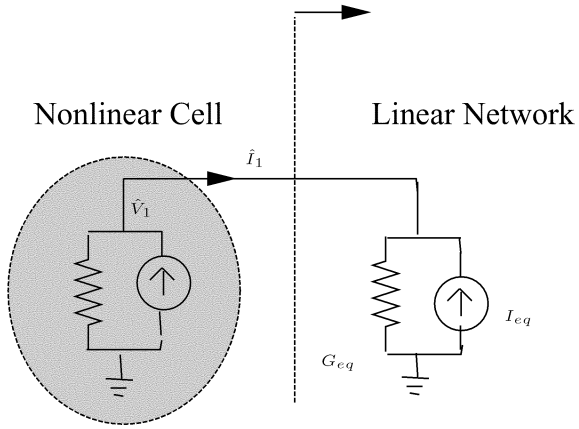
Norton Equivalent Circuit of Linear Network Interface

Norton equivalent models of linear networks at the interfaces can be constructed in the same way as nonlinear components. However, it is not efficient to solve linear networks repeatedly for every boundary because linear networks may contain millions of elements. Instead, we use a discrete approximation method in which the Norton equivalent conductance G_{eq} and current I_{eq} of linear network at the boundary are approximated from boundary conditions (nodal voltages and branch currents of the corresponding nonlinear component) of previous Newton-Raphson iterations.

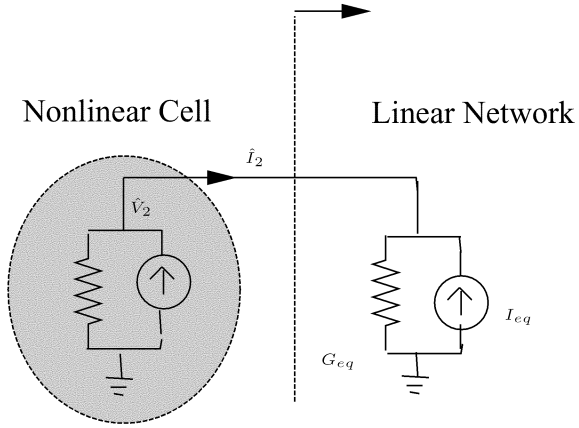
As can be revealed from Figure III.8(a) and III.8(b), the nonlinear component boundary nodal voltages $\hat{V}_1$, $\hat{V}_2$ and branch currents $\hat{I}_1$, $\hat{I}_2$ of two previous Newton-Raphson iterations have the following relation:

$$\begin{cases} \hat{V}_1 &= (I_{eq} + \hat{I}_1)/G_{eq} \\ \hat{V}_2 &= (I_{eq} + \hat{I}_2)/G_{eq} \end{cases} \tag{III.6}$$

The Norton equivalent circuit model G_{eq} and I_{eq} are then obtained by solving the



(a) First Newton-Raphson Iteration



(b) Second Newton-Raphson Iteration

Figure III.8: Linear Stage Newton-Raphson Procedure at Port J

above equation as below:

$$\begin{cases} G_{eq} &= \frac{\hat{I}_2 - \hat{I}_1}{\hat{V}_2 - \hat{V}_1} \\ I_{eq} &= G_{eq}\hat{V}_2 - \hat{I}_2 \end{cases} \tag{III.7}$$

III.B.2 Algorithm Description

Table III.1 describes the detailed procedure of the two-stage Newton-Raphson method at a single linear-nonlinear boundary. The first column is the index of Newton-Raphson iteration. The second column shows the Norton equivalent circuit of linear network and the third column gives the Norton equivalent

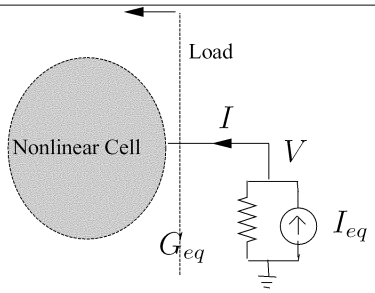
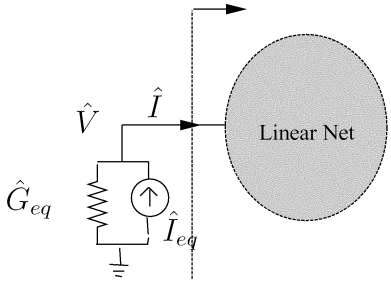
circuit of nonlinear components. $\hat{V}$ and $\hat{I}$ denote the boundary nodal voltage and branch current when nonlinear component is modelled as Norton equivalent circuit. V, I are the boundary nodal voltage and branch current when linear network is modelled as Norton equivalent circuit. $\hat{G}_{eq}$ and $\hat{I}_{eq}$ denote the Norton equivalent conductance and current source of nonlinear component at the boundary, while G_{eq} and I_{eq} are the equivalent conductance and current source of linear network. The subscript denotes the Newton-Raphson iteration count. Each table entry lists the boundary conditions given before the current stage and solution after that. Because there is no previous Newton-Raphson iterations to apply the discrete Newton-Raphson approximation in the first two iterations, the equivalent conductance of linear network is set to a small value, which implies the boundary is approximated as open circuit. Though not very accurate, it provides good initial guess for the following iterations especially when finding the DC operating point before transient analysis.

In addition, similar to the dynamic conductance limiting strategy implemented in Berkeley SPICE3, the Norton equivalent conductance has an upper bound ($1e^{6}$) and lower bound ($1e^{-12}$) for the sake of numerical stability.

The overall transient analysis flow (Figure III.9) is similar to Berkeley SPICE3 except for the steps shown in dashed boxes. The linear/nonlinear iteration replaces the LU decomposition solver inside the Newton-Raphson algorithm.

After the integration approximation and Newton-Raphson linearization, the linear networks and nonlinear components are solved separately. First, the linear networks are modelled as equivalent circuits at the linear-nonlinear boundaries and nonlinear components are solved by LU decomposition method with those boundary conditions. Next, all nonlinear components are replaced with Norton equivalent circuits and the linear networks are solved using adaptive multigrid method discussed in the next section. The linear-nonlinear iteration continues until the difference between boundary nodal voltages calculated from linear and nonlinear stages is smaller than the error tolerance at every boundary.

Table III.1: Equivalent Interface Model

NR	Linear	Nonlinear
		
1	$Given$ $G_{eq1} = 1e^{-6}$ $I_{eq1} = \hat{V}_{init} G_{eq1}$ $Output$ V_1 $I_1 = I_{eq1} - V_1 G_{eq1}$	$Given$ $\hat{G}_{eq1} = \frac{\delta I_1}{\delta V_1}$ $\hat{I}_{eq1} = V_1 \hat{G}_{eq1} - I_1$ $Output$ $\hat{V}_1$ $\hat{I}_1 = \hat{I}_{eq1} - \hat{V}_1 \hat{G}_{eq1}$
2	$Given$ $G_{eq2} = 1e^{-6}$ $I_{eq2} = \hat{V}_1 G_{eq2} - \hat{I}_1$ $Output$ V_2 $I_2 = I_{eq2} - V_2 G_{eq2}$	$Given$ $\hat{G}_{eq2} = \frac{\delta I_2}{\delta V_2}$ $\hat{I}_{eq2} = V_2 \hat{G}_{eq2} - I_2$ $Output$ $\hat{V}_2$ $\hat{I}_2 = \hat{I}_{eq2} - \hat{V}_2 \hat{G}_{eq2}$
3	$Given$ $G_{eq3} = \frac{\hat{I}_2 - \hat{I}_1}{\hat{V}_2 - \hat{V}_1}$ $I_{eq3} = \hat{V}_2 G_{eq3} - \hat{I}_2$ $Output$ V_3 $I_3 = I_{eq3} - V_3 G_{eq3}$	$Given$ $\hat{G}_{eq3} = \frac{\delta I_3}{\delta V_3}$ $\hat{I}_{eq3} = V_3 \hat{G}_{eq3} - I_3$ $Output$ $\hat{V}_3$ $\hat{I}_3 = \hat{I}_{eq3} - \hat{V}_3 \hat{G}_{eq3}$
...	...	...
k	$Given$ $G_{eqk} = \frac{\hat{I}_{k-1} - \hat{I}_{k-2}}{\hat{V}_{k-1} - \hat{V}_{k-2}}$ $I_{eqk} = \hat{V}_{k-1} G_{eqk} - \hat{I}_{k-1}$ $Output$ V_k $I_k = I_{eqk} - V_k G_k$	$Given$ $\hat{G}_{eqk} = \frac{\delta I_k}{\delta V_k}$ $\hat{I}_{eqk} = V_k \hat{G}_{eqk} - I_k$ $Output$ $\hat{V}_k$ $\hat{I}_k = \hat{I}_{eqk} - \hat{V}_k \hat{G}_{eqk}$

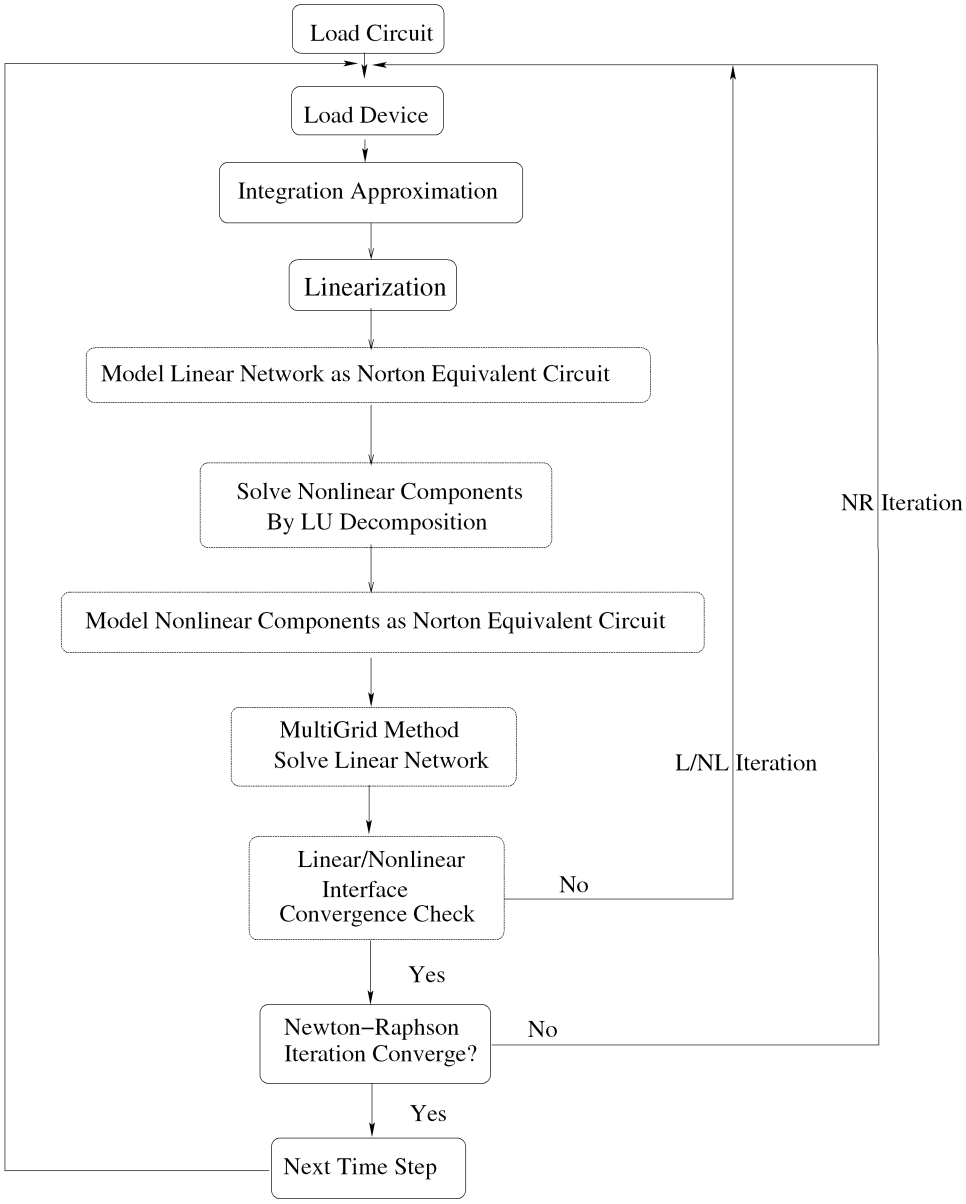


Figure III.9: Updated Transient Simulation Flow

III.B.3 Convergence Consideration

To derive the convergence property of the proposed approach, we define the internal nodes of linear networks as linear nodes, the internal nodes of nonlinear components as nonlinear nodes and the nodes connecting linear and nonlinear components as boundary nodes respectively. We write the linearized circuit equation according to the order of node groups:

$$\begin{bmatrix} G_l & G_b & 0 \\ B_g & B_l + B_n & B_m \\ 0 & M_b & M \end{bmatrix} \begin{bmatrix} V_g \\ V_b \\ V_m \end{bmatrix} = \begin{bmatrix} b_g \\ b_b \\ b_m \end{bmatrix} \tag{III.8}$$

where the subscripts g, b, and m refer to linear nodes, boundary nodes, and nonlinear nodes, respectively. Submatrices G_l and M are Jacobian matrices corresponding to the linear nodes and nonlinear nodes. Submatrix B_l denotes the Jacobian matrix contributed by linear circuits at the boundary nodes, while submatrix B_n denotes the Jacobian matrix contributed by nonlinear circuits at the interface nodes. Submatrices B_g, G_b, M_b and B_m represent the linearized relation between node groups. Vectors V_g, V_b, V_m are nodal voltages of linear, boundary and nonlinear nodes, respectively. Vectors b_g, b_b, b_m denote the corresponding currents.

The two-stage Newton-Raphson approach is described by the following iterative equations:

$$\begin{cases} \begin{bmatrix} G_l & G_b \\ B_g & B_l + D_1 \end{bmatrix} \begin{bmatrix} V_g^{(k+1)} \\ V_b^{(k+\frac{1}{2})} \end{bmatrix} = \begin{bmatrix} b_g \\ b_{b1} \end{bmatrix} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k = 0, 1, 2... \\ \begin{bmatrix} B_n + D_2 & B_m \\ M_b & M \end{bmatrix} \begin{bmatrix} V_b^{(k+1)} \\ V_m^{(k+1)} \end{bmatrix} = \begin{bmatrix} b_{b2} \\ b_m \end{bmatrix} \end{cases} \tag{III.9}$$

where $\begin{cases} D_1 = Diag(B_n - B_m M^{-1} M_b) \\ N_1 = B_n - B_m M^{-1} M_b - D_1 \\ D_2 = Diag(B_l - B_g G_l^{-1} G_b) \\ N_2 = B_l - B_g G_l^{-1} G_b - D_2 \\ b_{b1} = b_b - B_m M^{-1} b_m - N_1 V_b^k \\ b_{b2} = b_b - B_g G_l^{-1} b_g - N_2 V_b^{k+\frac{1}{2}} \end{cases}$

The convergence condition for the above iteration is given in Theorem IV.C.1.

Theorem III.B.1 *The two-stage Newton-Raphson approach converges if* $\|E_2^{-1} N_2 E_1^{-1} N_1\| < 1$ *where* $E_1 = B_n + D_2 - B_m M^{-1} M_b$ *and* $E_2 = B_l + D_1 - B_g G_l^{-1} G_b$

Most digital circuits without strong feedback loops will satisfy this convergence condition.

III.C Adaptive Algebraic Multigrid

The coupled linear network is solved by algebraic multigrid method, which is an efficient technique first widely used for solving partial differential equations [4, 41]. It has also been applied on the linear power network analysis [19, 30, 49]. Description of basic multigrid theory can be found in Chapter II and [4].

In general, there are two kinds of multigrid methods: geometric multigrid and algebraic multigrid. Geometric multigrid requires regular mesh structure but has less coarse grid reduction cost while algebraic multigrid (AMG) can handle problems in general topology but has more coarsening overhead. We choose algebraic multigrid over geometric multigrid based on the following considerations:

- Algebraic multigrid method operates directly on matrices. The coarse level matrix is assembled by matrix multiplication. Consequently, AMG has no special topological requirement as opposed to geometric multigrid.
- With the signal frequency increasing rapidly, the inductance effect can no longer be just ignored. However, system matrices formulated by Modified

Nodal Analysis (MNA) for circuits with inductors are not symmetric and positive definite (S.P.D.), thus cannot satisfy the convergence requirement of multigrid or conjugate gradient method. Even though system matrices can be reformulated to be symmetric and positive definite, the topology of reformulated system matrices is no longer the same as original circuits. Moreover, it is not easy to obtain coarse level grids directly from the circuit topology under these circumstances even if the circuit is a regular mesh.

Due to the non-uniformly distributed circuit switching activities and inhomogeneous circuit elements, circuits exhibit multi-rate behavior as well as latency. This phenomenon can be utilized to avoid unnecessary computation. In the algebraic multigrid solver, we apply adaptive methodologies specially designed for the circuit simulation applications to take the advantage of non-uniform circuit activities.

III.C.1 Hierarchical Grid Construction

The hierarchical grid structure is the infrastructure of multigrid method. The residue and error correction are passed up and down between grid levels such that all error components are reduced efficiently.

Multigrid method requires matrices to be symmetric and positive definite in order to converge. Although system matrices after numerical integrations for RC circuits are symmetric and positive definite, it is not true for circuits with inductors. Reformulation is needed to satisfy the convergence requirement of multigrid method. Accordingly, we also propose a special grid construction for circuits with inductors.

Reformulation of System Matrix for RLC circuits

Circuit equation for RC circuits is formulated as

$$C\dot{V}(t) + GV(t) = U(t) \tag{III.10}$$

where V is the vector of nodal voltages, C is the capacitance matrix, G is the conductance matrix and vector U is inputs. Apply trapezoidal approximation with time step h on Equation (III.10), we have

$$(G + \frac{2}{h}C)V(t+h) = -(G - \frac{2}{h}C)V(t) + U(t) + U(t+h) \tag{III.11}$$

The left hand side matrix in Equation (III.11) is symmetric and positive definite, which makes the iterative methods converge quickly.

When inductors are included, the circuit equation becomes:

$$\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{I} \end{bmatrix} + \begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} V \\ I \end{bmatrix} = \begin{bmatrix} U \\ 0 \end{bmatrix} \tag{III.12}$$

where I is the vector of branch currents, L is the inductance matrix and A is the incidence matrix.

Apply the trapezoidal approximation with time step h, the solution of Equation (III.12) is derived by

$$\begin{bmatrix} \frac{2C}{h} + G & -A^T \\ A & \frac{2L}{h} \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{2C}{h} - G & A^T \\ -A & \frac{2L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + \begin{bmatrix} U(t+h) + U(t) \\ 0 \end{bmatrix} \tag{III.13}$$

Although MNA method can handle elements without admittance description, the transient analysis system matrix in (III.13) is no longer symmetric and positive definite when inductors are included because of the introduction of extra branch current variables. Since multigrid as well as PCG method requires matrices to be symmetric positive definite, some extra processing is needed to reformulate the system. Similar to the method used in [5], we split the variable vector into nodal voltage vector and branch current vector. Through block matrix operations, we decompose Equation (III.13) into two separate formulas (III.14) for nodal voltages and branch currents, respectively.

$$\begin{cases} \left(\frac{2C}{h} + G + \frac{h}{2}A^T L^{-1} A\right) V(t+h) &= \left(\frac{2C}{h} - G - \frac{h}{2}A^T L^{-1} A\right) V(t) \\ & \quad + [U(t) + U(t+h)] + 2A^T I(t) \\ I(t+h) &= I(t) - \frac{h}{2}L^{-1} A[V(t+h) + V(t)] \end{cases} \tag{III.14}$$

Similar result (III.16) can be derived for the system equation (III.15) integrated by Backward Euler approximation:

$$\begin{bmatrix} \frac{C}{h}+G & -A^T \\ A & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{C}{h} & 0 \\ 0 & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + \begin{bmatrix} U(t+h) \\ 0 \end{bmatrix} \quad \text{(III.15)}$$

$$\begin{cases} \left(\frac{C}{h}+G+hA^TL^{-1}A\right)V(t+h) &= \frac{C}{h}V(t)+U(t+h)+A^TI(t) \\ I(t+h) &= I(t)-hL^{-1}AV(t+h) \end{cases} \quad \text{(III.16)}$$

Inverse matrices L^{-1} in Equations (III.14) and (III.16) correspond to the K matrix proposed in [9], where the matrix inversion overhead is reduced by sparsification methods. If the inductance matrix is symmetric and positive definite (S.P.D.), we can prove that system matrices in Equations (III.14) and (III.16) remain S.P.D.. One thing to keep in mind is that the topology of reformulated system matrices in (III.14) and (III.16) is no longer the same as original circuits. Geometric based coarse grid reduction algorithm cannot be applied on RLC circuits directly.

Review of Standard Algebraic Grid Reduction

The proposed adaptive algebraic grid reduction for RLC circuits is extended from the classical algebraic multigrid method introduced in [4]. A brief summary of classical algebraic grid reduction algorithms is discussed below, which is necessary to better illustrate the proposed extended grid construction algorithm for general RLC circuits as well as the adaptive methodologies. Detailed classical algebraic grid reduction algorithm can be found in [4].

Coarse Node Selection by Coloring Scheme At each hierarchical grid level, a set of coarse nodes is selected to the coarse level and the rest nodes become fine nodes. Coarse nodes represent those fine nodes at coarse level. The coarse node selection algorithm should ensure that every fine node has at least one coarse level neighbor node. The coarse level matrix size is determined by

the number of coarse nodes, therefore congested coarse nodes selection should be avoided in order to reduce the coarse level size as much as possible. In other words, coarse nodes should not be strong-connected neighbors at fine level.

A two-level coloring scheme for coarse node selection is explained below for the sake of clarity. In practice, the coloring scheme is executed at every level except the coarsest one. Initially, the coarse potential of each node is set to its degree. The node with the maximum coarse potential is selected to be a coarse node and all its unassigned neighboring nodes are set as fine nodes. For each newly set fine node, we increase the coarse potential of its neighbor by 1. The process is repeated until every node has been assigned as either fine or coarse node. In the end, each fine node has at least one neighboring coarse node. Figure III.10 gives a step-by-step example of the coloring sequence.

For circuit transient analysis, the system is solved at many discrete time points over the entire simulation period. Although the matrix values vary at different time point because of various time step size and transistor devices switching activities, the structure of those matrices remains unchanged. Consequently, the coarse nodes are only selected once for each level. In the following time points, only mapping relations and coarse level matrices are reconstructed.

Coarse Matrix Construction For matrix equation $Mx = b$ at each level, the error is smoothed such that the residue is close to zero, as shown in Equation (III.17). However, the error vector itself may not be close to zero at all. That is why basic iterative methods like Gauss-Seidel and SOR have slow convergence rate after first several iterations.

$$Me \approx 0 \tag{III.17}$$

Equation (III.17) can be rewritten as (III.18). Hence, the error of fine node

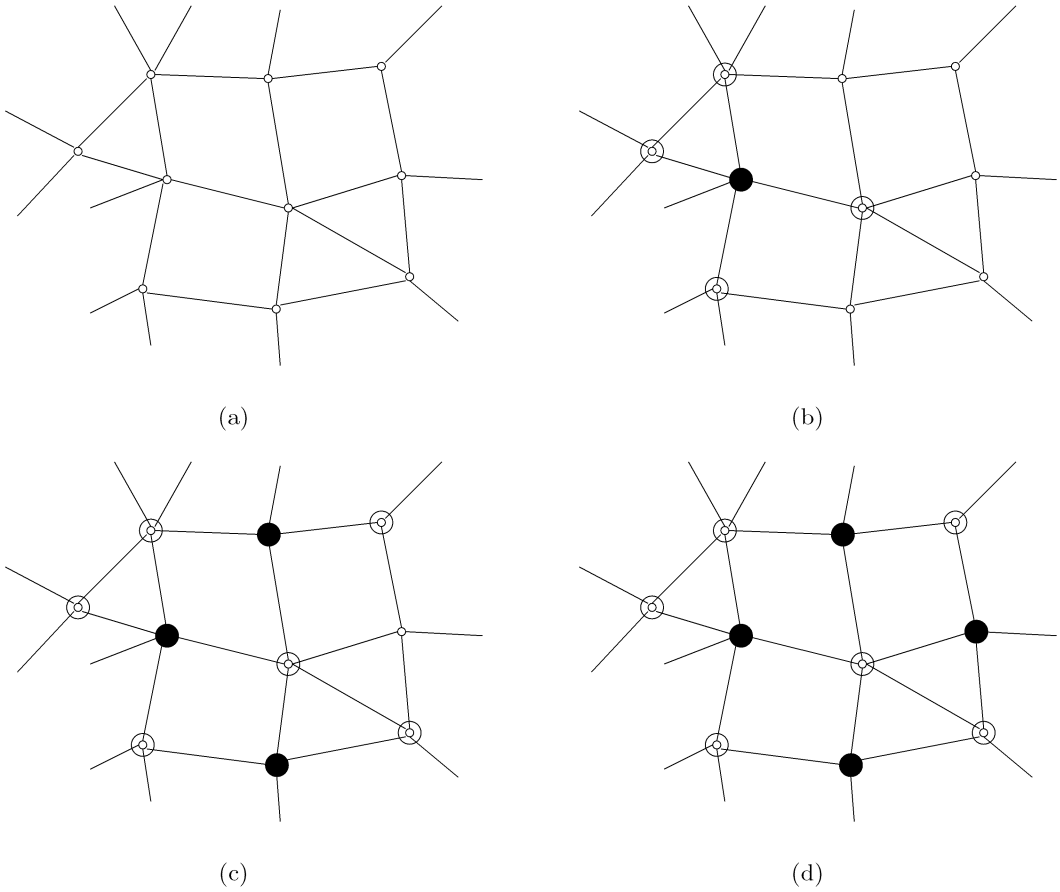


Figure III.10: Example of Coarse Node Selection by Coloring Scheme. (Black circle represents coarse node and blank circle denotes fine node)

can be represented by a linear combination of its neighbors' error.

$$m_{ii}e_i \approx -\sum_{j\neq i} m_{ij}e_j \tag{III.18}$$

Suppose node i is a fine node. Let C_i denote its coarse neighbors, D_i^s denote its fine neighbors with strong connection and D_i^s denote its fine neighbors with weak connection respectively. We want to represent the error at fine node i by errors of its coarse neighbors, such that the mapping relation between fine nodes and coarse nodes can be derived. From Equation (III.18), we have:

$$m_{ii}e_i \approx -\sum_{j\in C_i} m_{ij}e_j - \sum_{j\in D_i^s} m_{ij}e_j - \sum_{j\in D_i^w} m_{ij}e_j \tag{III.19}$$

where m_{ij} is the matrix element at i^{th} row and j^{th} column.

Since influence from weak connection neighbors is relatively insignificant, it can be moved to the diagonal terms:

$$(m_{ii} + \sum_{j\in D_i^w} m_{ij})e_i \approx -\sum_{j\in C_i} m_{ij}e_j - \sum_{j\in D_i^s} m_{ij}e_j \tag{III.20}$$

There are still fine node error components e_j on the right hand side in Equation (III.20), those error components can be approximated from the weighted average of error components at coarse nodes that are neighbors of both node i and j ($j \in D_i^s$) as illustrated in Equation (III.21).

$$e_j \approx \frac{\sum\limits_{k\in C_i} m_{jk}e_k}{\sum\limits_{k\in C_i} m_{jk}} \tag{III.21}$$

From (III.20) and (III.21), the interpolation relation between fine node i and coarse node j is derived as:

$$\omega_{ij} = -\frac{m_{ij} + \sum\limits_{l\in D_i^s} \frac{m_{il}m_{lj}}{\sum\limits_{k\in C_i} m_{lk}}}{m_{ii} + \sum\limits_{n\in D_i^w} m_{in}} \tag{III.22}$$

From (III.22), we can define the interpolation operator $P^{kh}_{(k+1)h}$ and restriction operator $P^{(k+1)h}_{kh}$ between fine level kh and coarse level $(k+1)h$. Because of the symmetry of the mapping between levels in two directions, the restriction operator is the transposition of the interpolation operator, i.e. $(P^{kh}_{(k+1)h})^T = P^{(k+1)h}_{kh}$. The coarse level $(k+1)h$ matrix is constructed as:

$$M^{((k+1)h)} = P^{kh}_{(k+1)h} M^{(kh)} P^{(k+1)h}_{kh} \tag{III.23}$$

It can be proved that coarse level matrix $M^{((k+1)h)}$ remains symmetric and positive define as long as the fine level matrix $M^{(kh)}$ is symmetric and positive definite.

Grid Reduction with Mutual Inductance

For the sake of clarity, we simplify the notation of the system equation (III.13) after Trapezoidal Rule integration approximation for RLC circuits by Modified Nodal Analysis as (III.24) by introducing vectors I_0 and V_0 to represent the right hand side.

$$\begin{bmatrix} \frac{2C}{h} + G & -A^T \\ A & \frac{2L}{h} \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} I_0 \\ V_0 \end{bmatrix} \tag{III.24}$$

As discussed before, Equation (III.24) is reformulated as (III.25) to satisfy the convergence condition of multigrid method.

$$(\frac{2C}{h} + G + \frac{h}{2} A^T L^{-1} A) V(t+h) = I_0 + \frac{h}{2} A^T L^{-1} V_0 \tag{III.25}$$

Equation (III.26) gives the formulation for branch current after the conversion.

$$I(t+h) = \frac{h}{2} L^{-1} (V_0 - AV(t+h)) \tag{III.26}$$

Let b_0 denote the right hand side of (III.25). Simplify (III.25) as (III.27), which becomes the original problem at the finest level in multigrid. Here the subscript represents the grid level. Level 0 refers to the finest level.

$$F_0 : (\frac{2C_0}{h} + G_0 + \frac{h}{2} A_0^T L_0^{-1} A_0) V_0(t+h) = b_0 \tag{III.27}$$

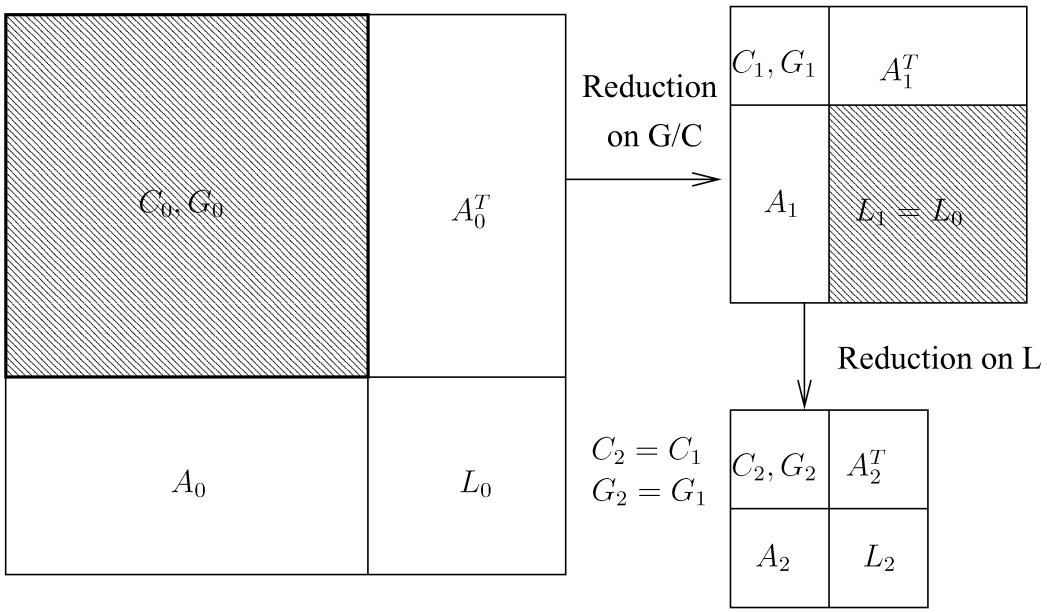


Figure III.11: Grid Reduction on G/C/L Matrix

Figure III.11 depicts a simple two-level grid reduction flow, which starts with G and C matrix reduction, followed by reduction on L matrix. The shaded submatrix is the matrix to be reduced. Please note that the hierarchical grid is constructed for L matrix instead of the L^{-1} matrix. By doing so, the system matrix at each level can still keep the similar form as the original problem.

After reduction, the matrix equation at level k is formulated as

$$F_k : (\frac{2C_k}{h} + G_k + \frac{h}{2} A_k^T L_k^{-1} A_k) V_k(t+h) = b_k \tag{III.28}$$

Only G_k, C_k and $Diag(L_k^{-1})$ are taken into account when constructing the mapping relation between levels. This is because L_k^{-1} matrix is not directly available. However, multigrid method can still converge as long as every fine node has properly representing coarse node.

III.C.2 Smoothing Operator without Inductance Matrix Inversion Approximation

The inverse inductance matrix appears in the left-hand side matrix of each level. Although the inductance matrix could be sparse, its inverse is usually

not. It is not efficient to inverse the inductance matrix directly. Various sparsification techniques of inductance matrix and its inverse can be found in [3,9,12,15]. Conversely, the proposed smoothing operator integrates the multigrid concepts and can avoid the expensive inductance matrix inversion or error-prone sparsification approximation.

Since the inverse of inductance matrix is not directly available, Jacobi iterative method is chosen as the smoothing operator to eliminate high frequency error at each level in our adaptive algebraic multigrid solver.

For the matrix equation $Mx = b$, the Jacobi method iterates as below:

$$D(x^{(i+1)} - x^i) = b - Mx^{(i)} \tag{III.29}$$

where D is the diagonal matrix of M and the superscript i represents the Jacobi iteration count.

Apply the Jacobi iterative method to Equation (III.28), we get,

$$\begin{aligned} &D_k V_k^{(i+1)}(t+h) \\ &= b_k - (\tfrac{2}{h}C_k + G_k + \tfrac{h}{2}A_k^T L_k^{-1} A_k + D_k)V_k^{(i)}(t+h) \end{aligned} \tag{III.30}$$

where $D_k = Diag(\frac{2}{h}C_k + G_k + \frac{h}{2}A_k^T L_k^{-1} A_k)$, superscript i is the Jacobi iteration count.

The matrix-vector multiplications $\frac{2}{h}C_k V_k^{(i)}(t+h)$, $G_k V_k^{(i)}(t+h)$ and $D_k V_k^{(i)} i(t+h)$ are straightforward. As to the multiplication $\frac{h}{2}A_k^T L_k^{-1} A_k V_k^{(i)}$, though the inverse of L_k matrix is not directly available, it can still be calculated implicitly as below.

Let $y = A_k V_k^{(i)}$, which can be obtained from matrix-vector multiplication, then $\frac{h}{2}A_k^T L_k^{-1} A_k V_k^{(i)}$ becomes $\frac{h}{2}A_k^T L_k^{-1} y$.

Let $L_k^{-1} y = p$, p can be solved from equation $L_k p = y$ using multigrid method. This is because the symmetric positive definite property of L_k matrix is guaranteed by multigrid grid reduction algorithm provided that the original inductance matrix L_0 is symmetric positive definite.

The final result is given in Equation (III.31).

$$\frac{h}{2}A_k^T L_k^{-1} A_k V_k^i = \frac{h}{2}A_k^T p \tag{III.31}$$

The inverse of inductance matrix shown on the right-hand side in the original problem (III.25) and branch current formulation (III.26) can be handled in the same way.

In order to complete the iteration, Jacobi method also needs the diagonal term of the left-hand side matrix in (III.28). Now the problem is reduced to how to get the diagonal term of $\frac{h}{2}A_k^T L_k^{-1} A_k$ because G_k and C_k are directly available. Since exact diagonal term is not necessary for Jacobi iterative method, the diagonal term of inverse matrix can be approximated by window-based block inversion or even the inverse of diagonal terms of the inductance matrix.

III.C.3 Adaptive Methods

Due to the non-uniformly distributed circuit switching activities and inhomogeneous circuit elements, circuits exhibit multi-rate behavior as well as latency. This phenomenon can be utilized to avoid unnecessary computations. We introduce two adaptive metrologies targeting grid construction and error smoothing respectively.

Adaptive Incremental Grid Construction

At every time point, the linear networks are solved several times for the Newton-Raphson iterations. However it is not efficient to reconstruct the hierarchical grid structure for every Newton-Raphson iteration since each time only boundary circuit models are changed. The most expensive operation during grid construction is the generation of coarse level matrices, which are actually calculated from matrix multiplication as shown in Equation (III.23).

At each time point, the coarse level matrices are generated from scratch only at the first Newton-Raphson iteration and then incrementally updated in the following Newton-Raphson iterations.

A two-level incremental update from level k to level k+1 is illustrated in Equation (III.32), where $M^{(kh)}$, $M^{((k+1)h)}$ are matrices at level k and $k+1$ in

previous Newton-Raphson iteration. $\hat{M}^{(kh)}$, $\hat{M}^{((k+1)h)}$ are matrices at level k and $k+1$ in current Newton-Raphson iteration. $\triangle M^{(kh)}$ and $\triangle M^{((k+1)h)}$ are their differences. Similarly, $P^{kh}_{(k+1)h}$, $P^{(k+1)h}_{kh}$ and $\hat{P}^{kh}_{(k+1)h}$, $\hat{P}^{(k+1)h}_{kh}$ are the mapping relations in previous and current Newton-Raphson iterations, respectively. $\triangle P^{kh}_{(k+1)h}$ and $\triangle P^{(k+1)h}_{kh}$ are their differences.

$$\begin{aligned}
\hat{M}^{((k+1)h)} &= \hat{P}^{kh}_{(k+1)h}\hat{M}^{(kh)}\hat{P}^{(k+1)h}_{kh} \\
&= (P^{kh}_{(k+1)h} + \triangle P^{kh}_{(k+1)h})(M^{(kh)} + +\triangle M^{(kh)})(P^{(k+1)h}_{kh} + \triangle P^{(k+1)h}_{kh}) \\
&= P^{kh}_{(k+1)h}M^{(kh)}P^{(k+1)h}_{kh} + \triangle P^{kh}_{(k+1)h}M^{(kh)}P^{(k+1)h}_{kh} \\
&\quad + P^{kh}_{(k+1)h}(\triangle M^{(kh)}P^{(k+1)h}_{kh} + M^{(kh)}\triangle P^{(k+1)h}_{kh}) \\
&= M^{((k+1)h)} + \triangle P^{kh}_{(k+1)h}M^{(kh)}P^{(k+1)h}_{kh} \\
&\quad + P^{kh}_{(k+1)h}(\triangle M^{(kh)}P^{(k+1)h}_{kh} + M^{(kh)}\triangle P^{(k+1)h}_{kh})
\end{aligned} \tag{III.32}$$

$$\triangle M^{((k+1)h)} = \triangle P^{kh}_{(k+1)h}M^{(kh)}P^{(k+1)h}_{kh} + P^{kh}_{(k+1)h}(\triangle M^{(kh)}P^{(k+1)h}_{kh} + M^{(kh)}\triangle P^{(k+1)h}_{kh}) \tag{III.33}$$

Hence, instead of reconstructing coarse level matrix from scratch after the first iteration at every time point, only the difference $\triangle M^{(k+1)h}$ (III.33) is calculated. Since $\triangle M^{kh}$ is much more sparse, the incremental grid construction offers significant runtime advantage.

Adaptive Smoothing

Different regions of circuits can be simulated with various time step sizes according to their activities and time constants, but SPICE has to apply a uniform time step size to the entire circuits. Many partition-based commercial fast simulators use various time step sizes for subcircuits and solve each subcircuit separately. Though saving a lot of computation overhead, partition-based methods have difficulties to guarantee the convergence and cannot capture mutual inductive/capacitive coupling effect correctly. Instead of trying to directly employ various time step sizes, the proposed approach uses adaptive smoothing operations to capture the multi-rate behavior and circuit latency. As shown in Figure III.12,

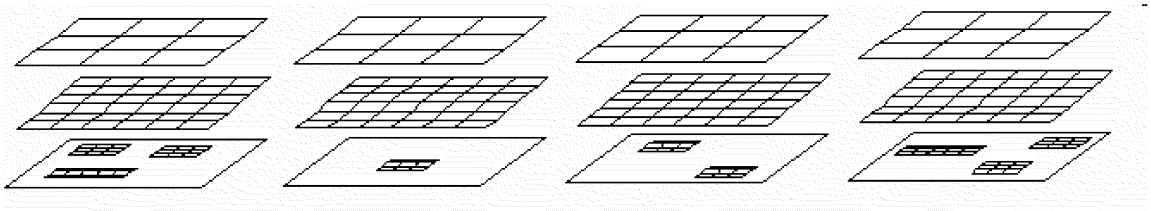


Figure III.12: Adaptive Smoothing (Grid represents smoothing operation)

active regions have more error smoothing operations than inactive regions. Convergence is guaranteed by the multigrid method, because multigrid converges as long as smoothing operations at each level can damp the high frequency error. With the adaptive smoothing, we actually use different "time step size" for active and inactive regions in the sense that inactive regions may only get chance to have error smoothed at finest level once every several time points.

III.D Application and Experimental Results

The proposed approach is implemented in C programming language. The schemes of dynamic time step size control and integration approximation follow Berkeley SPICE3f5. We use Berkeley BSIM3 for transistor model in all test cases. The linear networks are solved by the adaptive multigrid method. Four industry designs are tested on a Linux machine with 2.6 GHz CPU and 4 Gigabytes memory.

III.D.1 RLKC Power/Ground/Clock Network Example

The P/G network case contains a power ground network and a two-level H-tree clock. The circuit has 5400 inductors and 43700 mutual inductors. Coupling capacitors link between power, ground and clock networks. Figure IV.6 shows the waveform of one node on the clock. Transient simulation of 10ns is completed in 1859 seconds, which is more than 20 times faster than SPICE3 as shown in Table IV.1.

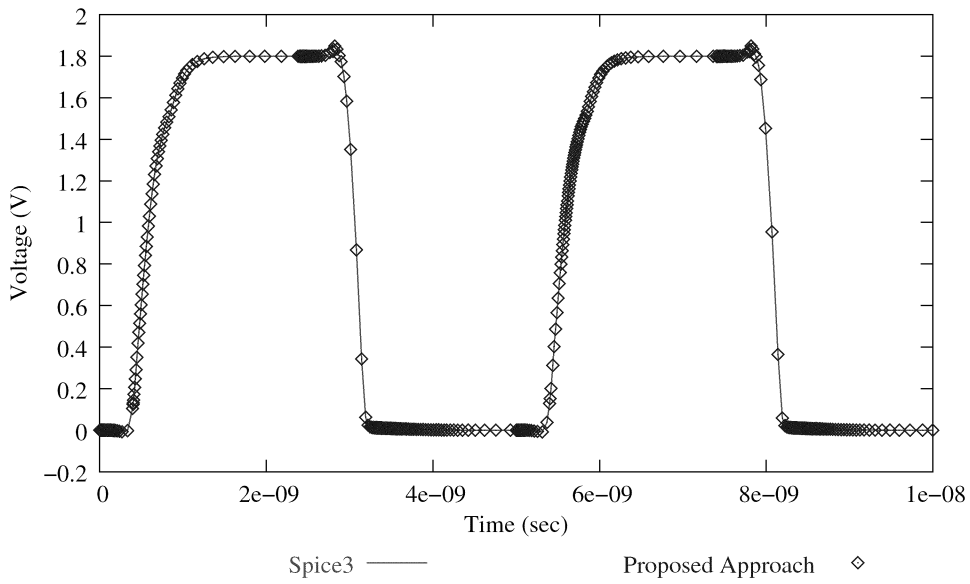


Figure III.13: Transient Waveform of RLKC Power/Ground/Clock Network Example

Table III.2: Transient Simulation Runtime

Examples	P/G Network	1K-cell	10K-cell	IO
#Nodes	29,100	10,200	123,600	1,062,000
#Transistors	720	6,500	69,000	56,600
Simulation Period	10ns	20ns	20ns	20ns
SPICE3(sec)	41323	2121	44293	N/A
Proposed Method(sec)	1859	261	3572	15337
Speedup	22.2	8.1	12.4	N/A

III.D.2 Transistor Devices Dominant Examples

Two 1K and 10K cell designs are tested. The 1k cell circuit has 10,200 nodes and 6,500 transistors. The 10k cell circuit has 123,600 nodes and 69,000 transistors. We assume that ideal power and ground supply is provided in those RC examples.

The proposed approach takes 261 seconds for 1K cell circuit and 3572 seconds for 10K cell circuit to finish 20ns transient simulations. The speedup over SPICE3 is 8.1x and 12.4x for these two examples (Table IV.1). The detailed transistor device BSIM3 model evaluation time becomes the bottleneck for these

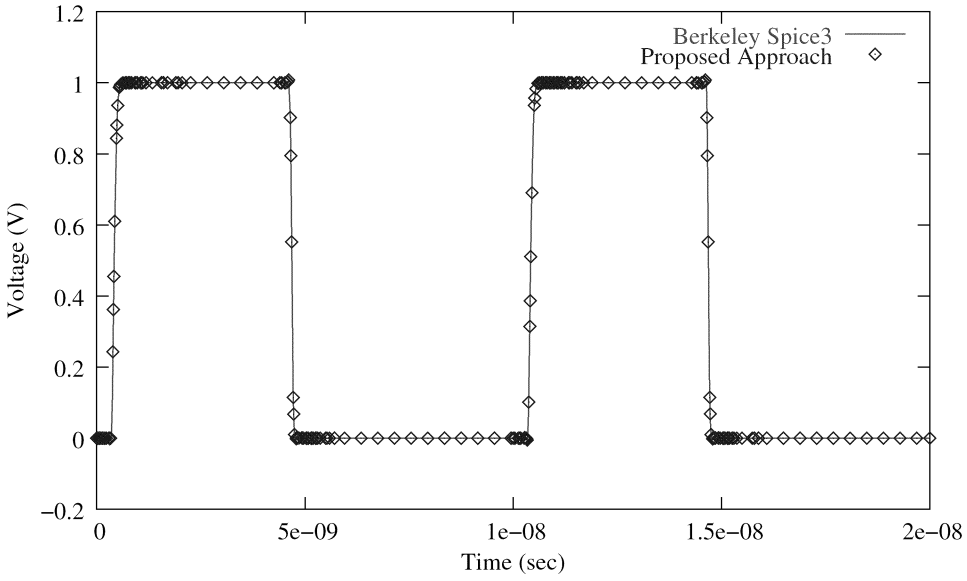


Figure III.14: Transient Waveform of 1K cell Design

transistor devices dominant circuits. Further speedup is expected if simplified device model is used.

Output waveforms are thoroughly tested against Berkeley SPICE3. We observe accurate matches in both examples. Figure III.14 shows the transient waveform of one gate output in the 1K cell design.

III.D.3 IO Cells,Power/Ground Network Example

This example contains a huge power/ground network (1 million nodes) and hundreds of big IO cells (6k transistors each). Berkeley SPICE3 and HSPICE fail to execute because of the memory size and computation time problem, while the proposed approach finished the transient analysis of 20ns in 4 hours. Figure III.15 shows input and output waveforms of one IO cell. Figure IV.7 illustrates the voltage drop of a node on on the power network.

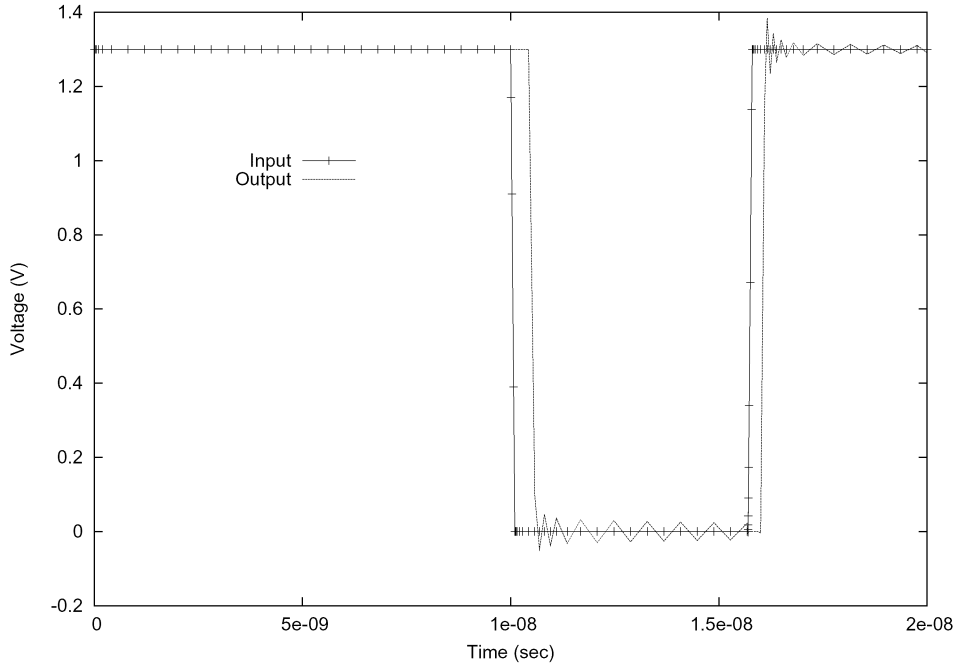


Figure III.15: Transient Waveform of IO cell

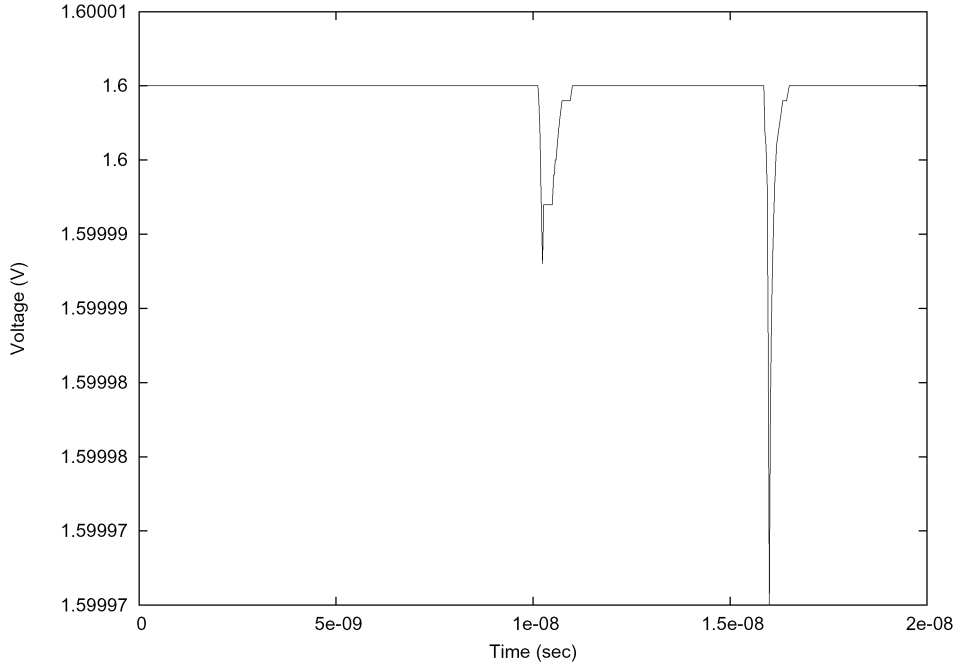


Figure III.16: Voltage drop on the power network

Chapter IV

General Operator-Splitting Method

IV.A Introduction

The operator splitting method has been adopted to partition the system based on the geometry of the physical adjacency and the locality of the processes. In 1999, Namiki and Ito [29] adopted its special form, the Alternating Direction Implicit (ADI), to simulate a two dimensional electromagnetic wave. They demonstrated the unconditional stability of the finite difference time domain analysis independent of the time step size under the proposed geometric structure. Later, Zheng et al extended the structure to three dimensions [48]. Since then, the method has been applied to tackle huge problems for finite difference time domain analysis [40]. In 2001 and 2003, Lee and Chen proposed TLM-ADI approach [20, 21] for power grid analysis, based on implicit FDTD methods.

In this Chapter, we present a generalized operator splitting method for transistor-level circuit simulation and demonstrate that the generalized method is unconditionally stable. Following the generalized approach, we partition the circuits using a network splitting algorithm with guaranteed DC paths and alternate the explicit and implicit integrations between the partitions. The splitting

algorithm partitions the circuit into structures that produce much fewer nonzero fill-ins during LU factorization. Thus direct methods remain efficient for large-scale circuits.

This chapter is organized as follows. General operator splitting method and its application on linear and nonlinear circuits are discussed in section IV.B. Section IV.C proves the unconditional stability of the proposed method. Section IV.D discusses the local truncation error (LTE) estimation and dynamic time step control. Experimental results are then demonstrated in section IV.E.

IV.B General Operator Splitting Method

The operator splitting method [2] was first introduced as a technique for solving partial differential equations. The basic idea of operator splitting can be explained by the following initial value problem (IVP) example of a simple ordinary differential equation (ODE) [34],

$$\frac{\delta u}{\delta t} = Lu \tag{IV.1}$$

where L is a linear or nonlinear operator and can be written as a linear sum of m suboperators on variable u as:

$$Lu = L_1 u + L_2 u + \cdots + L_m u \tag{IV.2}$$

Suppose $U_1, U_2, \cdots, U_m$ are updating operators on u with respect to $L_1, L_2, \cdots, L_m$ from time step n to time step $n+1$, the operator splitting approach has the form of:

$$\begin{aligned} u^{n+(1/m)} &= U_1(u^n, h/m) \\ u^{n+(2/m)} &= U_2(u^{n+(1/m)}, h/m) \\ &\cdots \\ u^{n+1} &= U_m(u^{n+(m-1)/m}, h/m)s \end{aligned} \tag{IV.3}$$

where each partial operation acts with all the terms of the original operator.

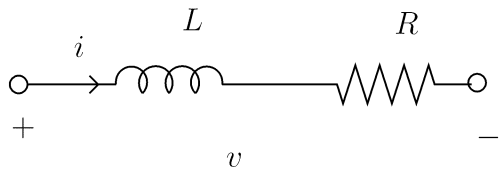


Figure IV.1: Inductor Resistor Branch Element

Our invention introduced in the next subsection generalizes the operator splitting method to graph based modeling. The generalization frees us from the geometry or locality constraints. We prove that the method is unconditionally stable.

IV.B.1 Formulation

We use a general circuit system to describe our operator splitting method. The circuit contains resistors, capacitors, and inductors with mutual couplings. We consider the resistor and inductor branch shown in Figure IV.1 as one single element with constitutional equation as $v = L\frac{di}{dt} + iR$. Hence, the system equation formulated by Modified Nodal Analysis [6, 8, 16] using Backward Euler numerical integration can be expressed as below:

$$\begin{bmatrix} \frac{C}{h} + G & -A^T \\ A & \frac{L}{h} + R \end{bmatrix} \begin{bmatrix} V(t+h) \\ I(t+h) \end{bmatrix} = \begin{bmatrix} \frac{C}{h} & 0 \\ 0 & \frac{L}{h} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t+h) \qquad \text{(IV.4)}$$

where C, L, R, G are matrices of capacitance, inductance, resistance, and conductance. Matrix A is an incidence matrix linking between the topology of capacitance nodes and inductance branches. Vectors V, I, and U describe the vector of nodal voltages, branch currents and system inputs, respectively. Scalar h is the time step from time t to $t+h$. Note that the four matrices, C, L, R, and G, are symmetric by construction and positive semi-definite because the elements: capacitor, inductor, resistor, and conductor, are non-active. In addition, we can assume that matrices C and L are positive definite for a non-degenerated case.

The generalized operator splitting formulation allows us to make partitions of the circuit. Thus, we have corresponding partitions of matrices A, R, and

G, i.e. $A = A_1 + A_2$, $R = R_1 + R_2$ and $G = G_1 + G_2$. By construction, matrices R_i and G_i for $i \in \{1, 2\}$ remain to be symmetric and positive semi-definite. Following the circuit partition, we divide the integration into two half steps and alternate the forward and backward integrations between the partitions as shown in formulation (IV.5). In the first half step, we use forward integration for the subcircuit with matrices A_2, G_2 and R_2. Then, in the second half step, we use forward integration for the subcircuit with matrices A_1, G_1 and R_1. In both half steps, the other partition is integrated by backward implicit integration.

$$\begin{cases} \begin{bmatrix} \frac{2C}{h} + G_1 & -A_1^T \\ A_1 & \frac{2L}{h} + R_1 \end{bmatrix} \begin{bmatrix} V(t + \frac{h}{2}) \\ I(t + \frac{h}{2}) \end{bmatrix} = \\ \qquad\qquad \begin{bmatrix} \frac{2C}{h} - G_2 & A_2^T \\ -A_2 & \frac{2L}{h} - R_2 \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t + \frac{h}{2}) \\ \begin{bmatrix} \frac{2C}{h} + G_2 & -A_2^T \\ A_2 & \frac{2L}{h} + R_2 \end{bmatrix} \begin{bmatrix} V(t + h) \\ I(t + h) \end{bmatrix} = \\ \qquad\qquad \begin{bmatrix} \frac{2C}{h} - G_1 & A_1^T \\ -A_1 & \frac{2L}{h} - R_1 \end{bmatrix} \begin{bmatrix} V(t + \frac{h}{2}) \\ I(t + \frac{h}{2}) \end{bmatrix} + U(t + h) \end{cases} \tag{IV.5}$$

Let $P_1 = \begin{bmatrix} G_1 & -A_1^T \\ A_1 & R_1 \end{bmatrix}$, $P_2 = \begin{bmatrix} G_2 & -A_2^T \\ A_2 & R_2 \end{bmatrix}$, and $S = \begin{bmatrix} \frac{2C}{h} & 0 \\ 0 & \frac{2L}{h} \end{bmatrix}$, then the notation of the two half steps of operator splitting formulation (IV.5) can be simplified as:

$$\begin{cases} (P_1 + S)X(t + \frac{h}{2}) &= -(P_2 - S)X(t) + U(t + \frac{h}{2}) \\ (P_2 + S)X(t + h) &= -(P_1 - S)X(t + \frac{h}{2}) + U(t + h) \end{cases} \tag{IV.6}$$

where $X = \begin{bmatrix} V \\ I \end{bmatrix}$.

IV.B.2 Splitting Operation

The performance of direct methods such as LU decomposition can still beat those of iterative methods for small circuits with up to tens of thousands of nodes. Direct methods become prohibitive for large circuits because of the significant amount of nonzero fill-ins generated during factorization. However, it can be proved that the LU decomposition method does not create nonzero fill-ins for circuits in tree/forest structure if nodes elimination always starts from leaves. The elimination order can be captured by ordering algorithms based on minimum degrees. Following this observation, the proposed operator splitting algorithm tries to split the circuits into two partitions in structures close to tree or forests such that the number of nonzero fill-ins is minimized. Even though general circuits may not be able to get optimized partitions in terms of the number of nonzero fill-ins because of its structure limitation or the restriction of DC paths (Splitting algorithm should not generate floating nodes at DC stage), the number of overall nonzero fill-ins is significantly reduced for most circuits.

Applications on Linear Circuits

Only resistive connections are considered during partition. Capacitors and inductors are duplicated into both partitions. Resistors are divided into two partitions using graph theory algorithms. In order to obtain DC convergence, some adjustments are needed to ensure that every node in both partitions has a DC path to some voltage source. When solving each partition, the rest of circuit is modelled as equivalent current sources, following the operator splitting formulation (IV.5). Detailed algorithm is discussed in section IV.B.2.

Applications on Circuits with transistors

We extend the algorithm to handle circuits with transistors. In typical digital circuits, transistors are grouped as various gates. Taking into consideration the nonlinear property of transistor devices and gates, the proposed approach does

not split a single transistor or gate into different partitions; instead, each partition has a full-version of all transistor devices. In other words, transistor devices are duplicated in both partitions and solved at every half time point.

The splitting algorithm actually regards each gate as a super node. The details inside each gate are invisible to the splitting algorithm. The same splitting operation for linear circuits is applied on the super node structure instead of the original circuits.

Moreover, the introduction of super node brings some advantages during LU decomposition if all the internal nodes in one gate are eliminated together. Even though each gate may have many internal nodes, it usually has only a few input/output nodes connecting to the outside, which implies that the nonzero fill-ins are confined inside each gate and will not propagate to outside if all the internal nodes in the same gate are eliminated together during LU decomposition. However, the ordinary minimum degree algorithm or the Markowitz Product method used in SPICE3 does not have a global view of the circuit structure and introduce many unnecessary nonzero fill-ins. We modified the ordering algorithm in SPICE3 to incorporate the super node concept.

Network Splitting with Guaranteed DC Paths

We define an undirected graph $G = (V, E)$ to represent the circuit structure. There are two kinds of nodes in this graph: super node and branch node. Super node denotes end point of resistors in large linear networks or a single gate. Branch node represents end point of resistors on signal wires connecting gates in the circuit. The edge denotes the resistor branch in the circuit since only resistors are divided into partitions.

Figure IV.3 gives the undirected graph representation of a simple digital circuit. The original circuit shown in Figure IV.3(a) is mapped to an undirected graph in Figure IV.3(b). Figure IV.3(c) demonstrates one possible splitting result of the graph, where dashed lines and solid lines denote two different partitions.

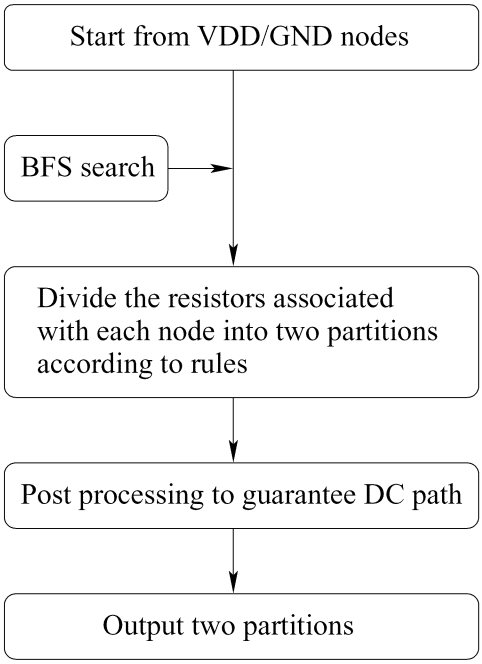
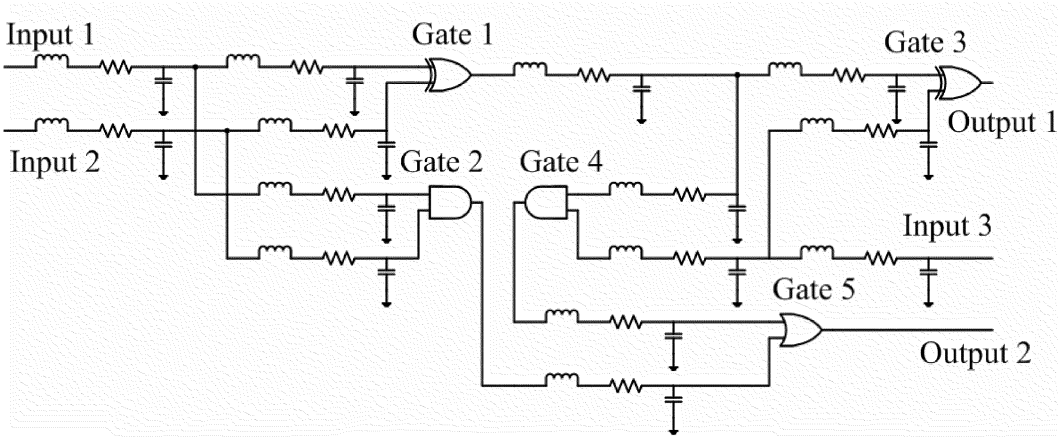


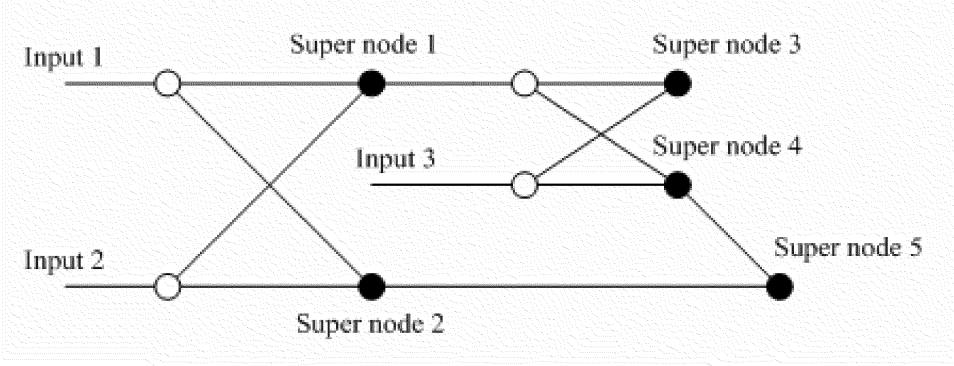
Figure IV.2: Splitting Algorithm Flow

The splitting algorithm divides the graph into two partitions. The objective is to minimize the total number of non-zero fill-ins of both partitions generated in LU decomposition. The basic idea of our rule-based splitting algorithm is to distribute the edges of every node into different partitions and avoid loops as much as possible, because sparse and tree-like structures produce much fewer nonzero fill-ins.

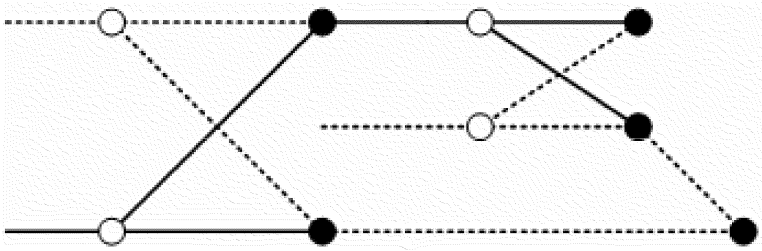
Figure IV.2 summarizes the splitting algorithm flow. The input information includes the undirected graph, super node identification and VDD/GND nodes set. There are two main steps, breath first search (BFS) partition and DC path post-processing. In the BFS partition step, we start from VDD/GND nodes simultaneously, go through all the nodes in the graph using BFS and divide the edges of every node into two partitions according to the partition rules defined below. Nodes and edges are associated with labels to record the partition and DC path connection status. Based on the labelling information, the partition rules are defined to benefit DC path available for all nodes. In the post-processing step, we



(a) Example of a Simple Digital Circuit



(b) Undirected Graph Representation



(c) Splitting Result of The Undirected Graph

Figure IV.3: Example for Undirected Graph Representation and Splitting

adjust the partition for nodes without a DC path to VDD/GND.

We define seven types of node labels:

- CONNECTED
- UNCONNECTED
- ZERO_UNCONNECTED
- ONE_UNCONNECTED
- ZERO_ONE_UNCONNECTED
- ZERO_CONNECTED
- ONE_CONNECTED

where "ZERO" and "ONE" represent different partitions. The label describes the partition and DC path status of the node. For example, label "ZERO_UNCONNECTED" means the node has an edge in partition ZERO without a DC path to VDD/GND.

Similarly, we define five types of edge labels

- UNCONNECTED
- ZERO_UNCONNECTED
- ONE_UNCONNECTED
- ZERO_CONNECTED
- ONE_CONNECTED

There are four partition rules for the splitting process.

1. Branch rule: the edges in one branch belong to the same partition. In the undirected graph, a branch is consisted of edges connected by branch nodes. Branch rule assigns nodes on the same signal wires into one partition, which will accelerate the nonlinear convergence.

2. Degree rule: the edges of node with degree two should be assigned to the same partition. This is because the line structure would not cause many non-zero fill-ins and will be propitious to provide DC path in both partitions.

3. Loop rule: the loop will be avoided in both partitions if possible. Edge loops will potentially introduce certain number of non-zero fill-ins.

4. Balance rule: the edges for each node in the graph will be evenly divided into two partitions. Thus, each partition will be much simpler than the original graph.

As an example, Figure IV.4 illustrates the step by step splitting process on a 6x6 mesh shown in Figure IV.4(b). The legend used to represent different types of nodes and edges is given in Figure IV.4(a). Figure IV.4(c) - IV.4(g) reveal the stepwise changes of splitting status for all the nodes and edges in BFS partition stage. Figure IV.4(h) gives the final splitting result after the post-processing step. Both partitions in the final result have a tree/forest structure, which will greatly benefits the LU decomposition.

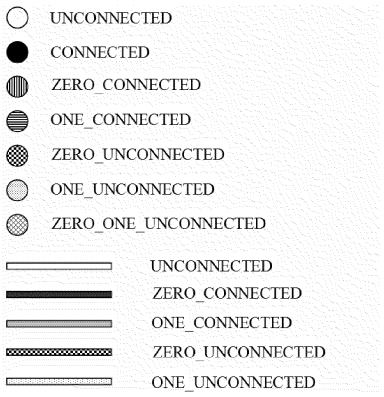
IV.C Unconditional Stability Analysis

For the analysis of the error propagation, we can ignore the inputs in the operator splitting formulation (IV.5). We then combine the two half steps and reduce the two equations in (IV.5) to a recursive formula:

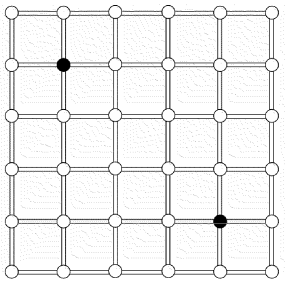
$$X_{(k+1)} = \Lambda X_{(k)} \tag{IV.7}$$

where $\Lambda = (P_2 + S)^{-1}(P_1 - S)(P_1 + S)^{-1}(P_2 - S)$.

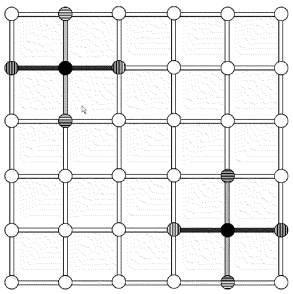
In the proof of the convergence, we use the norm $\|x\|_{s^{-1}} = (x^T S^{-1} x)^{1/2}$. Note that matrix S^{-1} is positive definite because matrix S is positive definite and the inverse of a positive definite matrix remains to be positive definite. We first state the theorem of the unconditional stability. The proof of the statement is assisted by the lemmas, which follow the theorem.



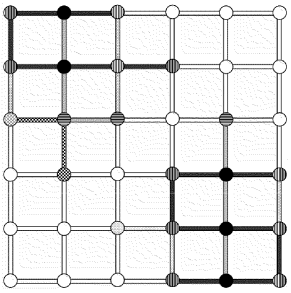
(a) Node and Edge Labels



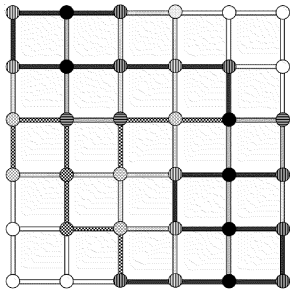
(b) 6 by 6 Mesh.Black Nodes Represent VDD/GND



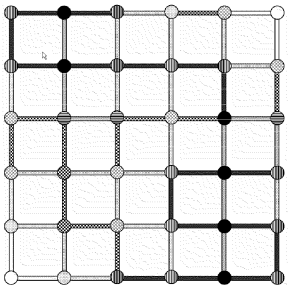
(c) Splitting Step 1



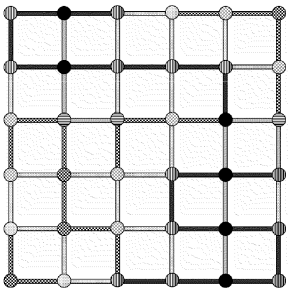
(d) Splitting Step 2



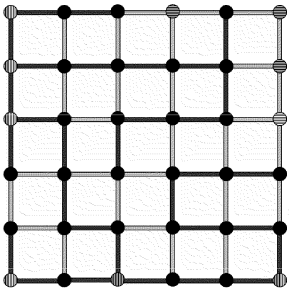
(e) Splitting Step 3



(f) Splitting Step 4



(g) Splitting Step 5



(h) Final Splitting Result

Figure IV.4: 6x6 mesh splitting

Theorem IV.C.1 *The operator splitting formula (IV.5) is A-stable (stable independent of the step size h)*

Proof: Let $\rho(\Lambda) = max(|\lambda_i(\Lambda)|)$, where $\lambda_i(\Lambda)$ is the i^{th} eigenvalue of matrix Λ. The proposed operator splitting approach is stable if $\rho(\Lambda) <= 1$.

From Lemma IV.C.4, we have

$\|(P_1 - S)(P_1 + S)^{-1}x\|_{S^{-1}} \leq \|x\|_{S^{-1}}$ and

$\|(P_2 - S)(P_2 + S)^{-1}x\|_{S^{-1}} \leq \|x\|_{S^{-1}}$.

Let $\rho(\tilde{\Lambda}) = (P_1 - S)(P_1 + S)^{-1}(P_2 - S)(P_2 + S)^{-1}$

From Lemma IV.C.2 and IV.C.3, we can deduce: $\rho(\Lambda) = \rho(\tilde{\Lambda}) \leq 1$.

Lemma IV.C.2 $\rho((P_2 + S)^{-1}(P_1 - S)(P_1 + S)^{-1}(P_2 - S)) = \rho((P_1 - S)(P_1 + S)^{-1}(P_2 - S)(P_2 + S)^{-1})$

Proof: We can derive that $\rho(AB) = \rho(BA)$ if matrix A or B is nonsingular. Thus, we prove the lemma by setting $A = (P_2+S)^{-1}$ and $B = (P_1-S)(P_1+S)^{-1}(P_2-S)$.

Lemma IV.C.3 *Given a real matrix M, if $\|Mx\|_{S^{-1}} \leq \gamma\|x\|_{S^{-1}}$ for all real x, then $\rho(M) \leq \gamma$.*

The proof can be found in [45].

Lemma IV.C.4 $\|(P_i - S)(P_i + S)^{-1}x\|^2_{S^{-1}} \leq \|x\|^2_{S^{-1}}$ *for* $i \in \{1, 2\}$ *and every real vector x.*

Proof:

$\|(P_i - S)(P_i + S)^{-1}x\|^2_{S^{-1}} \leq \|x\|^2_{S^{-1}}$ is equivalent to $\|(P_i - S)y\|^2_{S^{-1}} \leq \|(P_i + S)y\|^2_{S^{-1}}$ where $y = (P_i + S)^{-1}x$.

We expand the inequality expression stated in the Lemma according to the norm definition as:

$$y^T(P_i^T - S^T)S^{-1}(P_i - S)y \leq y^T(P_i^T + S^T)S^{-1}(P_i + S)y \tag{IV.8}$$

The inequality is further reduced to:

$$y(\ P_i + P_i^T)y^T \geq 0 \tag{IV.9}$$

which is true since $P_i + P_i^T$ is positive semidefinite for $i \in \{1, 2\}$.

IV.D Local Truncation Error and Time Step Control

Although the general operator splitting approach is A-stable, the local truncation error still needs to be controlled below the error tolerance in order to ensure the accuracy. By estimating the local truncation error at every time point, we can dynamically adjust the time step to control the local truncation error.

IV.D.1 Review of Time Step Control in SPICE

Before we derive the time step estimation formula for the proposed general operator splitting approach, let us review the dynamic time step control in Berkeley SPICE. When SPICE is about to move to the next time point during transient analysis, the next time step is estimated from the maximum local truncation error (LTE) at the present time point. The local truncation error is defined as the difference between the calculated solution and the estimated exact solution derived from the Taylor expansion of previous time point solutions. The local truncation error is examined at every energy storage element and the maximum is picked to estimate the time step.

The exact solution x_{n+1} at time point $n+1$ with time step h by Taylor Expansion is given as:

$$x_{n+1} = x_n + h_n \dot{x}_n + \frac{h_n^2}{2}\ddot{x}(\xi) \tag{IV.10}$$

or

$$\dot{x}_{n+1} = \dot{x}_n + h_n \ddot{x}(\xi) \tag{IV.11}$$

when in terms of $\dot{x}_{n+1}$.

The solution $\hat{x}_{n+1}$ derived from Backward Euler integration approximation is :

$$\hat{x}_{n+1} = x_n + h_n \dot{x}_{n+1} - \frac{h_n^2}{2}\ddot{x}(\xi) \tag{IV.12}$$

The local truncation error (LTE) is defined as the difference between $\hat{x}_{n+1}$ and x_{n+1}:

$$\begin{aligned} \epsilon_x &= \hat{x}_{n+1} - x_{n+1} \\ &= -\tfrac{h_n^2}{2}\ddot{x}(\xi) \end{aligned} \tag{IV.13}$$

or

$$\epsilon_{\dot{x}} = -\frac{h_n}{2}\ddot{x}(\xi) \tag{IV.14}$$

when in terms of $\dot{x}$.

The $\ddot{x}$ is approximated by the divided difference.

Define

$$\begin{aligned} DD_1 &= \tfrac{x_{n+1}-x_n}{h_n} \\ DD_k &= \tfrac{DD_{k-1}(t_{n+1})-DD_{k-1}(t_n)}{\sum_1^k h_{n+1-i}} \end{aligned} \tag{IV.15}$$

Then in general, the k^{th} order differential of x is approximated as:

$$\frac{d^k x}{dt^k}(\xi) = k!DD_k \tag{IV.16}$$

To ensure the consistency, the local truncation error should not exceed the error tolerance E_D:

$$|\epsilon_{\dot{x}}| \leq E_D \tag{IV.17}$$

Hence, the time step estimation for Backward Euler integration approximation is derived as:

$$h_n \leq \frac{2E_D}{|\ddot{x}(\xi)|} = \frac{E_D}{|DD_2|} \tag{IV.18}$$

IV.D.2 Dynamic Time Step Control for General Operator Splitting Method

Similar to the time step control of SPICE, we estimate the time step from local truncation error. However, the operator splitting formation prevents us from calculating local truncation error element by element. Instead, the local truncation error is estimated via matrix operations.

Given the system equation (IV.19) before numerical integration,

$$\begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} \dot{V}(t) \\ \dot{I}(t) \end{bmatrix} = \begin{bmatrix} -G & A^T \\ -A & -R \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} + U(t) \tag{IV.19}$$

Let $M = \begin{bmatrix} C & 0 \\ 0 & L \end{bmatrix}$, $N = \begin{bmatrix} -G & A^T \\ -A & -R \end{bmatrix}$ and ignore the input vector U, Equation (IV.19) can be simplified as:

$$\begin{aligned} M\dot{X} &= NX \\ \dot{X} &= M^{-1}NX \end{aligned} \tag{IV.20}$$

.

The exact analytic solution X with time step h is derived as below:

$$\begin{aligned} X_{n+1} &= e^{M^{-1}Nh}X_n \\ &= (1 + M^{-1}Nh + \tfrac{h^2(M^{-1}N)^2}{2} + \tfrac{h^3(M^{-1}N)^3}{6} + O(h^4))X_n \end{aligned} \tag{IV.21}$$

The general operator splitting approach can also be formulated as:

$$\frac{M}{h}(\hat{X}_{n+1} - X_n) = N_1\hat{X}_{n+1} + N_2X_n \tag{IV.22}$$

where $N = N_1 + N_2$, N_1 represents the the partition applied Backward Euler and N_2 denotes the partition applied Forward Euler integration method.

The analytic solution $\hat{X}$ of operator splitting approach is derived as below:

$$(\frac{M}{h} - N_1)\hat{X}_{n+1} = (\frac{M}{h} + N_2)X_n \tag{IV.23}$$

$$\begin{aligned} \hat{X}_{n+1} &= (\tfrac{M}{h} - N_1)^{-1}(\tfrac{M}{h} + N_2)X_n \\ &= (I - hM^{-1}N_1)^{-1}(I + hM^{-1}N_2)X_n \\ &= (I + hM^{-1}N_1 + h^2M^{-1}N_1M^{-1}N_1 + h^3M^{-1}N_1M^{-1}N_1M^{-1}N_1 + \cdots) \\ &\quad (I + hM^{-1}N_2)X_n \\ &= (I + hM^{-1}N_1 + h^2M^{-1}N_1M^{-1}N_1 + h^3M^{-1}N_1M^{-1}N_1M^{-1}N_1 + \cdots)X_n \\ &\quad +(hM^{-1}N_2 + h^2M^{-1}N_1M^{-1}N_2 + h^3M^{-1}N_1M^{-1}N_1M^{-1}N_2 + \cdots)X_n \\ &= [I + hM^{-1}N + h^2M^{-1}N_1M^{-1}N + O(h^3)]X_n \end{aligned} \tag{IV.24}$$

The local truncation error is the difference between operator splitting solution $\hat{X}$ and exact solution X:

$$LTE = \|\hat{X} - X\| = \|h^2M^{-1}(\tfrac{N}{2} - N_1)\dot{X}_n + O(h^3)\| \tag{IV.25}$$

The local truncation error at each time step should not exceed the error tolerance. If we ignore the high order terms of local truncation error, the time step when forward Euler integration is applied to partition corresponding to N_1 is estimated as:

$$h_1 < \sqrt{\frac{\text{Error Tolerance}}{\|M^{-1}(\frac{N}{2} - N_1)\dot{X}_n\|}} \tag{IV.26}$$

Similarly, the time step when forward Euler integration is applied to partition corresponding to N_2 is estimated as:

$$h_2 < \sqrt{\frac{\text{Error Tolerance}}{\|M^{-1}(\frac{N}{2} - N_2)\dot{X}_n\|}} \tag{IV.27}$$

And the new time step h is twice of the minimum time step of each partition:

$$h = 2min(h_1, h_2) \tag{IV.28}$$

IV.E Experimental Results

The proposed approach is implemented in C programming language. For all circuits presented in this section, we compare the proposed approach with Berkeley SPICE3 using BSIM3 models for transistor devices. We did not compare the result with commercial fast simulators because we do not trade off any accuracy for speed. Convergence and accuracy are guaranteed. Examples are tested on a Linux machine with 2.6 GHz CPU and 4 Gigabytes memory.

IV.E.1 Power Network with Nonlinear Current Sinks

We test a number of RLC power networks with size ranging from 11k nodes to 160k nodes. Various transistor gates draw current from the power networks. Those power networks are approximately in mesh structures. The splitting algorithm results in very limited nonzero fill-ins and we observe linear runtime of the proposed method. The CPU runtime and speedup are given in Table IV.1. One

Table IV.1: Transient Simulation Runtime

Examples	circuit1	circuit2	circuit3	circuit4	Power and Clock Network	Large Power Network	1K-cell	10K-cell
#Nodes	11,203	41,321	92,360	160,657	29,100	615,446	10,200	123,600
#Transistors	74	512	1,108	2,130	720	0	6,500	69,000
Simulation Period	10ns	10ns	10ns	10ns	10ns	10ns	20ns	20ns
SPICE3(sec)	602.44	8268.92	39612.32	N/A	12015	N/A	2121	44293
Operator Splitting(sec)	74.64	305.38	681.18	1356.21	649.5	4083.7	415.9	3954.7
Speedup	8.1x	27.1x	58.2x	N/A	18.5x	N/A	5.1x	11.2x

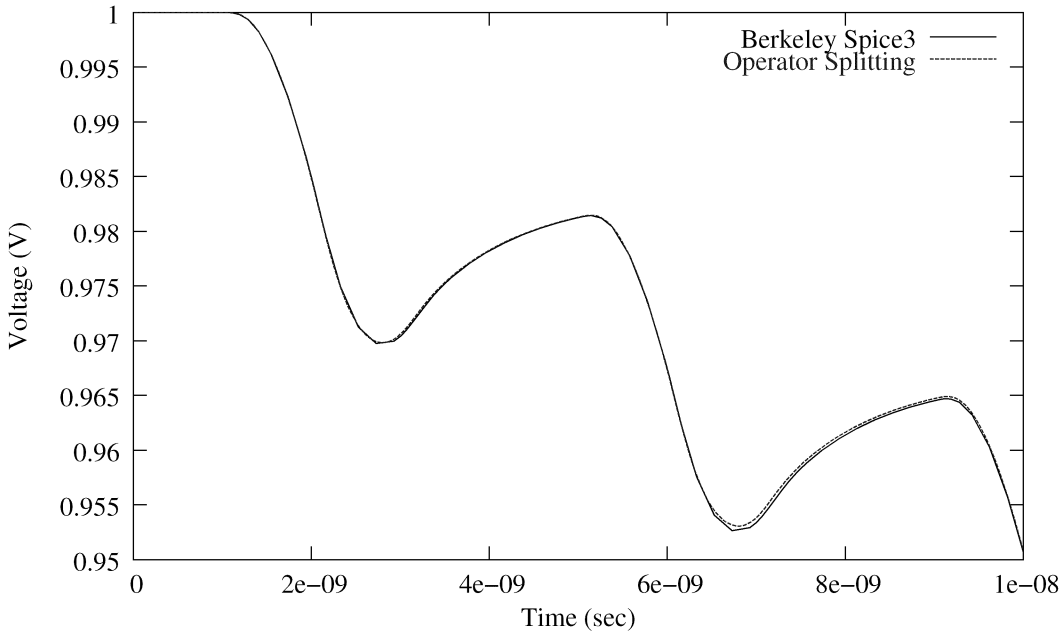


Figure IV.5: Transient Response of Circuit3

or two orders of magnitude speedup (8.1x to 58.2x) against SPICE3 is obtained. The transient waveform circuit3 is given in Figure IV.5.

Since we only replace the LU decomposition procedure inside the SPICE3, other overhead such as device evaluation and dynamic time step control takes more than 30% of the total runtime, thus limits the overall speedup.

IV.E.2 Power and Clock Network

The Power and clock network case contains a RLC power ground network and a two-level H-tree clock. Figure IV.6 shows the voltage drop at one node of the power network. Transient simulation of 10ns is completed in 649.5 seconds, which is 18.5 times faster than SPICE3 as shown in Table IV.1.

IV.E.3 Large Power Network Example

This example contains a huge RC power network (0.6 million nodes) with irregular structure (some nodes have thousands of neighbors). The switching activities that draw current from the power network are modeled as piecewise

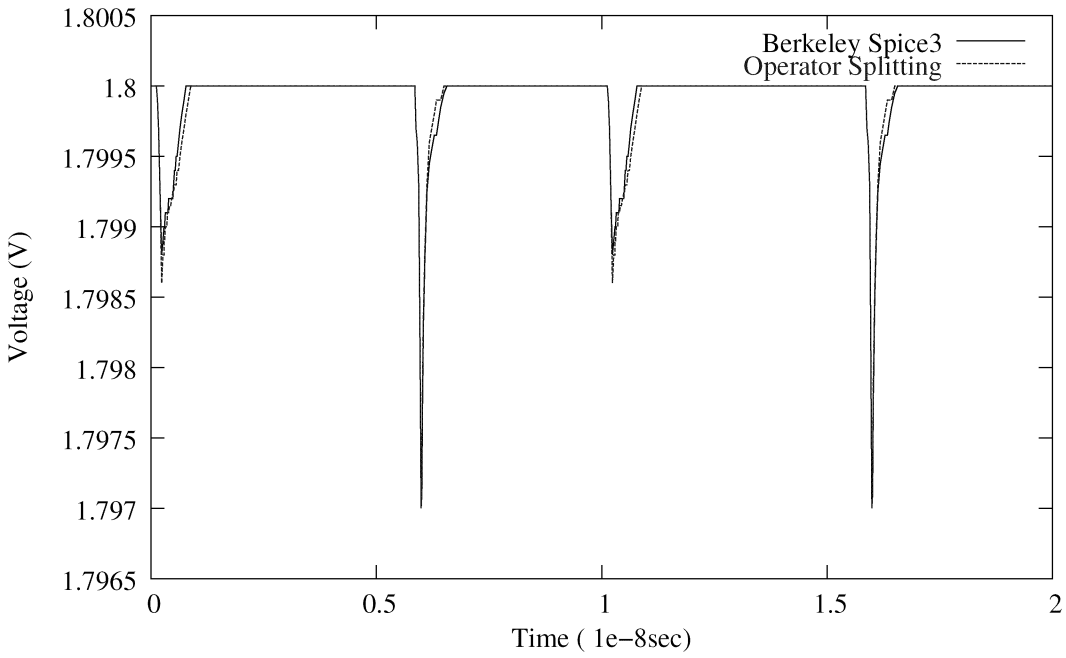


Figure IV.6: Voltage Drop of RLC Power and Clock Network Example

linear current waveform. Berkeley SPICE3 fails to execute because of the memory size and computation time problem. The operator splitting approach finished the transient analysis of 10ns in just 4083 seconds. Figure IV.7 illustrates the voltage drop of a node on the power network.

IV.E.4 ASIC designs

Two 1K and 10K cells ASIC designs are tested to demonstrate the proposed approach's ability of handling transistor dominated nonlinear circuits. The 1k cell circuit has 10,200 nodes and 6,500 transistors. The 10k cell circuit has 123,600 nodes and 69,000 transistors. We assume that ideal power and ground supplies are provided in those RC examples. The proposed approach takes 415.9 seconds for 1K cell circuit and 3954.7 seconds for 10K cell circuit to finish 20ns transient simulations. The speedup over SPICE3 is 5.1x and 11.2x for these two examples (Table IV.1). We observe accurate waveform match for both examples. Figure IV.8 shows the transient waveform of one gate output in the 1K cell design.

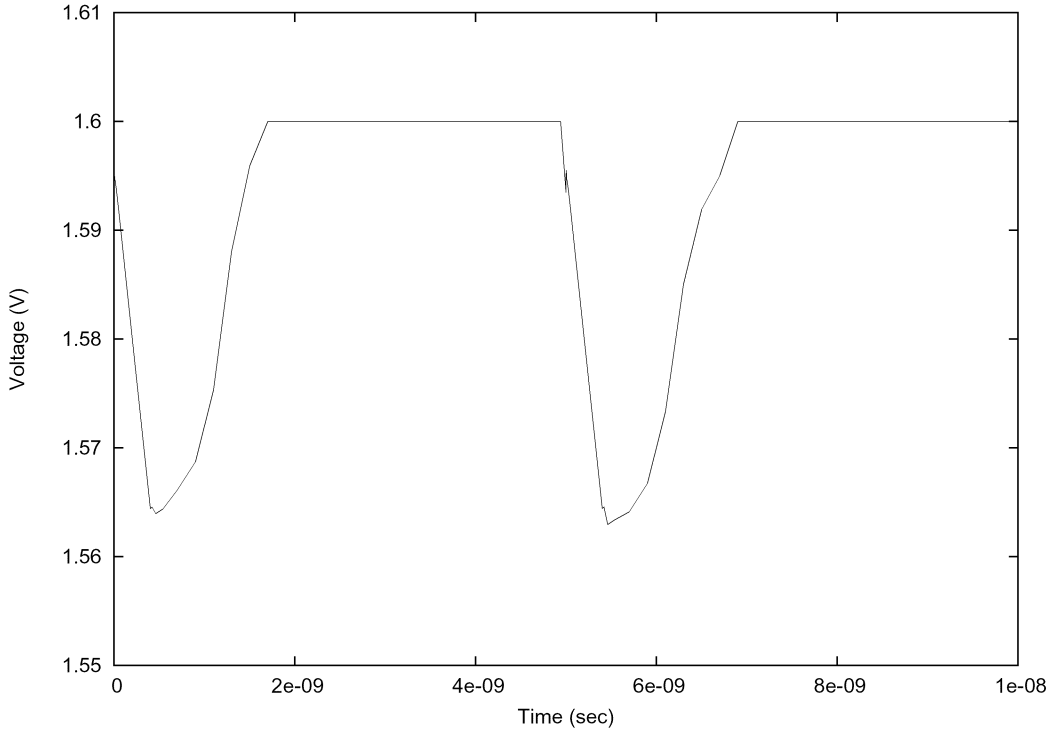


Figure IV.7: Voltage Drop on The Power Network

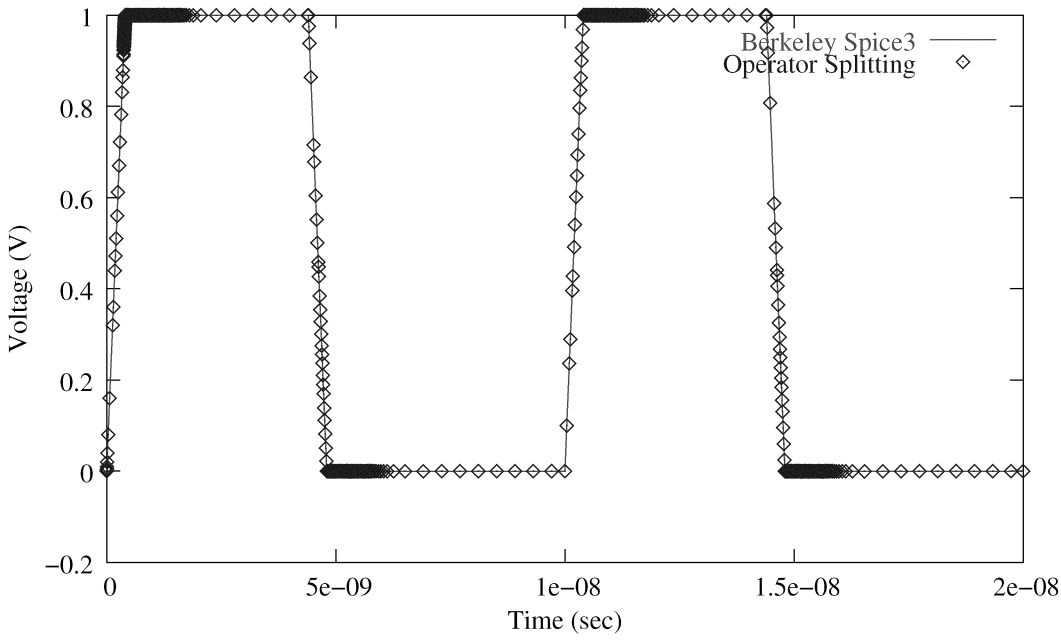


Figure IV.8: Transient Waveform of 1K Cell Design

Chapter V

Conclusion

V.A Dissertation Contribution

According to the famous Moores's law, the number of transistors on a silicon die and the clock frequency double every two years. As we have entered the nanometer era, the clock frequency has reached several giga Hz and the new process technologies can pack hundreds of millions of transistors into a single silicon die. Nowadays, designers pay more attentions to the signal integrity problems such as interconnect delay, crosstalk, voltage drop and ground bounce. Inductance effect, which is usually ignored before at chip-level simulation, becomes an issue as the clock frequency reaches giga Hz range. Transistor devices under advanced processing technologies exhibit more nonlinearity and require more accurate device model. All these factors raise significant challenges for circuit simulators. However the biggest challenge is the design complexity. Tens of millions of transistors and other elements introduce serious memory and runtime problems for simulation tools. SPICE along with other SPICE-like simulation tools cannot perform large-scale circuit analysis. Their capacity is limited to about 50,000 transistors, due to the super linear complexities.

Inspired by the circuit simulation challenges, we propose two new simulation technologies targeting large-scale digital circuits with large interconnect of

strong coupling.

In Chapter III, we present an efficient technique to solve linearized circuit equation. Taking into account the different physical nature and the resulting matrix properties of linear networks and nonlinear components, we solve linear and nonlinear portions of circuits separately inside the Newton-Raphson nonlinear iterations. The linear and nonlinear interfaces are dynamically abstracted as equivalent circuit models and iterations between linear and nonlinear portions of circuits ensure the global convergence. The algebraic multigrid method is used to solve the large linear networks especially power and ground distribution networks. We extend the standard algebraic multigrid method (AMG) to handle general linear RLC circuits with self and mutual inductors. Several adaptive strategies are adopted to take the advantages of the circuit latency.

Chapter IV introduces a new A-stable numerical integration procedure. The new integration method splits the circuits into two halves and alternates the implicit and explicit integration on each portion. Although the original circuit is difficult to solve using direct method, each half is much easier. Hence, the most robust direct methods such as LU decomposition can still be applied to large-scale circuits. This dissertation discusses the detailed splitting algorithm as well as the dynamic time step size control algorithm that ensures the consistency of new numerical integration method.

Part of the dissertation has been published in [49, 50]. The efficient transistor-level transient analysis approaches proposed in this dissertation have been implemented as complete software packages and successfully applied to the analysis of large-scale nanometer designs. The packages were implemented within the framework of Berkeley SPICE3f5. They include all SPICE time-domain analysis options and commands and support most circuit elements available in SPICE except for the distributed circuit elements such as transmission line and S-parameter. The proposed approaches can perform analysis of interconnect delay, clock distribution, power grid analysis and simultaneous switching noise.

Unlike partition-based fast simulation methods, both approaches proposed in this dissertation offer guaranteed SPICE-level accuracy. This is because we simulate the entire system without any simplified assumptions on device model, linearization process and numerical integration. Coupling noise and inductance effect are well-captured. There is no trade off accuracy for speedup and the rigorous global convergence check guarantees the accuracy. Both approaches proposed in the dissertation offer significant runtime advantage. Orders of magnitude speedup over SPICE is observed. Moreover, both approaches are expandable. Some fast simulation techniques appeared in commercial tools and literature can be combined without introducing any confliction.

V.B Future Work

V.B.1 Two-State Newton-Raphson Approach

The two-stage Newton-Raphson has several limitations that need future improvements:

Accuracy of Boundary Modelling for Linear Network Unlike the exactly derived nonlinear component boundary conductance, the equivalent boundary conductance for large linear networks is approximated from boundary conditions of previous Newton-Raphson iterations. Except for boundary nodes directly connecting to voltage sources, discrete approximation does not provide accurate estimation for boundaries with rapidly changing voltage level, such as output ports of switching gates. Although it will not lead to wrong solutions due to the Newton-Raphson and linear-nonlinear boundaries convergence check and the dynamic time step control, inaccurate boundary models increase the number of Newton-Raphson and linear-nonlinear iterations and even cause unnecessary smaller time step size. One possible future direction is to incompletely factorize the matrix of large linear network and derive the equivalent conductance in the similar manner of nonlinear compo-

nents. However, since incomplete factorization drops relatively insignificant off-diagonal terms during elimination, the resulting equivalent conductance is not an exact solution and the error relates to the drop-off threshold and node ordering. Smaller drop-off threshold generates more accurate result, but the complexity will be close to the complete LU decomposition, which is prohibitive for large linear networks.

Complexity of Boundary Modelling for Nonlinear Components One problem of the nonlinear component boundary conductance modeling is the complexity issue when the boundary nodes take up a considerable percentage of the total nodes in large nonlinear components. Under this circumstance, reordering all the boundaries to the bottom of matrices, which is discussed in Section III.B.1, may not help much. Splitting the large nonlinear component into smaller ones will reduced the modeling overhead. If we extend the linear-nonlinear iterations to between nonlinear components of circuits also, global convergence is still guaranteed at a cost of more iterations between different portions of circuits.

Convergence The two-stage Newton-Raphson approach may encounter convergence problem for circuits with strong feedbacks. If the feedback loops are not in the same linear or nonlinear portion, the two-stage Newton-Raphson method may converge slowly or even fail to converge. This is the main cause that limits the application from analog components. Currently, we go around this problem by regarding the analog component as one super nonlinear component, even though it also contains linear elements.

V.B.2 General Operator Splitting Approach

For the general operator splitting approach, the splitting algorithm is essential to the transient analysis performance. Since the convergence is always guaranteed, the splitting algorithm should focus on the following two issues:

Nonzero Fill-ins The primary goal of splitting algorithm is to reduce the total number of nonzero fill-ins as much as possible. The quality of splitting algorithm directly affects the complexity of LU decomposition. Continued investigation on efficient splitting algorithms is necessary to improve the runtime performance.

Local Truncation Error We have proofed that the general operator splitting approach satisfies the two convergence conditions of numerical integration methods. The operator splitting approach is A-stable and the local truncation error can be controlled by time step size. However, for a given time step size, the local truncation error is also sensitive to the splitting algorithm itself. We have not studied the relation between the partition assignment and the local truncation error yet. Up to now, the objective function of splitting algorithm is mainly the total number of nonzero fill-ins. If the splitting algorithm also aims to reduce the local truncation error, larger time step size can be employed and the number of time points can be reduced, which will improve the transient runtime performance.

Right now, we only partition the circuit into halves. For some circuits with 3D structure such as substrate networks, two-way partitions still generate significant amount of nonzero fill-ins. One possible future direction is to extend the proposed general operator splitting approach from two partitions to multi-partitions and get more freedom on the splitting algorithm to reduce the number of nonzero fill-ins.

Bibliography

[1] E. Acar, F. Dartu and L. T. Pileggi, "TETA: Transistor level Waveform Evaluation for Timing Analysis," IEEE Trans. on Computer-Aided Design, Vol. 21, No. 5, May 2002

[2] W. F. Ames, "Numerical Methods for Partial Differential Equations," 2nd ed. New York Academic Press, 1977

[3] M. Beattie, L. Pileggi, Efficient Inductance Extraction via Windowing, Proc. Design Automation & Test in Europe (DATE) (March 2001)

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick, A Multigrid Tutorial, 2nd Ed. SIAM, 2000.

[5] T. Chen and C. Chen, "Efficient Large-Scale Power Grid Analysis Based on Preconditioned Krylov-Subspace Iterative Methods", IEEE/ACM Design Automation Conference, 2001

[6] L. Chua and P.M. Lin, "Computer-Aided Analysis of Electronic Circuits," Prentice Hall, 1975.

[7] B. David, P. Rajendran, Design and Analysis of Power Distribution Networks, "Design of high-performance microprocessor circuits," Chapter 24, IEEE Press, 2001

[8] C. Desoer and E.S. Kuh, "Basic Circuit Theory," McGraw-Hill, 1969.

[9] A. Devgan, H. Ji, W.Dai, How to Efficiently Capture On-Chip Inductance Effect: Introducing a New Circuit Element K, Proc.

[10] A. Devgan and R. A. Rohrer. Adaptively Controlled Explicit Simulation. IEEE Transactions on Computer-Aided Design of ICs and Systems, vol, CAD-13(6), pp. 746-762, June 1994.

[11] P. Feldmann, R. W. Freund, "reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," *Proceedings of 32th DAC*, pp.376–80, 1995.

[12] Gala, K., et al, On Chip Inductance Modeling and Analysis, DAC, June 2000, pp 63-68 Inductance Models, DAC, June 1999, pp 915-920

[13] A. J. de Geus. SPECS: simulation program for electronic circuits and systems. In Proceddings of the International Symposium on Circuits and Systems, pp. 534-537, pp. 10-16, 1984

[14] G. H. Golub, "Matrix Computations", Johns Hopkins University Press, 1996

[15] He, L., et al, An Efficient Inductance Modeling for Onchip Interconnects, CICC, May 1999, pp 457-460

[16] C. W. Ho, A. Ruehli, and P. A. Brennan, The Modified Nodal Approach to Network Analysis, IEEE Trans. Circuits and Systems

[17] K. J. Kerns, I. L. Wemple, A. T. Yang, "stable and efficient reduction of substrate model networks using congruence transforms," *Proceedings of ICCAD*, pp. 207–14, 1995.

[18] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, Steady-State Methods for Simulating Analog and Microwave Circuits. Norwell, MA: Kluwer, 1990.

[19] J. N. Kozhaya, S. R. Nassif, F. N. Najm, Multigrid-like Technique for Power Grid Analysis. Proc. ICCAD 2001, pp. 480-487.

[20] Y.-M Lee and C.P. Chen. "Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit," ICCAD 75-80, 2001.

[21] Y.-M Lee and C.P. Chen. "The power grid transient simulation in linear time on 3D alternating-direction-implicit," Date 2003

[22] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli. The Waveform relaxation method for the time-domain analysis of large scale integrated circuits. IEEE Transactions on Computer-Aided Design of ICs and Systems, vol. CAD-1(3), pp.131-145, July 1982

[23] Z. Li, C. J. Shi, "SILCA: Fast-Yet-Accurate Time-Domain Simulation of VLSI Circuits with Strong Parasitic Coupling Effects," ICCAD, pp.793-799, 2003

[24] S. Lin, and E. S. Kuh, "transient simulation of lossy interconnects based on the recursive convolution formulation," *IEEE Transactions on CAS I: Fundamental Theory and Applications*, vol.39, pp.879–92, Nov. 1992.

[25] S. Lin, E. S. Kuh, and M. Marek-Sadowska. Stepwise equivalent conductance circuit simulation. IEEE Transactions on Computer-Aided Design of ICs and Systems, vol CAD-12(5), pp672-683, May 1993

[26] K. Mayaram, D. C. Lee, S. Moinian, D. Rich, and J. Roychowdhury, "Computer-Aided Circuit Analysis Tools for RFIC. Simulation: Algorithms, Features, and Limitations," IEEE Trans. CAS II, pp. 274-286, April 2000.

[27] E. Ngoya and R. Larcheveque, Envelop transient analysis: A new method for the transient and steady state analysis of microwave communication circuits and systems, in IEEE MTT Symp. Dig., June 1996, pp. 1365C1368.

[28] L. Nagal, "Spice2: A Computer Program to Simulate Semiconductor Circuits," Tech. Rep. ERL M520, Electronics Research Laboratory Report. UC Berkeley, 1975

[29] T. Namiki and K. Ito, "New FDTD algorithm free from the CFL condition restraint for a 2D-TE wave," IEEE Antennas Propagat. Symp. Dig., pp, 192-195, July 1999.

[30] S. R. Nassif, J. N. Kozhaya, "Fast Power Grid Simulation", Proceeding of DAC, pp.156-61, 2000

[31] A. R. Newton and A. L. Sangiovanni-Vincentelli. Relaxation based electrical simulation. IEEE Transaction on Computer-Aided Design of ICs ans Systems, vol. CAD-3(4), pp.308-330, October 1984

[32] A. Odabasioglu, M.Celik, and L.T. Pileggi, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm," ICCAD, 1997.

[33] J. R. Phillips, L. Daniel, M. Silveira, "guaranteed passive balancing transformations for model order reduction," *Proceedings of 39th DAC*, pp.52–7, 2002.

[34] W. H. Press, S. A. Teukolsky,W. T. Vetterling, "Numerical Recipe in C," 2nd ed. Cambridge University Press, 1992

[35] V. Raghavan, J. E. Bracken, R. A. Rohrer, "AWESpice: A general tool for the accurate and efficient simulation of interconnect problems," *Proceedings of 29th DAC*, pp.87–92, 1992.

[36] K. A. Sakallah and S. W. Director, "SAMSON2: An Event Driven VLSI Circuit Simulator," IEEE Trans. on Computer-Aided Design of ICs and Sytems, vol. 4(4), pp. 668-684, October 1985.

[37] J. E. Storer, "Passive Network Synthesis," *McGraw-Hill, Electrical and Electronic Engineering Series*, New York, 1957.

[38] K. Steuben, "A review of algebraic multigrid" Journal of Computational and Applied Mathematics, vol.128, (no.1-2), Elsevier, 1 March 2001. p.281-309

[39] K. Steuben, "Algebraic Multigrid: An Introduction with Applications", GMD Report No.53, March 1999

[40] W. Sui, "Time-Domain Computer Analysis of Nonlinear Hybrid Systems," CRC Press, 2002.

[41] U. Trottenberg, C. Oosterlee, and A. Schuller, Multigrid, Academic Press, 2001.

[42] R. Telichevesky, K. Kundert, and J. White, Efficient steady-state analysis based on matrix-free Krylov subspace methods, in Proc. IEEE DAC, 1995, pp. 480C484.

[43] H. A. Van Der Vorst, Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Soltion of Non-symmetric Linear Systems, SIAM Journal on Scientific and Statisfical Computing, 13(1992), pp. 631-644

[44] C. Visweswariah and R. A. Rohrer. Piecewise Approximate Circuit Simulation. IEEE Transactions on Computer-Aided Design of ICs and Systems, vol CAD-10(7), pp861-870, July 1991

[45] E. L. Wachspress and G. J. Habetler, "An alternating-direction-implicit iteration technique," J. Soc. Ind. and Appl. Math. 8, 403-424(1960)

[46] H. F. Walker, Implementation of the GMRES method using Householder transformations, SIAM Journal on Scientific Computing, 9(1988), pp. 152-163

[47] J.White and A. L. Sangiovanni-Vincentelli. RELAX2: a new waveform relaxation approach for the analysis of LSI MOS circuits. In Proceedings of the International Symposium on Circuits and Systems (ISCAS), pp. 756-759 vol.2 May 1983

[48] F. Zheng, Z. Chen, J. Zhang, "Toward the development of a three-dimensional unconditionally stable finite-difference time-domain method," IEEE Tran. Microwave Theory and Techniques, vol 48, No. 9, Sep 2000

[49] Z. Zhu, B. Yao, and C.K. Cheng, "Power Network Analysis Using an Adaptive Algebraic Multigrid Approach," DAC, pp. 105-108, 2003

[50] Z. Zhu, M. Borah, K. Rouz, C.K. Cheng, E.S. Kuh, Efficient Transient Simulation for Transistor-Level Analysis, ADP-DAC 2005, pp. 240-243

[51] www.apachedesignsolutions.com/Products/redhawk.htm

[52] www.cadence.com/products/custom_ic/ultrasim/

[53] www.cadence.com/products/dfm/voltagestorm/

[54] www.nassda.com/hsim.html

[55] www.synopsys.com/products/etg/powermill_ds.html

[56] www.synopsys.com/products/mixedsignal/nanosim

[57] www.synopsys.com/products/mixedsignal/timemill_ds.html

[58] www.synopsys.com/products/phy_syn/railmill_ds.html

[59] www.apachedesignsolutions.com/Products/redhawk.htm

[60] www.cadence.com/products/custom_ic/ultrasim/

[61] www.cadence.com/products/dfm/voltagestorm/