# CIRCUIT SIMULATION FOR VLSI AND ELECTRONIC PACKAGING DESIGN

by

Shen Lin

Memorandum No. UCB/ERL M95/80

13 October 1995

# CIRCUIT SIMULATION FOR VLSI AND ELECTRONIC PACKAGING DESIGN

by

Shen Lin

Memorandum No. UCB/ERL M95/80

13 October 1995

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering
University of California, Berkeley
94720

*To my parents and Chiang-Chiang.*

# Acknowledgments

I am most indebted to my research advisor, Prof. Ernest Kuh, for introducing me to an amazing *CAD* world and for being a consistent source of guidance, support, and encouragement. His guidance and help have made my Ph.D. program a smooth and enjoyable one. I also would like to thank him for cultivating my ability to make clear presentations, both oral and written.

Several other professors have contributed graciously their time on my behalf, and I would like to express my gratitude. I would like to thank Professor Robert Brayton and Professor Kobayashi for their careful reading of this dissertation, and Professor Donald Pederson for serving on my qualifying committee.

I would like to express my appreciation particularly to Prof. M. Marek-Sadowska for her support and encouragement. I have benefited from numerous enlightening discussions with her, which kept my work on track.

I gratefully acknowledge crucial contributions to this project from a special friend of mine Jyh-Shing Jang and from other members in Prof. Kuh's group. They are: Narasimha Bhat, Kamal Chaudhary, Michael Jackson, Massoud Pedram, Arvind Srinivasan, Minshine Shih, Ren-Song Tsay, and Deborah Wang. I also want to thank Jaijeet Roychowdhury for helping me prepare several experimental results and giving me important suggestions.

This work was performed while the author was supported by NSF under Grant 88-03711, and SRC under Grant 91-DC-008, and MICRO. To each of these institutions and to the many others that have provided assistance I am grateful and express my thanks.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 The Circuit Simulation Problem.

Computer simulation is used in a variety of different fields to predict the behavior of physical systems whenever it is inappropriate, or too expensive, to build the actual system to observe its behavior. In electrical engineering, circuit simulation is used routinely in the design of integrated circuits (IC) to verify circuit correctness and to obtain detailed timing information before an expensive and time-consuming fabrication process is performed. Moreover, circuit simulation results can be used to guide the circuit optimization process. In fact, circuit simulation is one of the most heavily used computer-aided design (CAD) tools in terms of CPU-time in the IC design cycle. The popularity of this simulation is primarily due to its ability to provide precise electrical waveform information for circuits containing complex devices and all associated parasitics.

Detailed circuit simulation has been used extensively for IC design since the early 1970s. However, the ever-increasing number of devices on a single silicon chip has led to

development of a number of higher-level simulation tools to cope with the complexity of the problem. These tools include behavior simulators, register-transfer-level (RTL) simulators, gate-level logic simulators, and more recently, switch-level simulators [1]. These programs have been used to verify circuit functionality and to obtain first-order timing characteristics. However, there is still a significant gap between a functioning circuit and a circuit which meets all the design specifications - particularly in the case of high-performance custom integrated circuits. In fact, circuit simulation is the only tool which provides enough detailed information to ensure that the simulated circuits will meet specifications over a wide range of operating conditions. Furthermore, the lossy interconnect effects in high-speed Very Large Scale Integrated (VLSI) chips even point out the need for circuit simulation.

At the present time, the most popular circuit simulator tool is the SPICE2 program [2]. There are many thousands of copies of this program in use, as well as a number of versions of "alphabet-SPICE" (e.g. HSPICE, PSPICE, IGSPICE) being marketed commercially. All of these programs offer a wide variety of analyses including DC analysis, time-domain transient analysis, AC analysis, noise analysis and distortion analysis. Of these, the time-domain transient analysis, which solves the time-domain waveforms of each output node voltage and each output branch current, is the most often used and the most computationally expensive in terms of CPU-time. This analysis has always been considered a crucial step in digital VLSI circuit designs. Its results can be used to verify the functionality and the detailed timing performance of simulated circuits and also can be used to guide the circuit optimization process. In this dissertation, we will focus on time-domain transient analysis.

## 1.2 Motivation and Goals.

The SPICE2 program [2] has been the bread and butter of industry and university researchers in circuit design for over two decades. It was originally designed to simulate circuits containing up to hundreds of transistors. However, the continuing improvements of IC technology have made the device feature sizes decreased and the chip sizes increased continuously. With this trend, circuit designers have to design very dense chips. Each chip will have millions of transistors. SPICE2 will be infeasible to simulate circuits of this size. Therefore, the development of fast and accurate simulation methods for VLSI circuits continues to be an important area of research and is one of the goals of this research.

Also, with the trend, it is anticipated that in the near future, the time delay and the speed performance of VLSI systems will be primarily determined by interconnections rather than by device limitations [3]. Therefore, the design of a reliable network for communication and fast computers requires the use of numerical simulation tools that implement models to consider the problems encountered in propagating high-speed signals on lossy interconnects, such as (1) crosstalk, (2) reflections incurred by discontinuity, (3) rise-time slowdown due to dispersion, and (4) dielectric-loss, over a wide frequency range. Fast pulses in excited lines can generate transients via coupling to neighboring lines; these transients can trigger logic gates and other devices, resulting in corrupted data transmission. Circuit designers may also have to ensure that the receiving devices switch on the first incidence of signals. Sometimes it may take the signal voltage several trips back and forth on the transmission lines to turn on the receiving devices, resulting in extra delays, which cannot be estimated by the widely-used first order Elmore delay model.

Improvements in process technology have also made it possible to bond many silicon chips to a common silicon substrate and connect them by thin film interconnects running over the surface of the substrate. With the rapid increase in clock rate and the interconnect lengths of these multi-chip modules (MCMs), electrical length of interconnects can become a significant fraction of the signal wavelength. Consequently, the conventional lumped-impedance interconnect model is no longer adequate. Instead, a distributed transmission line model should be used.

The fundamental difficulty encountered in integrating transmission line simulation into a transient circuit simulator arises because circuits containing nonlinear devices or time-dependent characteristics must be characterized in the time domain while transmission lines with loss, dispersion, or discontinuities are best characterized in the frequency domain. Hence, we need an approach that can handle the two domains at the same time. Currently, there is no generally accepted and efficient approach for the transient simulation of lossy interconnects. This is the second goal of this research.

Therefore, an efficient simulation method for VLSI circuits and an approach for simulating lossy interconnects are the two goals of research in this dissertation.

## 1.3 Contribution.

In order to avoid the Newton-Raphson iterations used in SPICE2's implicit multistep integration of nonlinear circuits, we proposed the Stepwise Equivalent Conductance integration algorithm based on the use of a stepwise equivalent conductance model of a nonlinear resistive device [4, 5]. We proved that a nonlinear circuit can be transformed into

a linear circuit composed of time-varying conductors, and that for the integration of a time-varying circuit within one time step, an effective constant conductance can be determined for each time-varying conductor with a second order of accuracy [1]. The implicit integration of the equivalent linear time-invariant circuit does not require solving any nonlinear equation. This technique, when applicable, is consistent, absolutely stable, and convergent.

When applying the integration algorithm to digital CMOS circuits, we demonstrated that additional speedups in the simulation can be achieved by taking advantage of the fact that voltage waveforms can be modeled to a good approximation as *piecewise-linear* functions. A specific event-driven approach employing this piecewise-linear waveform property is proposed. The StepWise Equivalent Conductance digital CMOS timing simulator, SWEC, has been implemented based on the above proposed techniques. Comparisons have been made with Relax2.3[6], iSPLICE3.0[7], XPsim[8], and SPECS2[9] on a large number of circuit examples. The results indicate that SWEC, while accurate, exhibits far better efficiency [10].

Moreover, a new approach for transient simulation of lossy interconnects terminated in arbitrary nonlinear elements is also proposed. The approach is based on convolution simulation. By using the Pade approximations of each line's characteristics or of each multiconductor lines' modal functions, we derive a recursive convolution formulation, which greatly reduces the computation used to perform convolutions. The approach can handle frequency-varying effects, such as skin effects, and general coupling situations. A large circuit can be decomposed into subcircuits by making a cut on every line and the integration

[1]The local truncation error for integration is of the cubic order of the time step used.

of each subcircuit can be performed independently. Furthermore, we analyzed the errors introduced by Pade approximations and developed a scheme to determine the necessary order for an approximation. We have incorporated the proposed technique in SWEC. The comparisons with SPICE3.e [11] indicate that SWEC can be one to two orders-of-magnitude faster.

In summary, we will present the following three topics in this dissertation:

1. the Stepwise Equivalent Conductance implicit integration technique,
2. the Piecewise-Linear Waveform event-driven simulation
3. the lossy interconnect simulation based on the Recursive Convolution Formulation

## 1.4 Organization

Chapter 2 gives the background of transient circuit simulation. The chapter also reviews the previous work addressing the issues of time efficiency vs. accuracy by simplifying the numerical algorithms or by using simpler device models and the previous work on the lossy interconnect simulation. After indicating the strong and weak points of the approaches, we explain how a simulation approach will benefit from the strong points, which leads to the pursuit of this research.

Chapter 3 presents the Stepwise Equivalent Conductance implicit integration technique. We will give a proof for the correctness of the transformation from a nonlinear circuit to a linear time-varying circuit. We will determine the effective constant conductance of the time-varying conductors for the integration during a time step and then discuss the

choice of time steps for integration. The errors introduced by the transformations will be analyzed.

In chapter 4, the StepWise Equivalent Conductance digital CMOS circuit simulator SWEC will be presented. We will introduce SWEC's circuit partition technique, the piecewise-linear waveform approximation, and the event driven approach that employs digital CMOS circuits' piecewise-linear waveform property. Experimental results of SWEC along with comparisons with SPICE, iSPLICE3.0, Relax2.3, XPsim, and SPECS2 programs will be shown.

In chapter 5, we present our lossy interconnect simulation algorithm. The algorithm is based on two techniques: Pade approximation and Recursive Convolution Formulation. These techniques will be introduced. We will analyze the errors introduced by Pade approximations and develop a scheme to determine the necessary order for an approximation. The comparisons of our approach with SPICE3.e [11] will be shown.

Chapter 6 suggests possible future work. The transient simulation of lossy interconnects modeled by scattering parameters will be presented. The scattering parameter model is the key to handle very general coupling geometry and non-uniform lines.

Finally, in chapter 7, conclusions of this research are presented.

# Chapter 2

# Background and Review of Previous Work.

## 2.1 The Direct Approach of SPICE2.

### 2.1.1 Linear Multistep Numerical Integration.

The KCL nodal equations for the simulated circuit will be of the form

$$\mathcal{F}(\mathbf{V}(t)) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I_s}(t), \tag{2.1}$$

where $\mathbf{V}(t)$ is the node voltage vector, $\mathcal{F}(\cdot)$ is a vector function of $\mathbf{V}(t)$ with its $i$-th entry representing the total current flowing out of node $i$ through resistive devices, $\boldsymbol{C}$ is the constant capacitance matrix, and $\mathbf{I_s}(t)$ is the vector for inputs (represented as current sources). Independent voltage sources can be considered by the Norton equivalent models and be represented as current sources. The Modified nodal analysis can be used to consider circuits with linear inductors, nonlinear capacitors, and nonlinear inductors. The circuit

equation will still be a first order differential equation as that of Eq.(2.1). For that case, the variable $\mathbf{V}(t)$ will include the inductor branch currents or the charges of nonlinear capacitors, and the matrix $\boldsymbol{C}$ will be a combination of the inductance matrix and the capacitance matrix.

The transient circuit simulation amounts to the numerical integration of the initial-value problem specified by Eq.(2.1). The initial value (or $\mathbf{V}(0)$) is the DC solution of the circuit. SPICE2 uses the linear multi-step (LMS) method to perform the integration step by step. It assumes the following equation will hold

$$\mathbf{V}(t_{n+1}) = \sum_{i=0}^{N} \alpha_i \mathbf{V}(t_n - ih) + h \sum_{i=-1}^{N} \beta_i \dot{\mathbf{V}}(t_n - ih), \tag{2.2}$$

where $h$ is the time step. If $\mathbf{V}(t)$ is a $k$-th degree polynomial of the time $t$ then the $\alpha$ and the $\beta$ of Eq.(2.2) can be determined to give the exact solution of $\mathbf{V}(t_{n+1})$ and we say the method is a $k$-th order method. The $\dot{\mathbf{V}}(t)$ of Eq.(2.1) is replaced by Eq.(2.2); therefore, from the values of $\mathbf{V}(t)$ and $\dot{\mathbf{V}}(t)$ at previous time points, $\mathbf{V}(t_{n+1})$ can be solved. The whole simulation is given by a series of thses steps.

If the $\beta_{-1}$ of Eq.(2.2) is not equal to zero, then the integration method is called an *implicit* integration method because the unknown variable $\mathbf{V}(t_{n+1})$ appears in the function $\mathcal{F}(\cdot)$ term and $\boldsymbol{C}\dot{\mathbf{V}}(.)$ term. If $\mathcal{F}$ is nonlinear, then the implicit integration for each time step involves solving a system of nonlinear equations. Computationally expensive Newton Raphson iterations are generally needed to find the solutions. For each Newton iteration, a Jacobian matrix, which involves evaluating device models, needs to be determined and a linear system of equations needs to be solved.

On the other hand, if $\beta_{-1}$ is zero, it is called the *explicit* integration approach; then

solving $\mathbf{V}(t_{n+1})$ at most takes one inversion of the $\boldsymbol{C}$ matrix because the unknown variable $\mathbf{V}(t_{n+1})$ appears only in the $\boldsymbol{C}\dot{\mathbf{V}}(.)$ term. However, the explicit method has poor *stability*: the numerical errors will explode if larger time steps are used. Therefore, the explicit method can not handle *stiff* circuits. Stability and circuit's stiffness will be explained in the next subsection. Because of this stability consideration, SPICE2 employs the computationally expensive implicit integration approach. SPICE2 has been proved to be very reliable and stable; however, it becomes impractical to simulate large circuits.

### 2.1.2 Properties of Numerical Integration.

In this subsection, we are going to introduce the issues to be considered for numerical integration methods. These properties are essential to guarantee the correctness of the integration results and are used as the guide lines to differentiate a better integration method from another. When we present our new integration approach in chapter 3, we will also examine these properties of our approach.

#### Existence and Uniqueness of Solutions.

Most numerical methods for solving Eq.(2.1) subject to the initial condition $\mathbf{V}(0) = \mathbf{V_0}$ assume the solution to this initial-value problem exists and is well defined for all times $t \geq 0$. In the event that the solution does not exist or is pathological, most numerical methods would still produce a set of numbers, which of course is meaningless. Therefore, in order for us to have confidence on the simulation results, it is important to derive some criteria on the simulated circuits to guarantee the existence and the uniqueness of a solution in the general case.

The **Peano existence theorem** shows that the *continuity* of $\dot{\mathbf{V}}(t)$ (or $\boldsymbol{C}^{-1}[\,\mathbf{I}_s(t) - \boldsymbol{\mathcal{F}}(\mathbf{V}(t))\,]$) is a sufficient condition to guarantee the existence of a solution around $t = 0$ [12]. Clearly, if $\boldsymbol{C}$ is singular or $\boldsymbol{\mathcal{F}}(\cdot)$ is not continuous for some intervals of interest, the solution may not exist.

The Peano's theorem is a local theorem, since it only proves the existence of a solution over some nonzero time interval centered about the initial time. To guarantee the existence of a solution over the whole simulation time, much more severe conditions must be imposed on $\boldsymbol{C}^{-1}[\,\mathbf{I}_s(t) - \boldsymbol{\mathcal{F}}(\mathbf{V}(t))\,]$, as shown by **Wintner's Global Existence Theorem** below

**Wintner's Global Existence Theorem:** *If there exists a piecewise-continuous function $L(t)$ defined for the $t$ over the whole simulation time such that*

$$\| \boldsymbol{C}^{-1}(\boldsymbol{\mathcal{F}}(\mathbf{V}') - \boldsymbol{\mathcal{F}}(\mathbf{V}")) \| \leq L(t) \| \mathbf{V}' - \mathbf{V}" \| \tag{2.3}$$

*for all $t$ in the simulation time (this is called a global Lipschitz condition), then Eq.(2.1) has a unique solution.* [13].

Wintner's global existence theorem proves both the existence and the uniqueness of a solution. However, the conditions required by the theorem are rather strong. Roughly speaking, these conditions prevent the functions $\boldsymbol{\mathcal{F}}(\mathbf{V}(t))$ and $\mathbf{V}(t) \quad \forall t$ from growing too rapidly. As a result, the simulated circuit cannot have fast switching inputs and all its devices should be modeled by smooth i-v curves. It must be emphasized that the above two theorems provide only sufficient conditions. Hence, even if the circuit does not satisfy Wintner's global existence theorem, the solution may still exist.

**Local Truncation Errors and Consistence.**

The error introduced by one integration step is called the *local truncation error* (LTE). As the time step $h$ gets smaller, the LTE will vary as a degree of $h$. If the linear multi-step method is of the $k$-th order, then its LTE will be of $O(h^{k+1})$, which means that the method preserves the $k$-th order of accuracy. Given an error bound on the LTE, the desired time step $h$ can be determined. If the integration method has a larger $k$, then we are allowed to use a larger time step $h$ to meet the error criterion and get better efficiency.

A numerical integration method is said to be *consistent* if as $h$ approaches zero, the ratio $\frac{LTE}{h}$ will vanish, where $LTE$ is the method's local truncation errors. For a nonconsistence method, no matter how small the integration time steps are used, nonzero global errors will always exist. Therefore, consistence is a desired property for the integration method.

**Stability and Convergence.**

To keep the accuracy requirement, we have to use small time steps. This means for a simulation, many steps of solving the difference equations introduced by numerical integrations will be needed. Thus, as a practical matter, it is important that these solutions should not be too sensitive to small errors in the computations (for example, roundoff errors). This sensitivity to errors is related to what is called the *stability* of the integration method. Numerical errors will be introduced by each integration step. These errors will die out, as simulation proceeds, for a stable integration method. However, for an unstable method, previous errors will magnify later errors; eventually, the global errors will blow up

to infinity.

Basically, the solution of each integration is a linear combination of exponential terms. The exponents of those terms are function of the time step $h$, the numerical method used, and the poles of the simulated circuit. To obtain stable simulation results, the exponents should be on the left half of the complex plane. We consider the case where the circuit has only left-half-plane poles. A numerical integration method is said to be *stable* if there exists a nonzero interval of time steps around $h = 0$ such that the exponents can be situated on the left half of the complex plane. To be more specific, to verify the stability of a method, let us consider the test equation

$$\dot{x} = \lambda x, \qquad x(0) = 1, \tag{2.4}$$

where $\lambda$ is a left-half-plane complex constant and represents the pole of the system modeled by Eq.(2.4). The region of the product $\lambda h$ that lets the exponents be situated on the left half of the complex plane is called the *region of stability* for the numerical method. If the region of stability includes the origin, then the method is stable.

The stability of a multistep method only guarantees that the local truncation and round-off errors are not amplified and remain bounded for a sufficiently small step size. It is important for us to derive a criterion to guarantee that the global error will tend to zero as $h \to 0$, which is defined as the *convergence*. It has been proved that every stable and consistent multistep method is convergent [14]. Conversely, every convergent multistep method is stable.

**Stiffness and Absolute Stability.**

For the sake of efficiency, the time step selection for integration will be based on the activity of the simulated circuit. We want to use large time steps for the situation when the circuit's node voltages do not change considerably. However, the stability property may not hold for some integration methods because the large time steps used cause the $\lambda h$ values to be outside the method's region of stability.

It is possible that a circuit may have several poles ($\lambda$) and some of them differ in many orders of magnitude. These circuits are called *stiff*. A simple example of stiffness is the case of a fast initial "transient" in the solution, which dies quickly, followed by a slower "steady-state" solution. To handle this type of behavior, it is natural to use small time steps in the transient portion to accurately follow the solution and then to increase the step size for the remainder of the solution. However, the strategy may lead to instability for the integration method, especially for explicit integration methods. For example, let us consider a circuit with two poles $\lambda_1 = -1.0$ and $\lambda_2 = -1.0e6$. The transient associated with $\lambda_2$ will approach its steady state very fast. After that, the circuit will have only minimum activity because $\lambda_1$ is small. In this situation, we want to increase the time step $h$. However, since $\lambda_2$ is very large, any increase on $h$ will make $\lambda_2 h$ go beyond an integration method's region of stability. Since the efficiency is one of the major concerns when we choose an integration method, we would like to have its region of stability cover as large the left half of the complex plane as possible. A numerical integration method which has its region of stability cover the whole left half of complex plane is said to have the *absolute stability*.

It has been proved by Dahlquist [13] that no multistep method that exceeds order 2

has absolute stability. Moreover, no explicit integration method can have absolute stability. Since the absolute stability is rather strong, for practical purpose, we relax the region of stability by a small strip left of the imaginary axis except origin. This kind of stability is called *stiff stability.* Because most circuits do not have poles with large imaginary parts, the stiff stability criterion can be adequate. However, again no explicit method has the stiff stability. Due to the stability consideration, SPICE2 uses stiffly stable and absolutely stable integration methods. Hence, when integrating nonlinear circuits, solving a system of nonlinear equations cannot be avoided.

However, compared with explicit algorithms, the extra computation does not really improve the solution's accuracy. One needs to go through the pain of Newton Raphson iterations in the implicit algorithms mainly to achieve **stability**. This makes one wonder if there is really no easy way of obtaining stability while retaining accuracy, and that is the motivation of our research on developing a new numerical integration approach.

## 2.2 Work on Improving Efficiency.

The overall goal in circuit simulation is to generate the solution as efficiently as possible while providing the desired level of accuracy. To improve the efficiency of circuit simulations, two types of approaches have been proposed previously. One is to simplify the numerical algorithms and the other is to simplify the device models. In this section, we will survey the work in these two categories. The pros and cons of each type of approach will be shown.

### 2.2.1 Simplify the Numerical Algorithms.

Previously, the direct approach of SPICE2 was modified to avoid the large number of Newton iterations, to maximize time steps used, or to exploit circuit's latency and multirate behavior. These approaches include (1) the Relaxation Approach, (2) the Waveform Relaxation Approach, (3) the Semi-Implicit Integration Approach, (4) the Exponential Integration, and (5) the Asymptotic Waveform Evaluation Method. These are described in the following subsubsections.

**Relaxation Approach:**

A large circuit is usually very sparse: every node is only connected to a small number of neighbor nodes. For the direct approach of SPICE2, all the node voltages are solved simultaneously. Even though a sparse linear equation solver is employed, the complexity is still superlinear. Once the number of nodes are large, the approach will become very computation-intensive. The relaxation approach [15] tries to exploit the circuit's sparsity to speed up the solving process. The relaxation approach, instead of solving all the system variables at once, only solves one of them each time by assuming the rest are correct solutions. Then, it iterates on this step over all variables until the solution of each variable converges. The approach can be used to solve the linear system of equations within each Newton Raphson loop or directly applied to solve the system of nonlinear equations.

The efficiency of the relaxation approach is determined by the speed of convergence, which heavily depends on the coupling between nodes. When simulating tightly coupled circuits (e.g. circuits with strong negative feedbacks), it takes a lot of computation

for the solutions to converge. If the circuit simulation program is intended for the simulation of MOS digital circuits, then it is possible to make use of the weak coupling between the gate node of a MOS transistor and the drain node (or the source node) to achieve faster convergence. Examples of relaxation-based simulators are the family of SPLICE [7] [16] [17].

**Waveform Relaxation Approach:**

The circuit simulators employing the direct method use a single common time step to integrate the whole circuit; the step size is, therefore, determined by the activity of the fastest changing part of the circuit. However, for a large circuit at one time point, it is common that only a small portion of the circuit has activity and the rest does not (being latent). As a result, many variables are solved using time steps which are much smaller than necessary to compute their solutions accurately. Waveform relaxation simulators try to improve the efficiency by exploiting the circuit's *latency*, which refers to the situation when most node voltages stay the same from one integration to another.

The waveform relaxation approach [18, 19] solves a node voltage waveform for the whole simulation time by assuming the waveforms of its neighboring nodes are as pre-solved. Then, it iterates on this process until the waveform of each node converges. When solving the waveform at one node, it can use variable time steps to achieve the maximum efficiency because the waveforms on neighboring nodes are known. The reduction in run time is accomplished by computing fewer solution points for each waveform, thereby reducing the total number of model evaluations, and by avoiding the direct sparse-matrix solution. However, the tradeoff exists only for loosely coupled nodes; otherwise, the approach takes

a long time to converge. Circuit partitions become very important. Strongly connected nodes should be grouped into one solving process. Furthermore, the ordering of the solving processes is also important. Take an inverter chain as an example. The first few iterations will be obsolete if the ordering starts on the last inverter and goes backward. However, there will be no wastage of computation if the ordering starts on the first inverter and goes forward.

The approach is very suitable for simulating digital MOS circuits because of the weak coupling between the gate node of a MOS transistor and the drain node (or the source node). Examples of waveform-relaxation based simulators are Relax2.1 [6] and IDSIM2 [20] (the Partial Waveform Convergence [21]).

**Semi-Implicit Integration Approach:**

To avoid being trapped in the lengthy iteration process, the semi-implicit integration method has been proposed and applied in the simulators of MOTIS [22] and Event-EMU [23]. They conjectured that there exists a small enough time step to obtain the solution in exactly one Newton iteration. The nonlinear devices of a circuit are linearized using the node voltages at the previous time point and then the linearized circuit is integrated. However, to maintain the desired accuracy in most cases very small time steps need to be used, and that unfortunately degrades the efficiency.

**Exponential Integration Approach:**

The solution of an initial-value linear differential equation is a linear combination of exponential functions. Therefore, if the solution $\mathbf{V}(t)$ of Eq.(2.1) between two time

points is approximated by an exponential function of $t$ instead of a polynomial employed by the linear multistep method then it is possible to use larger integration time steps, hence improving the efficiency. Based on this observation, the exponential integration approach was proposed in [8] and implemented in the MOS simulator XPsim. The voltage waveforms produced by the approach are piecewise exponentials. The node voltage at a new time point $t_{n+1}$ is equal to its previous value $v(t_n)$ multiplied by $e^{\lambda h}$, where $\lambda$ can be determined by matching the circuit's asymptotic solution at $t_n$. The authors of [8] have proved that for a first order exponential integration method, its local truncation error LTE at $t_n$ is equal to $v(t_n)\frac{h^2}{2}\frac{d\lambda}{dt}$. On the other hand, the LTE of a similar first order linear multistep method is equal to $\ddot{v}(t_n)\frac{h^2}{2}$. Since $v(t_n)\frac{d\lambda}{dt}$ will not change as fast as $\ddot{v}(t_n)$, larger time steps can be used in the exponential approach.

However, the approach is neither absolutely stable nor stiffly stable. The efficiency will be impaired when simulating stiff circuits.

**Asymptotic Waveform Evaluation Method:**

For the SPICE direct approach, when simulating linear RLC networks, even though the waveform at the network's internal nodes are not important, the approach still needs to solve all of them at each time point because the waveforms are the internal states of the RLC network. As the network size increases, there will be too many internal nodes and the computations to perform each integration will be infeasibly large. However, linear system theory tells us that if we treat the whole RLC network as a linear system and the external nodes as the terminals of the system then the convolution of the inputs and the impulse response of the system can be used to determine the waveforms at the system's output

nodes. Then, we don't need to solve the waveforms at the internal nodes and can save a lot of computation. This idea motivates the asymptotic waveform evaluation method of [24].

The timing analyzer AWEsim [25] was implemented based on the asymptotic waveform evaluation method. AWEsim assumes that step inputs are applied at the input nodes of the RLC network. Therefore, the transfer function of the output (or outputs) of the network can be determined. The transfer function is then expanded into a Maclaurin series of $s$, the Laplace transform variable, around $s = 0$. The first order Elmore delay of the output will be the first moment of the series. An approximated time domain output waveform can also be determined by matching the truncated transfer function. This approach can outperform SPICE by two to three orders of magnitude in speed. However, its efficiency will be impaired if nonlinear terminals exist in the circuit. The authors of [24] suggest using effective linear resistors to represent those nonlinear terminals; however, the accuracy degrades. Also, the approach suffers from the stability problem when the poles of the approximated transfer function are located on the right half of the complex plane. This becomes a phenomenon if the network is not very lossy.

### 2.2.2 Simplify the Device Models.

The approaches of approximating the i-v characteristics of nonlinear devices by piecewise-linear curves or stepwise-constant curves to avoid solving nonlinear equations have been proposed. The increase in efficiency is due to

1. avoiding solving a system of linear equations in each Newton Raphason iteration,

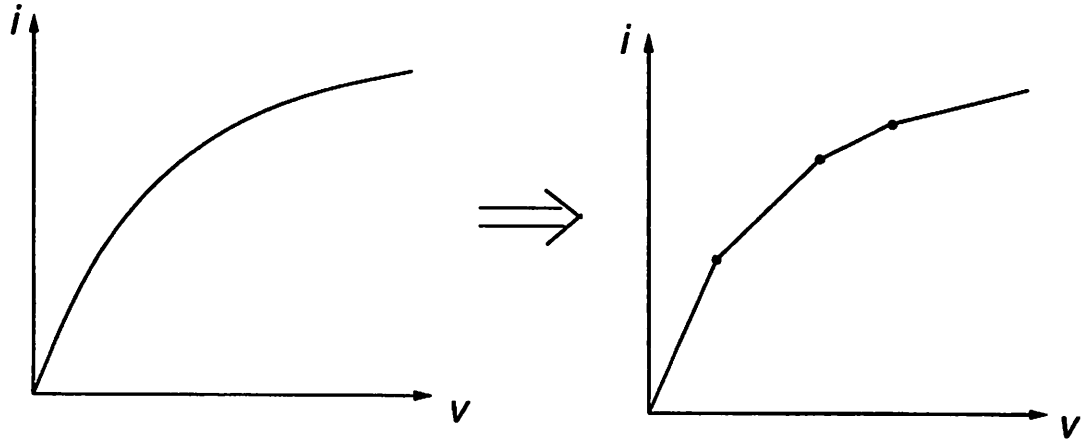


Figure 2.1: The Piecewise Linear Approximation of Nonlinear i-v Characteristic.

2. avoiding the model evaluation of nonlinear devices, which involves many floating point calculations.

These are described in this subsection.

**Effective Linear Conductance Model:**

The effective linear conductance model has been used in the timing analyzers Crystal [26], TV [27], and Rsim [28]. For the whole transition, these timing analyzers replace every MOS transistor by an effective conductor and use the RC-tree [29] approach to estimate first order timing information of the analyzed MOS circuit. Analog waveform information cannot be obtained from this type of analysis, which means that these methods cannot ensure that a circuit meets specifications. Furthermore, there is no mathematically rigorous way of determining the effective conductance. For the simulation that demands high accuracy, this type of approach is not adequate.

**Piecewise-Linear Device Model:**

The application of the Newton-Raphson algorithm to solve the nonlinear equations introduced in each integration step would require the evaluation of a Jacobian matrix at each iteration. This is usually a time-consuming procedure. It is possible to avoid evaluating the Jacobian if we approximate the i-v curve of each nonlinear device by piecewise-linear segments as shown in Fig. 2.1. The Katzenelson algorithm has been proposed to perform this piecewise-linear version of numerical integration [30]. If the i-v curve of each nonlinear device is approximated by only one linear segment, then each integration step of Eq.(2.1) will be brought down to solve just a system of linear equations

$$\boldsymbol{A}\mathbf{V} = \mathbf{b}, \tag{2.5}$$

where the solution $\mathbf{V}$ is the node voltage at the new time point. It is because the approximated circuit is linear. For the situation where the approximation of each nonlinear device has more than one piecewise-linear segment, the space of $\mathbf{V}$ can be partitioned into hypercubes such that inside each hypercube every nonlinear characteristic is approximated by one linear segment. The integration step will then require solving a series of linear equations below

$$\boldsymbol{A}^i\mathbf{V}^i = \mathbf{b}^i \quad i = 0, 1, ..., n. \tag{2.6}$$

The index $i$ represents the $i$-th hypercube, which contains the solution $\mathbf{V}^i$. The matrix $\boldsymbol{A}^i$ and the column vector $\mathbf{b}^i$ depend on the linear segment of each device specified in the hypercube $i$. $\mathbf{V}^0$ is equal to the $\mathbf{V}$ at the previous time point and the final $\mathbf{V}^n$ is the solution of $\mathbf{V}$ at the new time point. Based on the Katzenelson algorithm, the trajectory

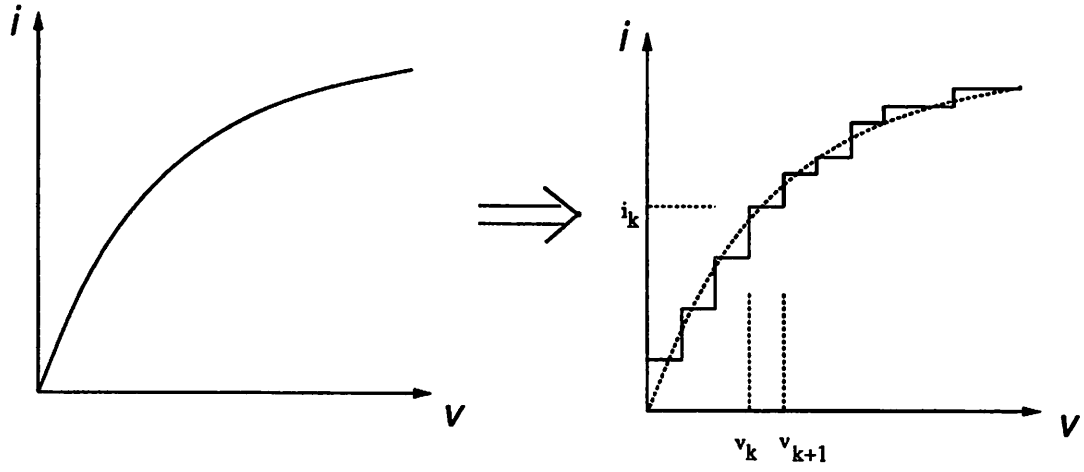


Figure 2.2: The Stepwise Constant Approximation of Nonlinear i-v Characteristic.

from $\mathbf{V}^0$ to $\mathbf{V}^n$ can be determined and the $\boldsymbol{A}^i$ and the $\mathbf{b}^i$ when crossing the boundary of two hypercubes can be updated. The piecewise-linear simulator PLATO was implemented based on the above approach [31].

This piecewise-linear integration will lose its efficiency if for each integration the trajectory needs to go through many hypercubes. Since the system of linear equations given in Eq.(2.6) needs to be solved for each hypercube, the computation will be comparable with the Newton-Raphson algorithm. Another weakness of this type of approach is that it is no longer consistent because the local truncation error will not vanish even when a very small integration time step is used. Its accuracy has been restricted by the piecewise-linear approximation of the i-v curve.

**Stepwise Constant Device Model:**

If the simulated circuit does not not have floating capacitors[1] or inductors then it is possible to perform the simulation without solving any system of equations by approximating the i-v characteristic of each device by a stepwise constant function as shown in Fig. 2.2. In other words, the i-v characteristics are represented by the constant current level $i_k$ for each voltage range $v_k \leq v < v_{k+1}$, for $k = 0, 1, ..., n$. The approximation can be made with a small number of segments (less accurately but more efficiently) or with a large number of segments (vice versa).

Devices with stepwise constant i-v characteristics have stepwise constant branch currents in time. These currents combine via KCL to force stepwise constant currents flowing into all the grounded capacitors. Therefore, by KVL all the node voltages (or the branch voltages) will be piecewise linear in time. Since the branch voltage of each device is piecewise linear, we can predict the time when the branch current will change from one level to another. The whole simulation is driven by these predicted event times. One event is associated with a device. Events are stored in a priority queue. The event at the top of the queue will be deleted and processed. Each event process has two steps: (1) update the branch current of the device whose branch voltage crosses different segments of its stepwise constant i-v curve, and (2) predict the next event time for the device and insert this event back to the queue.

This approach was proposed in [9] and implemented as a circuit simulator SPECS2. Since no system of linear or nonlinear equations need to be solved, the approach is efficient.

[1] A floating capacitor is a capacitor with neither terminal connected to the ground node.

The approach gives the flexibility to vary the tradeoff between accuracy and efficiency on a branch-by-branch basis. However, the approach can only be applied to a very small class of circuits because no floating capacitors and inductors are allowed in the simulated circuits. Furthermore, the accuracy is restricted by the stepwise constant approximation. From our observation, its efficiency degrades very quickly if more voltage segments are used on the device approximation to keep the desired accuracy.

**Electrical Logic Model:**

The application of the explicit forward Euler method to integrate Eq.(2.1), say from $t_n$ to $t_{n+1}$, yields

$$\mathcal{F}(\mathbf{V}(t_n)) + \boldsymbol{C}\frac{\mathbf{V}(t_{n+1}) - \mathbf{V}(t_n)}{h_n} = \mathbf{I_s}(t_n), \tag{2.7}$$

or,

$$\mathbf{V}(t_{n+1}) = \mathbf{V}(t_n) + h_n\boldsymbol{C}^{-1}[\mathbf{I_s}(t_n) - \mathcal{F}(\mathbf{V}(t_n))], \tag{2.8}$$

where $h_n = t_{n+1} - t_n$.

When there are no floating capacitors in the circuit, the capacitance matrix $\boldsymbol{C}$ and its inverse $\boldsymbol{C}^{-1}$ are diagonal matrices. Therefore, Eq.(2.8) turns out to be:

$$v_j(t_{n+1}) = v_j(t_n) + h_n C_{j,j}^{-1}[is_j(t_n) - f_j(\mathbf{V}(t_n))], \tag{2.9}$$

where $v_j(t_n)$ is the voltage at node $j$ at $t_n$, $C_{j,j}$ is the grounded capacitance at node $j$, $f_j(\mathbf{V}(t_n))$ is the current leaving node $j$ at $t_n$ through devices, and $is_j(t_n)$ is the current flowing into node $j$ at $t_n$ from power supplies.

From Eq.(2.9), we can compute the time for node $j$ to make a transition from $S_{Now}$ to $S_{Next}$, as follows:

$$h_n = \frac{C_{j,j}(S_{Next} - S_{Now})}{-(is_j(t_n) - f_j(\mathbf{V}(t_n)))}. \tag{2.10}$$

It is easy to compute Eq.(2.10) because every term is scalar. The electrical logic model simulation [32] introduced the idea of dividing the voltage range into discrete states, e.g. 0V, 0.5V, 1.0V,... The $S_{Now}$ and $S_{Next}$ in Eq.(2.10) are examples for that. The whole simulation is driven by the event times, the time points when a node voltage changes from one state to the next adjacent state. The event process has three steps:

1. Update the voltage at node $j$ to the next state.
2. Evaluate the $f_j(\mathbf{V})$ using the new $v_j$.
3. Compute the transition time for node $j$ using Eq.(2.10) and schedule node $j$ again to the time equal to the current time plus this transition time. Node $j$ will be processed again at that time.

The electrical logic model carries many similarities to the stepwise constant model introduced in the previous subsubsection. Both treat a device as a stepwise current source, both use the event processes to drive the simulation, and both are weak in handling floating capacitors and inductors. One difference is that the events for the stepwise constant model are associated with devices, while the events for the electrical logic model are associated with nodes. Furthermore, for the stepwise constant model, the voltage segment for a device to keep a current level is determined before running the simulation but that for the electrical logic model is determined during the simulation. Because of this difference, the electrical

logic model should preserve better efficiency. Both of them, however, experience problems with accuracy and sometimes stability because of their explicit integration approach.

## 2.3 Previous Lossy Interconnect Simulation Work.

The fundamental difficulty encountered in integrating transmission line simulation into a transient circuit simulator arises because circuits containing nonlinear devices or time-dependent characteristics must be characterized in the time domain while transmission lines with loss, dispersion, or discontinuities are best characterized in the frequency domain. To cope with this difficulty, four types of approaches have been proposed in the literature. One uses a network of lumped elements and segments of ideal transmission line to approximate the frequency response of each lossy transmission line or each lossy multiconductor system [33]. The approximated circuit models are suitable for existing general-purpose circuit simulators. However, the drawback of this type of approach is that the amount of computation increases for the simulation because a large number of extra nodes and elements are introduced [11]. The second type of approach adopts the convolution technique. For each integration, the outputs of linear lossy multiconductor lines are the convolutions of the inputs with the impulse responses (Green's function) of the multiconductor lines. The multiconductor lines are treated as a *linear N-Port system.* The difficulty of this type of approach lies in how to determine the impulse responses of an arbitrary multiconductor line system. People used the inverse Fast Fourier transformation technique [34], the numerical inverse Laplace transformation technique [35], inverse Fourier transformation of frequency-domain scattering parameters [36] [37], and even the explicit analytical approach[11] to

determine the impulse response. However, the convolution simulations suffer from a common drawback: the convolution operation needs to extend over the entire past history. The computation time required at any time point $t$ is then proportional to $t$; therefore, the convolutions at large time points will be very time-consuming. Furthermore, the inverse Fourier transformations of [34], [36], and [37] will suffer from either the aliasing effects or that too many frequency points are needed for the transformations to avoid aliasing.

To avoid the time-consuming convolution integrations, the state-based approach [38] and the waveform relaxation based approach [33][39] have been proposed. The state-based approach utilizes information about the internal states of a transmission line at a given time to solve the states for the next time point. The voltage and the current at the sample points are kept as the states of the line. The voltages and the currents are assumed to be piecewise-linear between adjacent sample points. Based on this assumption, the state variables can be determined by using integrations on space, hence avoiding convolutions. However, the efficiency of the approach will degrade for the simulation with many sharp edges in the line's waveforms. The samples will be chosen densely in the regions where waveforms are fast-varying. Typically, for the simulation containing a pulse with the rise time of 100 pico seconds and having the voltage resolution of 0.5 Volts, around 20 sample points per inch of the line are required [2]. If the rise time is smaller or the voltage resolution is more accurate than 0.5 Volts, we need even more sample points, which will increase the computation used to perform the simulation. The waveform relaxation based approach

[2]The signal travels at the speed of 5e-3 inch per pico-second, or in other words 200 pico seconds per inch. A full ramping-up, from 0 to 5 Volts, takes 100 pico seconds; therefore, with the 0.5 Volt resolution, we need 0.1 samples per pico-second. The number of required samples per inch is equal to 200 pico seconds per inch times 0.1 samples per pico-second, which is 20 samples.

solves the line's equations in the frequency domain and uses the FFT to transform the results back and forth between the two domains for each iteration. Hence, time domain convolutions are avoided by performing multiplications in the frequency domain. Again, this type of approach is not suitable for handling fast-varying signals. For the simulation containing a 100-pico-second-rise-time pulse and having the simulation time of 100 nano seconds, each FFT needs to process around one hundred thousands data points in order to avoid the aliasing effects. In summary, we think in order to solve the simulation involving fast-varying signals we need to focus our efforts on developing fast convolution integration algorithms.

# Chapter 3

# Stepwise Equivalent Conductance Circuit Simulation.

In this chapter, we present our Stepwise Equivalent Conductance circuit simulation method, which treats every nonlinear resistance device as a 2-terminal linear time-varying conductor. We show that implicit integration can be efficiently applied to this type of circuit under a given error criterion. No nonlinear equations need to be solved. The technique is proved to be consistent, absolutely stable, and convergent. Furthermore, we demonstrate that a second order of accuracy is achieved by solving a system of linear equations for each integration step.

The Stepwise Equivalent Conductance circuit simulation makes the following two assumptions regarding the simulated circuits:

- Every node in the circuit has nonzero capacitance to ground. In fact, this assumption does not place any restriction on the simulated circuit because practically every node

in a real circuit has nonzero parasitic capacitance to ground.

- Every nonlinear device in the circuit has a unique current path. Examples of these kinds of devices are MOS, JFET, and diodes.

The chapter is organized as follows. In section 3.1, we introduce the transformation from a nonlinear circuit to an equivalent circuit composed of 2-terminal linear time-varying conductors and discuss the exactness of the solution of the linear time-varying circuit under the "smooth" i-v characteristic assumption. In section 3.2, an accurate and efficient algorithm to integrate the linear time-varying circuit is introduced. The accuracy and the convergence of the algorithm are analyzed. In section 3.3, we discuss the choice of time steps for integration.

## 3.1 Equivalent Linear Time-Varying Circuit Transformation.

### 3.1.1 Circuit Equations.

For the sake of simplicity, we start with the assumption that there are no inductors in the simulated circuit and that all the capacitors are constant. We extend the discussion to circuits with linear inductors, nonlinear capacitors, and nonlinear inductors in Appendix A by using modified nodal analysis. The KCL nodal equations for the simulated circuit will be of the form

$$\mathcal{F}(\mathbf{V}(t)) + C\dot{\mathbf{V}}(t) = \mathbf{I_s}(t), \tag{3.1}$$

where $\mathbf{V}(t)$ is the node voltage vector, $\mathcal{F}(\cdot)$ is a vector function of $\mathbf{V}(t)$ with its $i$-th entry representing the total current flowing out of node $i$ through resistive devices, $\boldsymbol{C}$ is the constant capacitance matrix, and $\mathbf{I_s}(t)$ is the vector of inputs (represented as current sources). Independent voltage sources can be considered by the modified nodal analysis as well. Since every node is assumed to have nonzero grounded capacitance, $\boldsymbol{C}$ is diagonally dominant. If $\mathcal{F}$ is nonlinear, then the implicit integration of Eq.(3.1) for each time step involves solving a system of nonlinear equations. Computationally expensive Newton Raphson iterations are generally needed to find the solutions.

The unique current path assumption of nonlinear devices implies that the simulated circuit can be treated as an equivalent circuit with 2-terminal resistive elements only. To be more specific, the i-v characteristic of every nonlinear device at each time point can be characterized by its instantaneous equivalent conductance $G(t)$ defined as the ratio of $I$ and $V$ across the two terminals of the current path evaluated at that time instant[1]. Therefore, during the entire simulation process, we are simulating a circuit composed of only linear time-varying conductors and linear time-invariant elements.

Then, Eq.(3.1) can be transformed into the equation below:

$$\boldsymbol{G}(t)\mathbf{V}(t) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I_s}(t). \tag{3.2}$$

Here, $\boldsymbol{G}(t)$ represents the instantaneous equivalent conductance matrix for every branch in the circuit at time $t$. $\boldsymbol{G}(t)$ will satisfy the following relation

$$\boldsymbol{G}(t)\mathbf{V}(t) = \mathcal{F}(\mathbf{V}(t)) \tag{3.3}$$

for every time $t$.

[1] $G(t)$ is set to zero if $V(t) = 0$. The situation where $I(t) \neq 0$ when $V(t) = 0$ is practically impossible.

Instead of solving for the $\mathbf{V}(t)$ of Eq.(3.1) directly, the Stepwise Equivalent Conductance circuit simulation solves for the $\mathbf{V}(t)$ of Eq.(3.2) and uses it as the solution for Eq.(3.1). An efficient implicit integration scheme for Eq.(3.2) is developed, and no nonlinear equations need to be solved. The integration scheme will be introduced in section 3.2.

The question remaining is whether the solution of Eq.(3.2) will be equal to that of Eq.(3.1).

### 3.1.2 Exactness of the Transformation.

If we know $\boldsymbol{G}(t)$ beforehand, then the **uniqueness** of the solution of Eq.(3.1)[2] implies that the solution of Eq.(3.2) will be the same as the solution of Eq.(3.1). However, during the process of solving Eq.(3.2), for every time $t$, we only know the function $\boldsymbol{G}$ up to $t$ and have no idea of $\boldsymbol{G}$ after that. Will we end up with a different solution due to the lack of information on $\boldsymbol{G}$? To answer this question, we state the following theorem.

**Theorem 1** *If $\mathcal{F}(\cdot)$ and $\mathbf{I}_s(\cdot)$ of Eq.(3.1) are continuously differentiable then the solution of Eq.(3.2) will be exactly the same as that of Eq.(3.1).*

**Proof**: We prove this by contradiction. Denote the solution of Eq.(3.1) by $\mathbf{V}_1(t)$ and the solution of Eq.(3.2) by $\mathbf{V}_2(t)$. If they are not the same then there exists a time $t_0$ such that $\mathbf{V}_1$ and $\mathbf{V}_2$ coincide at $t_0$ but depart from each other afterwards, i.e. there exists a positive integer $k$ such that $\mathbf{V}_1^{(i)}(t_0) = \mathbf{V}_2^{(i)}(t_0) \ \ \forall \ \ i < k$ and $\mathbf{V}_1^{(k)}(t_0) \neq \mathbf{V}_2^{(k)}(t_0)$, where $\mathbf{V}_1^{(i)}(t_0)$ denotes the $i$-th derivative of $\mathbf{V}_1(t)$ evaluated at $t_0$.

[2]The sufficient condition of the uniqueness of the solution of Eq.(3.1) is that $\dot{\mathbf{V}}$ is Lipschitz continuous at every time $t$.

We know, from Eq.(3.1),

$$\dot{\mathbf{V}}_1(t) = \boldsymbol{C}^{-1}(-\boldsymbol{\mathcal{F}}(\mathbf{V}_1(t)) + \mathbf{I}_s(t)) \tag{3.4}$$

and from Eq.(3.2) and Eq.(3.3)

$$\begin{aligned} \dot{\mathbf{V}}_2(t) &= \boldsymbol{C}^{-1}(-\boldsymbol{G}(t)\mathbf{V}_2(t) + \mathbf{I}_s(t)) \\ &= \boldsymbol{C}^{-1}(-\boldsymbol{\mathcal{F}}(\mathbf{V}_2(t)) + \mathbf{I}_s(t)). \end{aligned} \tag{3.5}$$

Note, the inverse of $\boldsymbol{C}$ exists because $\boldsymbol{C}$ is diagonally dominant as mentioned before. From successive differentiations of $\dot{\mathbf{V}}_1(t)$ we have that $\mathbf{V}_1^{(k)}(t_0)$ is a function of $\mathbf{V}_1^{(i)}(t_0)$ and $\mathbf{I}_s{}^{(i)}(t_0)$ for $i < k$. Similarly, $\mathbf{V}_2^{(k)}(t_0)$ is the same function of $\mathbf{V}_2^{(i)}(t_0)$ and $\mathbf{I}_s{}^{(i)}(t_0)$ for $i < k$. Since $\mathbf{V}_1^{(i)}(t_0) = \mathbf{V}_2^{(i)}(t_0) \ \ \forall \ \ i < k$, we find that $\mathbf{V}_1^{(k)}(t_0)$ is equal to $\mathbf{V}_2^{(k)}(t_0)$, which contradicts that $t_0$ is a departing point for $\mathbf{V}_1(t)$ and $\mathbf{V}_2(t)$. ∎

Theorem 1 tells us that for the numerical integration of Eq.(3.2) although we do not know the function $\boldsymbol{G}$ after the current time $t$, we do know the time derivatives of $\boldsymbol{G}$ at $t$ up to the infinite order and thus can uniquely determine $\boldsymbol{G}$ for a small time interval beyond $t$. This can in turn be used to determine $\mathbf{V}$ for that small interval [3]. The sufficient condition of theorem 1 is that $\boldsymbol{\mathcal{F}}$ of Eq.(3.1) be continuously differentiable, which seems rather restrictive because it excludes piecewise continuous i-v characteristics. To relax this restriction for the purpose of numerical integration we

- use small time steps only when any piecewise characteristic in the circuit undergoes two different operating regions, and

[3] For the integration from $t_n$ to $t_{n+1}$, we only know $\mathcal{F}(\mathbf{V}(t))$ is equal to $G(t)\mathbf{V}(t)$ up to $t_n$. The above proves that $\mathcal{F}(\mathbf{V}(t))$ will be equal to $G(t)\mathbf{V}(t)$ from $t_n$ to $t_{n+1}$ once they coincide with each other up to $t_n$.

- use an absolutely stable integration scheme.

Then, even though the sufficient condition is not satisfied strictly, the numerical solution of Eq.(3.2) can still yield very good accuracy [4].

## 3.2 The Stepwise Equivalent Conductance Integration Algorithm.

For the integration of each time step, we assume that the equivalent conductances of the time-varying conductors remain constant during the time step. Therefore, for the calculation purpose, we are dealing only with linear constant circuit elements. The constant value assumed for each time-varying conductor can be determined to yield the necessary accuracy[5].

Using Taylor's series expansion of $G(t)$ at $t = t_n$, we obtain from Eq.(3.2)

$$[\boldsymbol{G}(t_n) + \dot{\boldsymbol{G}}(t_n)(t - t_n) + \frac{1}{2}\ddot{\boldsymbol{G}}(t_n)(t - t_n)^2 + \cdots]\mathbf{V}(t) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I}_s(t). \tag{3.6}$$

Let

$$h_n = t_{n+1} - t_n. \tag{3.7}$$

We show that for $t \in [t_n, t_{n+1}]$ Eq.(3.6) can be approximated by

$$\mathcal{G}\mathbf{V}(t) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I}_s(t), \tag{3.8}$$

[4] It must be emphasized that the above theorem provides only sufficient conditions. Hence, even if the circuit does not satisfy the continuous differentiability condition, the exactness of the transformation may still hold.

[5] At first glance, our approach may seem similar to Crystal [26] since both exploit the idea of using effective conductances. For the whole transition, Crystal replaces every MOS transistor by an effective conductor to estimate the timing information. There is no integration in Crystal. Whereas, our approach is trying to determine the effective conductances which can summarize the total electrical effects during a time step. Furthermore, the goal of our approach is to determine the complete transient characteristics.

with $\mathcal{G} = \boldsymbol{G}(t_n) + \frac{h_n}{2}\dot{\boldsymbol{G}}(t_n)$. To solve $\mathbf{V}(t_{n+1})$ from a given $\mathbf{V}(t_n)$ an error is introduced which is proven in Appendix B to be

$$-\boldsymbol{C}^{-1}\ddot{\boldsymbol{G}}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) - \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)(\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + \dot{\mathbf{V}}(t_n))(\frac{h_n^3}{12}). \tag{3.9}$$

By using the trapezoidal rule of integration, we obtain

$$\mathbf{V_{n+1}} = [\mathcal{G} + \frac{2}{h_n}\boldsymbol{C}]^{-1}(\frac{2}{h_n}\boldsymbol{C}\mathbf{V_n} + \boldsymbol{C}\dot{\mathbf{V}}_\mathbf{n} + \mathbf{I_{s_{n+1}}}). \tag{3.10}$$

This leads to the total local truncation error for the integration from $t_n$ to $t_{n+1}$ of the order $O(h^3)$. The method is therefore consistent with respect to the local truncation error, and since we know that the integration scheme, the trapezoidal rule, is absolutely stable, we have demonstrated the convergence of the algorithm.

## 3.3 Time Step Selection.

The local truncation error for each integration will be equal to the error given in Eq.(3.9) plus the error introduced from the trapezoidal rule approximation of $\dot{\mathbf{V}}(t_{n+1})$. Therefore, given the error criterion on the local truncation error at $t_n$, we can solve for the necessary time step $h_n$ exactly. A variable time step integration scheme can be implemented. However, Eq.(3.9) is very complicated. Determining $h_n$ involves several matrix operations. It would be impractical to perform the matrix operations at every time point. Therefore, we introduce a simpler scheme of choosing $h_n$. By using two parameters, a voltage error $\Delta V$ and a relative error $\epsilon$, we can derive the following: For each conductance $G_i$ and for each node voltage $V_j$, if $h_n$ meets the constraints imposed on Eq.(3.11), then the norm of

the error introduced in Eq.(3.9) will be less than $\frac{\epsilon}{3}\Delta V$, which is derived in Appendix C.

$$(|\frac{\ddot{G}_i(t_n)h_n^2}{G_i(t_n)}|) \leq \epsilon \quad \forall i, \qquad (a)$$

$$(|\frac{\dot{G}_i(t_n)h_n}{G_i(t_n)}|) \leq \epsilon \quad \forall i, \qquad (b)$$

and,

$$h_n\dot{V}_j(t_n) \leq \Delta V \quad \forall j. \qquad (c) \qquad (3.11)$$

The advantage of this is that the computation of determining a time step meeting all the constraints in Eq.(3.11) is linear in terms of nodes or devices in the circuit, while the computation needed to solve Eq.(3.9) is of the cubic order.

This concludes the introduction of our Stepwise Equivalent Conductance circuit simulation technique. As the first application of the technique, a digital CMOS circuit simulator SWEC was implemented, which will be discussed in the next chapter.

# Chapter 4

# SWEC: A Stepwise Equivalent Conductance CMOS Circuit Simulator.

When applying the Stepwise Equivalent Conductance Circuit Simulation Technique to digital CMOS circuits, we demonstrate that additional speedups can be achieved by the use of a specific event-driven approach to take advantage of the piecewise linear waveforms. Most of the time the voltage waveforms of the outputs from CMOS gates behave like straight line segments. We will show how to make use of this property in the timing simulation in this chapter. The program, called SWEC, has been implemented, and has proven to be accurate and efficient on a large number of circuit examples. The comparisons of results with Relax2.3[6], iSPLICE3.0[7, 17], XPsim[8], and SPECS2[9] will be given in this chapter.

The chapter is organized as follows. Section 4.1 gives an introduction to SWEC. In section 4.2, we discuss SWEC's MOS Electrical models and the time step selection for it. Section 4.3 presents our piecewise linear waveform approximation. In section 4.4, we discuss SWEC's event-driven mechanism, which exploits the piecewise linear voltage waveform property, in detail. In section 4.5, we present experimental results of SWEC along with comparisons with SPICE, iSPLICE3.0, Relax2.3, XPsim, and SPECS2 programs.

## 4.1 Introduction.

As an application, a timing simulator for digital CMOS VLSI circuits, SWEC, has been implemented based on the concepts introduced in chapter 3 and [4, 10]. To further speed up the simulation, SWEC first decomposes the circuit into weakly coupled subcircuits and applies the Stepwise Equivalent Conductance technique to each of the subcircuits. In addition, SWEC exploits another special property of CMOS circuits, that is, the voltage waveform can be modeled with piecewise-linear segments connected between regions with smooth curves. Thus, the voltage waveforms of the outputs from CMOS gates behave like straight line segments most of the time. Because of this property, larger time steps can be used. To handle feedback inside the circuits and to further exploit the latency and multirate behavior of MOS circuits, a special event driven mechanism based on the piecewise linearity of waveforms has also been developed and built into SWEC. We have developed an algorithm to determine the break points of the piecewise-linear voltage waveforms under the desired accuracy requirements.

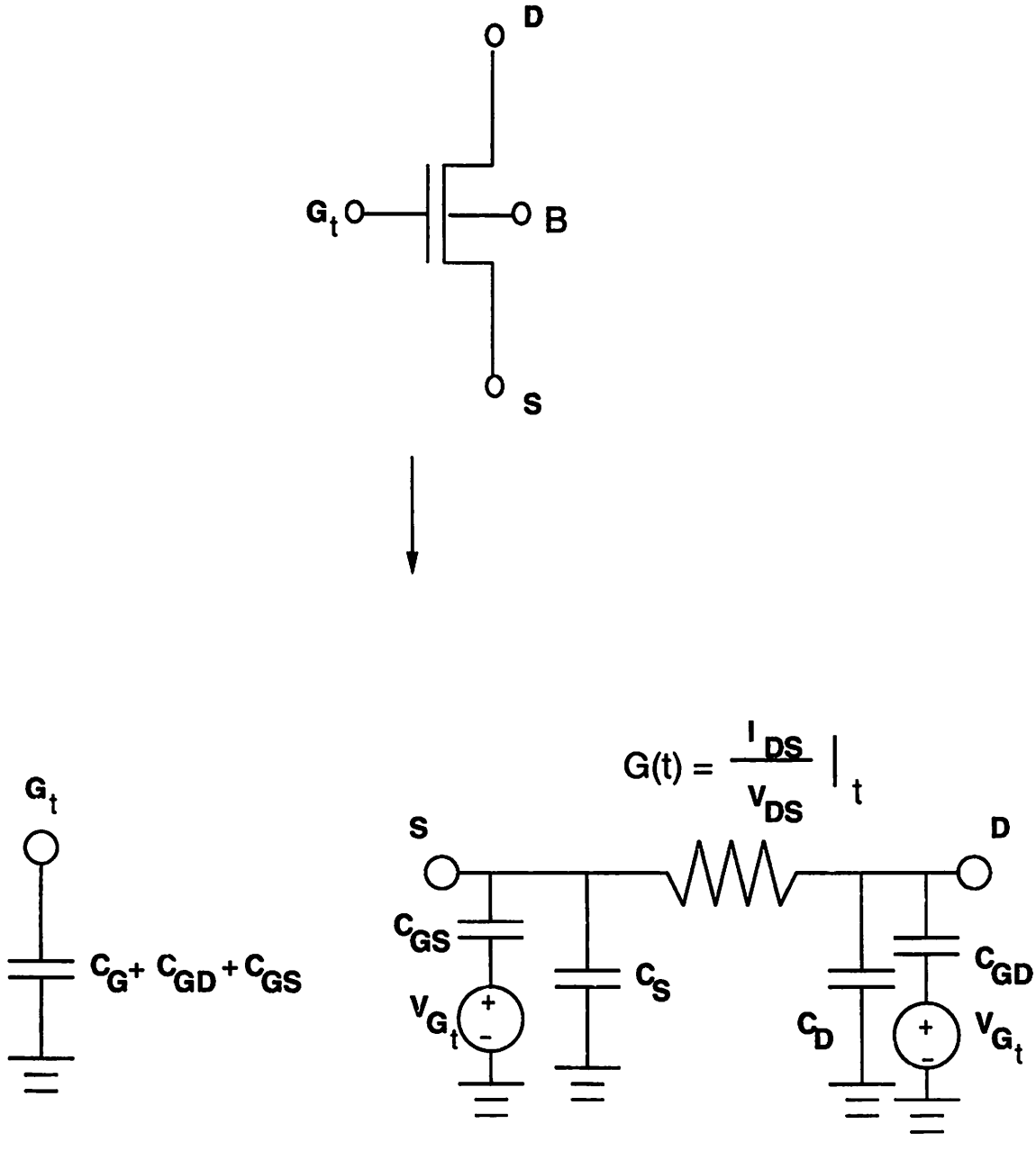


Figure 4.1: Electrical Model for a MOS transistor

## 4.2 MOS Electrical Models and Time Step Selection of SWEC.

The analytical expression for the $G(t)$ of a MOS transistor is given by the conventional formula:

$$\begin{aligned} G(t) = \beta * (2 * (V_{GS} - V_{th}) - V_{DS})|_t \quad & if \quad V_{GS} - V_{th} \geq V_{DS} \geq 0, \\ G(t) = \beta * \frac{(V_{GS} - V_{th})^2}{V_{DS}}|_t \quad & if \quad V_{DS} > V_{GS} - V_{th} \geq 0, \\ G(t) = 0.0, \quad & if \quad V_{GS} < V_{th}, \end{aligned} \tag{4.1}$$

where $V_{th}$, the threshold voltage, is a function of $V_{SB}$. Fig. 4.1 shows the electrical model of a MOS transistor. Between the drain node and the source node there is a time varying conductor with the conductance $G(t)$. The voltage at the gate node is represented by the voltage sources $V_{G_t}$. The grounded capacitors, $C_D$ and $C_S$, represent the side-wall and bottom junction grounded capacitances at the drain and the source nodes, respectively. $C_G$ is the gate oxide capacitance and wiring capacitance. $C_{GD}$ and $C_{GS}$ represent the gate-to-drain and gate-to-source overlap capacitances, respectively. Note that they are lumped with $C_G$ at the gate node $G_t$ as an approximate total capacitance looking at that node. Since in general $C_G$ is much larger than $C_{GS}$ or $C_{GD}$, turning $C_{GS}$ or $C_{GD}$ into grounded capacitors is a reasonable simplification when looking at the gate node. However, this is not true when looking at the source or drain node because $C_D$ is comparable with $C_{GD}$, and $C_S$ is comparable with $C_{GS}$. We account for this by introducing the two voltage sources $V_{G_t}$ representing the effects coming from the gate node.

In this way, the determination of the gate voltage does not depend on the voltage at the source node or the voltage at the drain node as long as there is no other charge transfer

path (resistor, capacitor, transistor channel) connected between them. Furthermore, if the gate voltage is evaluated prior to the evaluation of the voltages of the source node and the drain node then $V_{G_t}$ can be treated as a constant voltage source in determining the voltages of the source node and the drain node. If we can keep this ordering correct during the simulation, then solving the voltage at the gate node can be separated from solving them at the source and the drain nodes, and no iterations are needed between the two solving processes. Our event-driven approach is based on this idea and will be discussed in section 4.4.

Since there is no direct charge transfer path between the gate node and the conducting channel of the transistor (one-way circuits), we are able to perform circuit partitioning [40]. Prior to simulation, the circuit is partioned into subcircuits. They are tightly coupled groups of nodes connected by a charge transfer path. Each subcircuit is integrated with its own time step to take advantage of the time latency existing in the circuit. The integration of each subcircuit employs the stepwise equivalent conductance technique. For a subcircuit, the time steps are determined according to the slopes at inputs and the value of the equivalent conductance of each transistor before integration.

The conductance of each transistor used for the integration from $t_n$ to $t_n + h_n$ is $\mathcal{G}_n$, which is equal to $G(t_n) + \frac{h_n}{2}\dot{G}(t_n)$. The expressions are as follows:

$$\begin{aligned}
\mathcal{G}_n &= G(t_n) + \frac{h_n}{2}\beta * (2\dot{V}_{GS} - \dot{V}_{DS})|_{t_n} \quad if \quad V_{GS} - V_{th} \geq V_{DS} \geq 0, \\
\mathcal{G}_n &= G(t_n) + \frac{h_n}{2}\beta * (2\frac{V_{GS} - V_{th}}{V_{DS}}\dot{V}_{GS})|_{t_n} \quad if \quad V_{DS} > V_{GS} - V_{th} \geq 0, \\
\mathcal{G}_n &= 0.0 \quad if \quad V_{GS} < V_{th}. \qquad (4.2)
\end{aligned}$$

For the sake of efficiency, we neglect those terms associated with $\frac{d}{dt}V_{th}$ in deriv-

ing $\dot{G}(t)$ of Eq.(4.2). For the situation that $V_{DS} > V_{GS} - V_{th}$, the term associated with $\dot{V}_{DS}\frac{\partial}{\partial V_{DS}}\frac{(V_{GS}-V_{th})^2}{V_{DS}}$ is also neglected because after the differentiation, we will have the result $-\dot{V}_{DS}(\frac{V_{GS}-V_{th}}{V_{DS}})^2$, which is considered to be small due to $V_{DS} > V_{GS} - V_{th}$. In Appendix D, we demonstrate that, for a CMOS inverter, the term $\dot{V}_{DS}(\frac{V_{GS}-V_{th}}{V_{DS}})^2$ is negligible compared with the rest terms of $\dot{G}(t)$.

By making use of the piecewise linearity and from Eq.(3.11), we find that the time step selection of SWEC for each subcircuit is

$$h_n = MIN(\frac{\Delta V}{\dot{V}_j(t_n)}, \epsilon\frac{|(V_{GS} - V_{th})_i|}{|\frac{d}{dt}(V_{GS} - V_{th})_i|}|_{t_n}),\ \forall i,\ \forall j \tag{4.3}$$

where $i$ is the index of the transistors which are ON and $j$ is the index of nodes. The $\frac{\Delta V}{\dot{V}_j(t_n)}$ term inside the $MIN$ function comes directly from Eq.(3.11c); the other term function is due to the consideration of Eq.(3.11b). The $\dot{G}(t_n)$ of Eq.(3.11b) can be derived from the differentiation of Eq.(4.1). Eq.(3.11a) is not considered in Eq.(4.3) because the node voltage waveform is piecewise linear. Hence the second derivative for any node voltage vanishes.

## 4.3 Piecewise Linear Approximation of Waveforms.

After each integration of a subcircuit, a piecewise linear approximation is applied to the output waveform of the subcircuit. This piecewise linear waveform can then be fed to the fanouts of the subcircuit as one of their inputs. Thus, during the simulation the input and the output waveforms of every subcircuit are piecewise linear. When scanning the data points of the original waveform, the piecewise linear approximation tries to skip as many points as possible as long as those skipped points stay on an approximated line segment. Those points which are not skipped are kept as the break points of the piecewise

linear curve. The break points of a piecewise linear curve are the data points at which the curve changes slope.

To determine which points can be skipped, we have developed a local error criterion. The original waveform is assumed to be fed into a CMOS inverter as the input. The corresponding output waveform can be determined. Then, the errors on the output of the inverter are monitored when piecewise linear approximations are applied to the original waveform. The approximation algorithm searches for a piecewise linear approximation of the original waveform with errors within a pre-specified error bound and thus obtains as few break points as necessary. The tighter the error bound, the more points of the original waveform will be left as break points.

The details of the algorithm are as follows. For every point on the waveform, we calculate the worst possible relative error on the inverter's output when the extrapolation from the two preceding data points is used, instead, as the input. As shown in Fig. 4.2, $(t_i, v_i)$ is the point being considered currently and $(t_i, \hat{v}_i)$ is the extrapolated point from $(t_{i-2}, v_{i-2})$ and $(t_{i-1}, v_{i-1})$ at $t_i$. We check all the possible relative errors[1] on the inverter's output at $t_i$ when the line segment from $(t_{i-1}, v_{i-1})$ to $(t_i, \hat{v}_i)$ is used as the input. If the worst relative error violates a given local error bound, then the immediately preceding point, which is $(t_{i-1}, v_{i-1})$ in Fig. 4.2, should be kept as a break point. If not, the immediately preceding point can be skipped. When the local error criterion is violated, the waveform has a relatively large slope change around the immediately preceding point, and so it should be kept as a break point. The situation is shown pictorially in Fig. 4.2.

[1]Since the inverter can be in different operating regions and the effect of the approximation on its input may be different, we need to consider all the possible situations.

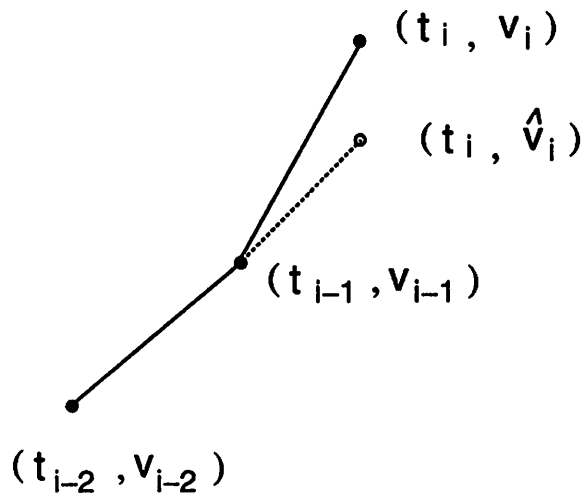


Figure 4.2: Piecewise-Linear Waveform Approximation.

Hence, the piecewise linear approximation algorithm we developed for SWEC is based on examining every three consecutive samples, say $(t_{i-2}, v_{i-2})$, $(t_{i-1}, v_{i-1})$, and $(t_i, v_i)$: if

$$|v_i - v_{i-1} - (t_i - t_{i-1})\frac{v_{i-1} - v_{i-2}}{t_{i-1} - t_{i-2}}| > 2\epsilon', \tag{4.4}$$

then $(t_{i-1}, v_{i-1})$ is kept as a breakpoint. Otherwise, $(t_{i-1}, v_{i-1})$ is not a breakpoint. The next triplet to be examined is: $(t_{i-1}, v_{i-1})$, $(t_i, v_i)$, and $(t_{i+1}, v_{i+1})$.

The $\epsilon'$ is the constant controlling the tolerable percentage errors. The number 2 in Eq.(4.4) is the average voltage value of $V_{GS} - V_{th}$ during a transition. The detailed derivation of Eq.(4.4) is given in Appendix E.

In [4], an example is given. Seven breakpoints out of 228 data points of the glitch output of a NAND gate were picked by our algorithm. Then, the piecewise-linear waveform was fed into an inverter. Again, eight breakpoints out of 169 data points were picked from the output of the inverter. Both piecewise-linear waveforms show a very good match with

the SPICE results.

## 4.4 The Event-Driven approach of SWEC.

SWEC uses an event-driven mechanism to handle feedback loops in large circuits and further exploit the latency of MOS circuits. The time instants at which each individual subcircuit is to be evaluated are termed event times for the subcircuit. Thus an event is a prediction of the time when integration should take place for the time step between the previous event time and the current event time. Events are predicted, stored, and scheduled on an event queue as in [23, 32, 9]. The prediction is based on the information at the input slopes and the conductances of MOS transistors in the subcircuit given by Eq.(4.3). After the integration of a subcircuit, the slope of its output may or may not change according to the piecewise linear waveform approximation introduced in section 4.3. If the slope changes, then the events associated with its fanout subcircuits will be incorrect because they were based on the previous slope. It is thus necessary to delete existing events for the fanouts and reschedule them by using the new slope for Eq.(4.3).

The whole simulation can be viewed as a series of event processes. Processing the event at the head of the queue involves the following steps:

1. Integrate the subcircuit for the time step between the subcircuit's previous event time and the current time.
2. Use the piecewise linear waveform approximation to check if the output slope has changed; if yes, delete the events of all the fanouts and reschedule them.
3. Calculate the time step of the subcircuit and insert an event associated with

the subcircuit into the queue with the event time being equal to the sum of the current time and the time step.

The procedure works for circuits with feedback loops and can be justified as follows. Let us consider the circuit given in Fig. 4.3, which has two subcircuits, A and B, and a feedback loop. The output of A, called x, is fed to B and the output of B, called y, is fed back to A. The next event time (or the next integrating time point) of A is determined by choosing the minimum of the two predictions (time step criterion):

- when x will change slope,
- when the voltage change of any node in A goes beyond a $\Delta V$.

The derivation of the $h_n$ given in Eq.(4.3) satisfies the above criterion for small $\Delta V$ and $\epsilon$. We discuss it in Appendix D. Therefore, for the integration of A, say from $t_n$ to $t_{n+1}$, the slope of x will not change, which guarantees that y is correct from $t_n$ to $t_{n+1}$. This implies that the integration result for x is correct because y is as predicted. Even though, after the integration, x may change slope beyond $t_{n+1}$, causing y to change slope, the correctness of the results up to $t_{n+1}$ still holds. By induction, the results of our approach are correct for the entire simulation.

The $h_n$ satisfying the criterion given above for circuits with feedback loops will be greater than zero because of the time latency in the circuits. The output slope of a subcircuit does not change immediately after any of its input changes slope. However, it is possible that a negative feedback loop will have very small or even zero time latency. In the negative feedback case, the subcircuits along the loop should be collapsed into one bigger

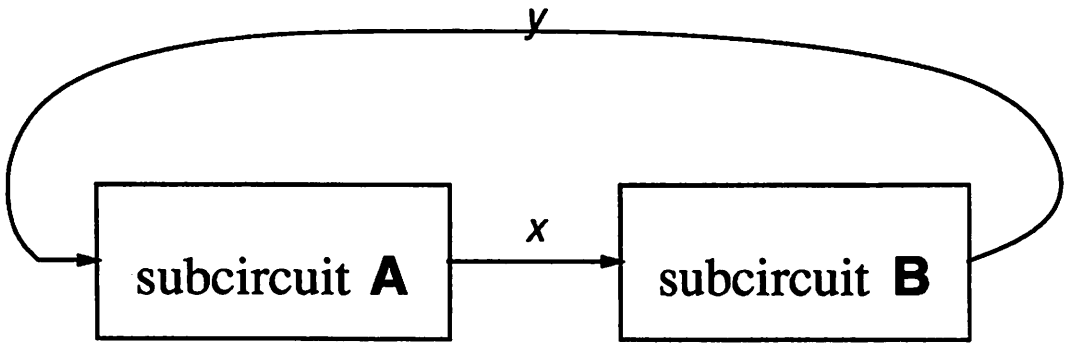


Figure 4.3: A circuit with a feedback loop.

subcircuit. The collapsing steps will not happen very often for digital circuits because the feedback loops inside digital circuits, such as those for flip-flop latches, are usually positive.

Event rescheduling using conventional means based on voltage level [23] [7] is computationally expensive. In SWEC the overhead is very small because the waveforms are assumed to be piecewise-linear. A subcircuit's fanouts need to be rescheduled only when its output waveform changes slopes. Since the piecewise linear output waveforms of digital MOS circuits seldom change slopes, relatively few reschedulings are necessary. If the output waveform from a subcircuit is a straight line segment then no event rescheduling will be needed for all its fanout subcircuits because the first prediction on the output waveform is true for all the time instances. Additionally, to avoid large computational overhead in event rescheduling, SWEC uses a very simple mechanism for calculating the updated event time. Only the second term inside the $MIN$ of Eq.(4.3) needs to be reevaluated.

| Simulator | CPU time | Period | % error |
|---|---|---|---|
| SPICE | 470 s | 4.52 ns | 0% |
| SWEC | 0.6 s | 4.52 ns | 0% |
| iSPLICE3.0 | 1.7 s | 4.52 ns | 0% |
| Relax2.3 | 1.2 s | 4.52 ns | 0% |
| SPECS2 | 1.7 s | 4.35 ns | 3.7% |
| XPsim | 1.8 s | $\infty$ ns | 100% |

Table 4.1: CPU times for a Ring Oscillator.

## 4.5 Experimental Results for SWEC.

We compared the performance of SWEC with the simulators SPICE, SPECS2, iSPLICE3.0, XPsim, and Relax2.3. Note that SPECS2 and XPsim use the explicit integration approach. The others use the implicit integration approach. The current version of SWEC uses the SPICE level one process technology. To make fair comparisons, we used the same device models and the same capacitance models for all the simulators. Unless otherwise mentioned all the simulations are performed on a DEC station 5000.

### 4.5.1 Ring Oscillator with 7 inverters.

The first test circuit was a ring oscillator made of an inverter chain of 7 inverters. The summary of accuracy and CPU times is listed in Table 4.1. The **Period** column lists the period of the waveform obtained from each simulator. The % error refers to that compared with the SPICE results. We did not observe any oscillation on the result of XPsim, so we have tabulated the error as 100%.

| Simulator | CPU time |
|---|---|
| SPICE | 174 s |
| SWEC | 5.6 s |
| iSPLICE3.0 | 82.6 s |
| Relax2.3 | 11.8 s |
| SPECS2 | 159 s |
| XPsim | 141 s |

Table 4.2: CPU times for 8-bit Ripple Carry Adder.

### 4.5.2 8-bit Ripple Carry Adder.

We built an 8-bit ripple carry adder with about 442 MOS transistors. We applied 8 different input vectors and monitored the carry-out waveform of the adder. In the worst case, the carry would propagate from the carry-in through the whole adder to the carry out. In this way, the errors would be accumulated so we could tell the accuracy of each simulator. The waveforms from the simulators are depicted in Fig 4.4. The solid line is the result of SPICE, which is assumed to be the correct result. We found that the results of SWEC, XPsim, iSPLICE and Relax2.3 completely overlapped the SPICE result. The result of SPECS2 had a slight error as shown. Table 4.2 lists the summary of the CPU times.

### 4.5.3 16-bit Multipliers.

We simulated two 16-bit multipliers of different designs with about 7200 transistors and 6700 transistors, respectively. We denote the multiplier with 7200 transistors as MULT7200 and the one with 6700 transistors as MULT6700. 12 input vectors were used on the simulations of the two multipliers. Part of the multipliers were made of pass gate logic, which means certain subcircuits would have quite a few transistors. The largest subcircuit has 24 transistors and 10 nodes.

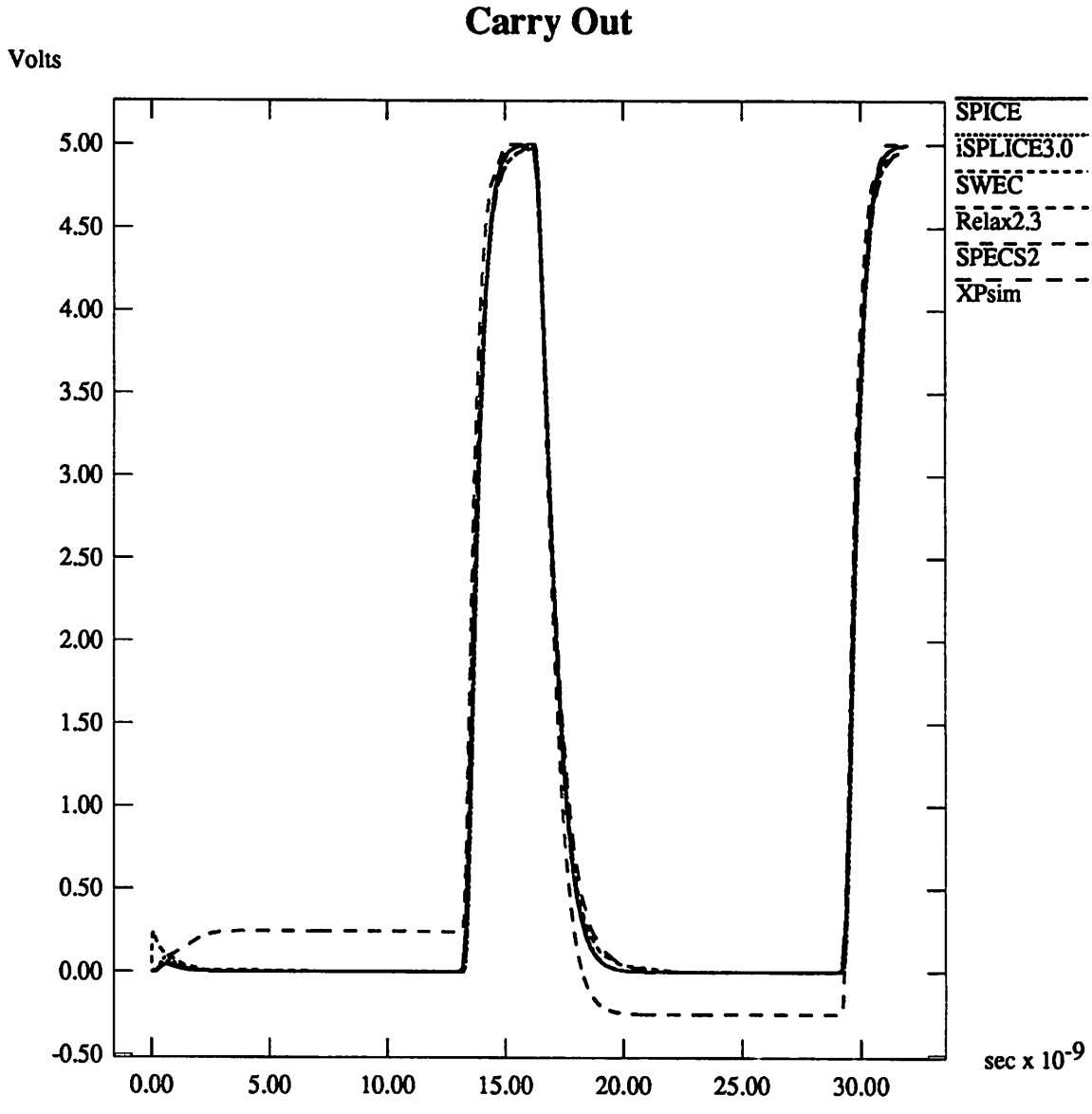


Figure 4.4: 8-bit Ripple Carry Adder.

| Simulator | CPU time |
|---|---|
| SPICE | 8800 s |
| SWEC | 146 s |
| iSPLICE3.0 | 8780 s |
| Relax2.3 | 179.6 s |
| XPsim | 2165 s |
| SPECS2 | 1092 s |

Table 4.3: CPU times 16-bit Multiplier(MULT7200).

| Simulator | CPU time |
|---|---|
| SPICE3 | 53532 s |
| SWEC | 240 s |
| iSPLICE3.0 | 560 s |

Table 4.4: CPU times 16-bit Multiplier(MULT6700).

Fig. 4.5 shows the waveforms of the 32nd bit output (the most significant bit) from different simulators for the simulations of MULT7200. We found that the results from SWEC, iSPLICE3.0, and Relax2.3 overlapped. The SPICE result convinced us that all of them were correct. The simulation for SPECS2 was performed on an IBM RS/6000. The corresponding CPU time was measured on the IBM RS/6000, which is about 2.5 times faster than DEC stations 5000. Table 4.3 gives the summary of the CPU times.

Fig 4.6 shows the waveform of the 18th bit output from our SWEC, SPICE3 and iSPLICE3.0 for the simulations of MULT6700. Relax2.3 has convergence problem when simulating the circuit so we did not include its result here. We observed that the waveforms from the three simulators coincide together. Table 4.4 gives the summary of the CPU times on a DEC station 5000/125.

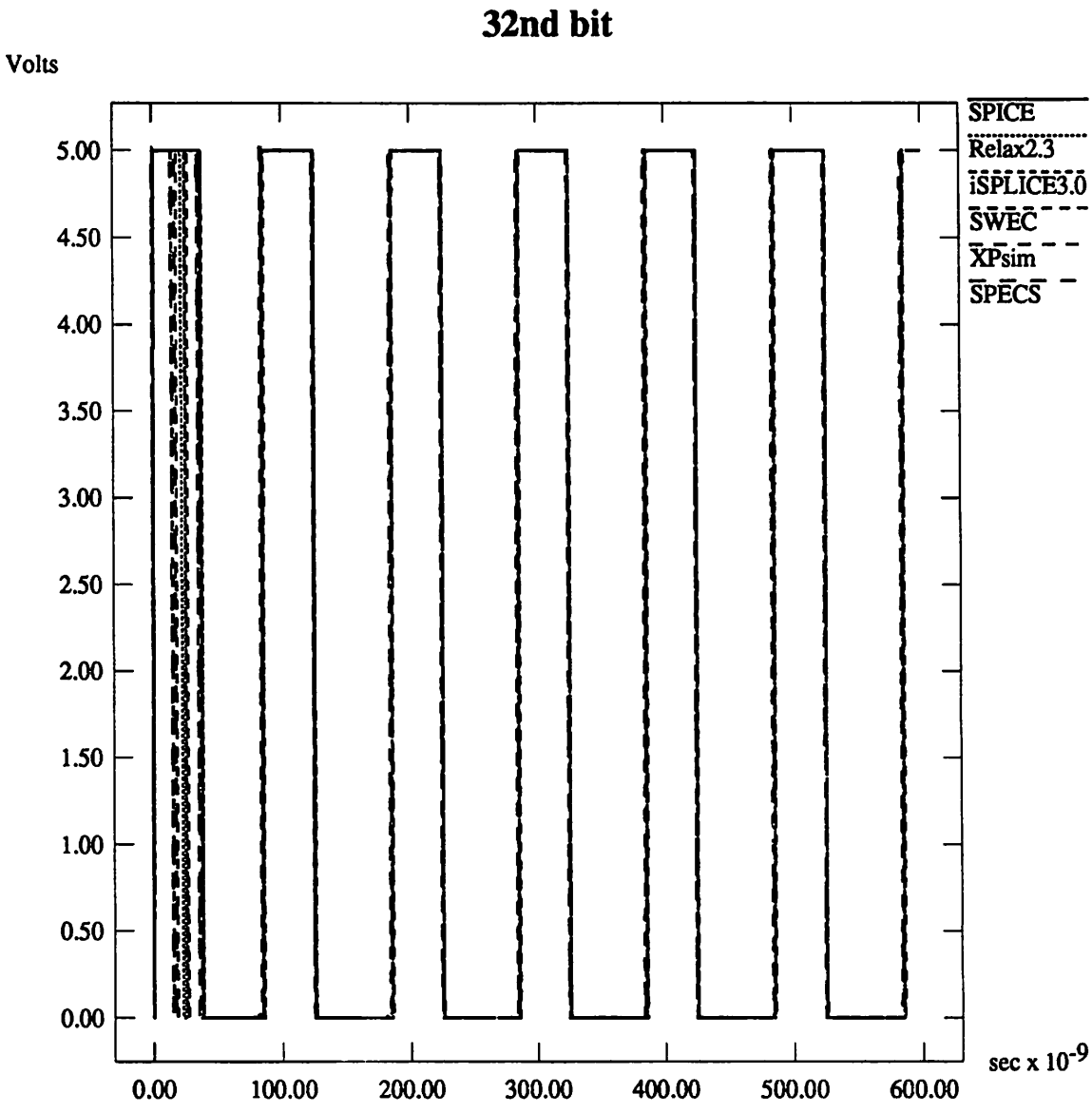


Figure 4.5: 16-bit Multiplier(MULT7200).

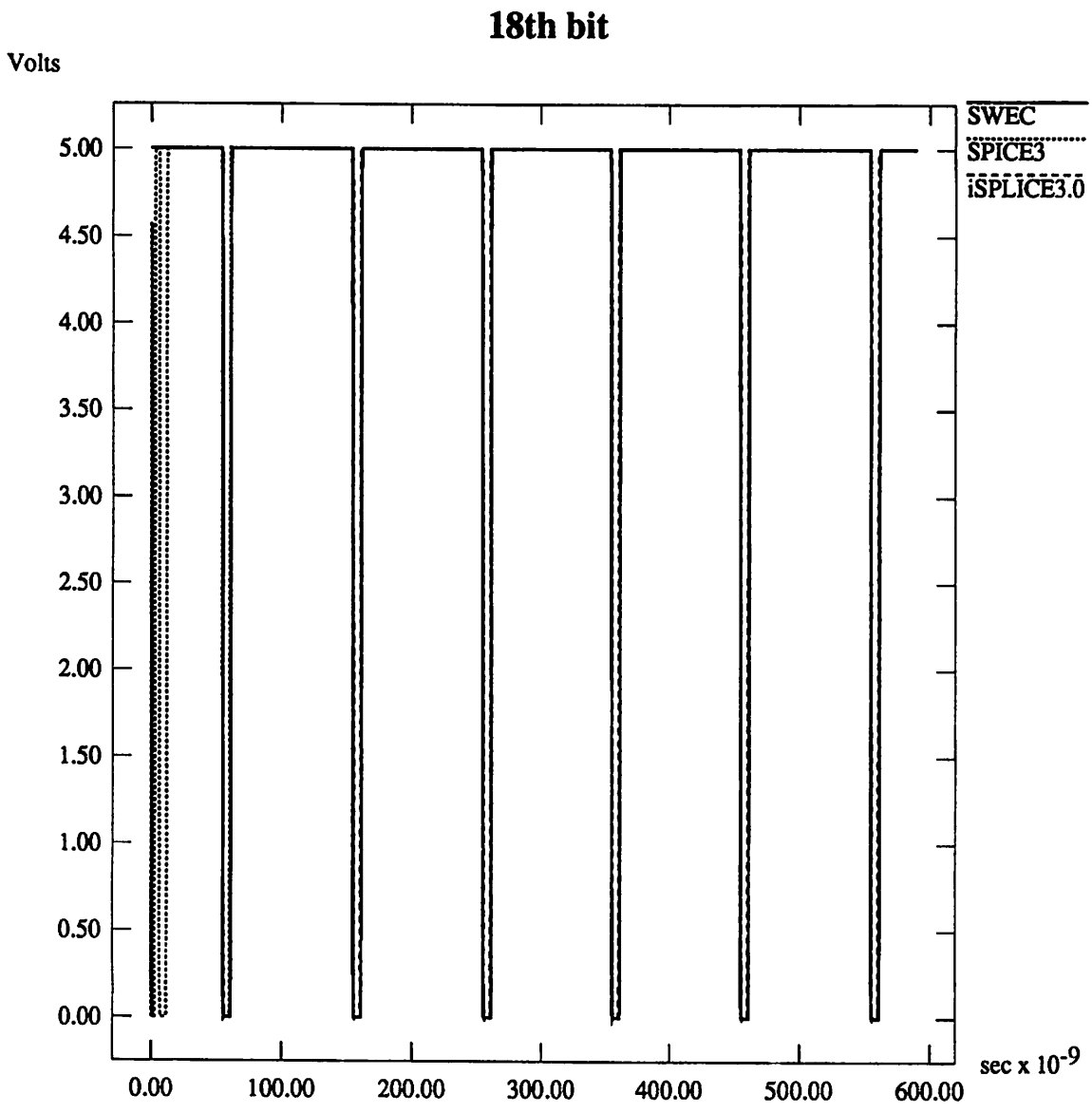


Figure 4.6: 16-bit Multiplier(MULT6700).

| Simulator | CPU time |
|---|---|
| SPICE | 2.02 s |
| SWEC | 0.2 s |
| iSPLICE3.0 | 0.8 s |
| Relax2.3 | 0.5 s |
| SPECS2 | 2.3 s |
| XPsim | 5.1 s |

Table 4.5: CPU times for a Stiff Circuit.

### 4.5.4 A Stiff Circuit.

To evaluate the stiff stability, we simulated a stiff circuit on all simulators. The circuit was an output pad with 14 transistors and 8 very small resistors, each connected with very small node capacitors. The time constants associated with those resistors were about two to three orders smaller than the input rise or fall time. The waveforms from the simulators are depicted on Fig. 4.7. Table 4.5 summarizes the CPU times.

Since SWEC is absolutely stable, the time step selection for SWEC is based on the accuracy consideration alone. From the simulation results, we found that SWEC was indeed both accurate and efficient. The 0.2s CPU time is the minimum computation for SWEC which is used to build the look-up table for device evaluation routines. We did not observe the expected numerical blow-ups in the results of SPECS2 and XPsim. We believed that the minimum time steps were used for most of the simulation to assure stability. In fact, the CPU times were more than an order of magnitude larger than the other implicit methods.

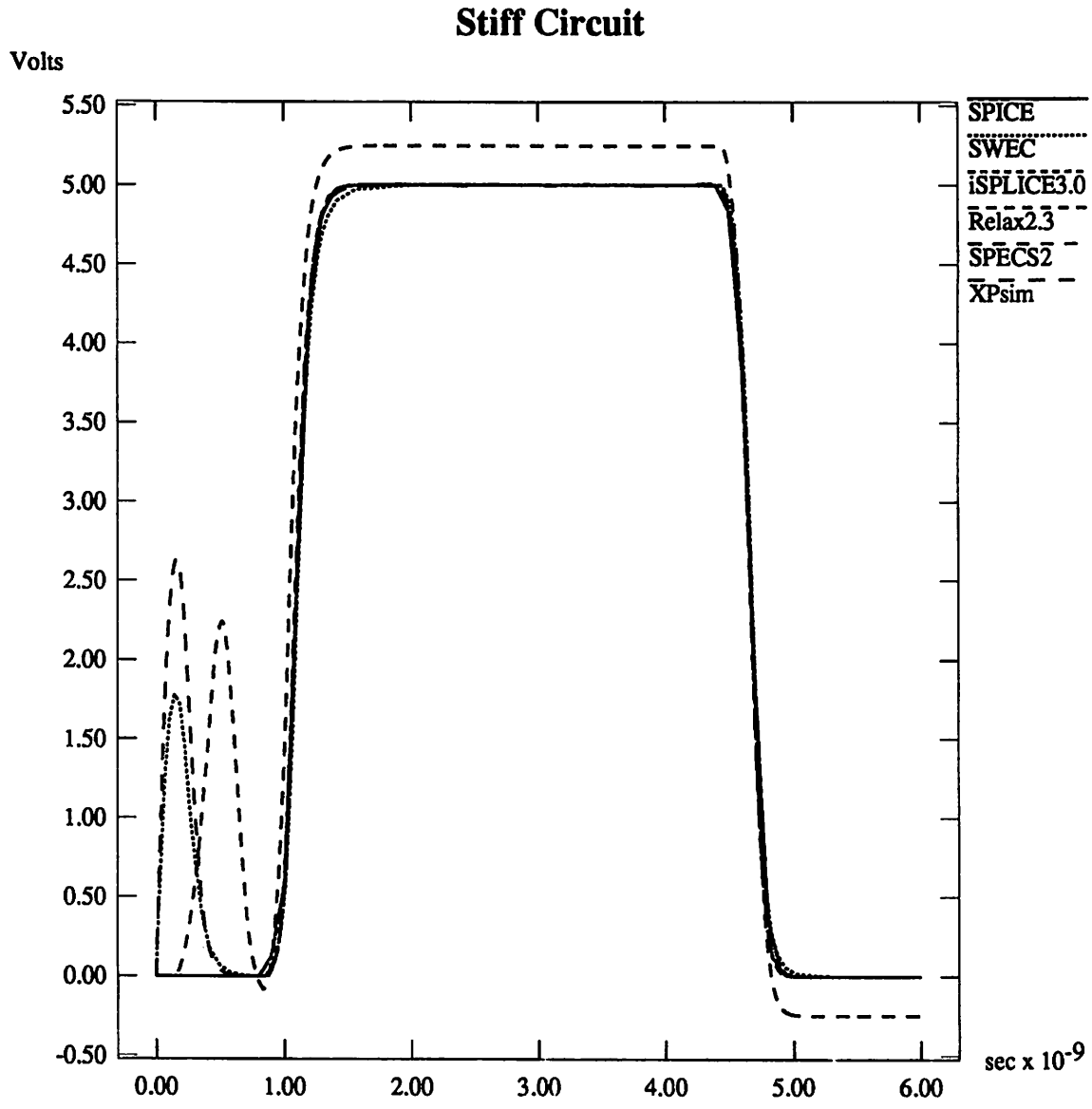


Figure 4.7: A Stiff Circuit.

| Simulator | CPU time |
|---|---|
| SWEC | 1,200 s |
| iSPLICE3.0 | 11,394 s |
| Relax2.3 | 56,480 s |

Table 4.6: CPU times for a Micro-Processor.

### 4.5.5 Micro-Processor.

We simulated a large circuit, which is a part of a micro-processor built in industry, on SWEC, Relax2.3 and SPLICE3.0. The circuit has about 11,300 nodes and 32,000 MOS transistors. The input has six vectors. The biggest subcircuit of the circuit is composed of about 131 nodes. Therefore, it is considered to be a difficult case for timing simulation. Fig. 4.8 and Fig. 4.9 show the waveform comparisons for two different nodes. The results from the three simulators coincided. The consumed CPU times on a DEC station 5000/125 are summarized in Table 4.6. We observed that iSPLICE3.0 was about five times faster than Relax2.3 and SWEC, however, was about 10 times faster than iSPLICE3.0. SPICE3 could not finish the simulation of this circuit successfully due to memory allocation errors so we did not include its results here.

The above experimental results show that our Stepwise Equivalent Conductance implicit integration and piecewise-linear event-driven simulation are very efficient, stable, and accurate. For the simulations of the multipliers and the micro-processor, SWEC used about only 10% of the sizes of memory used by Relax2.3 or by SPLICE3.0.

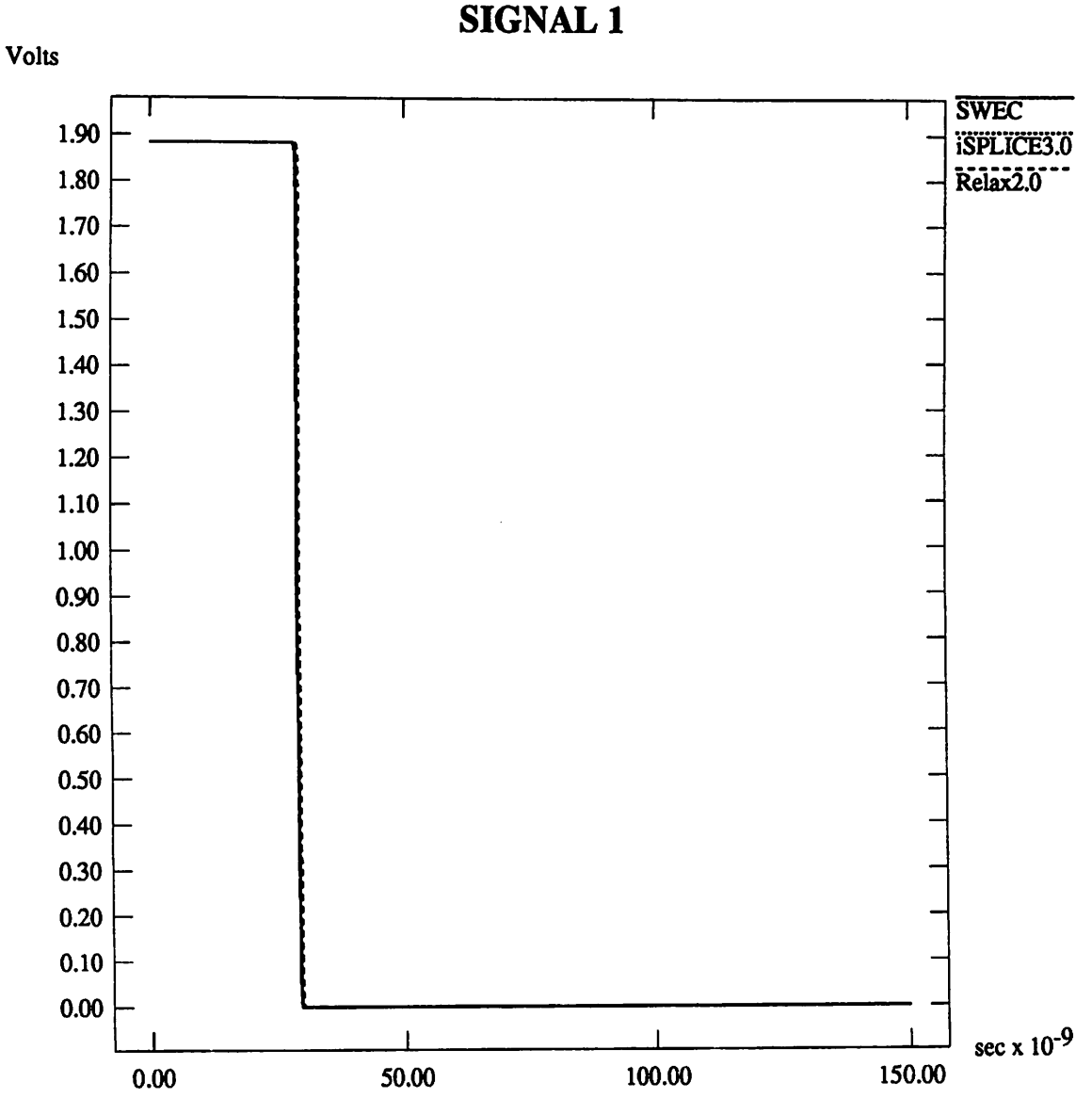


Figure 4.8: Simulation of a Micro-Processor (a).

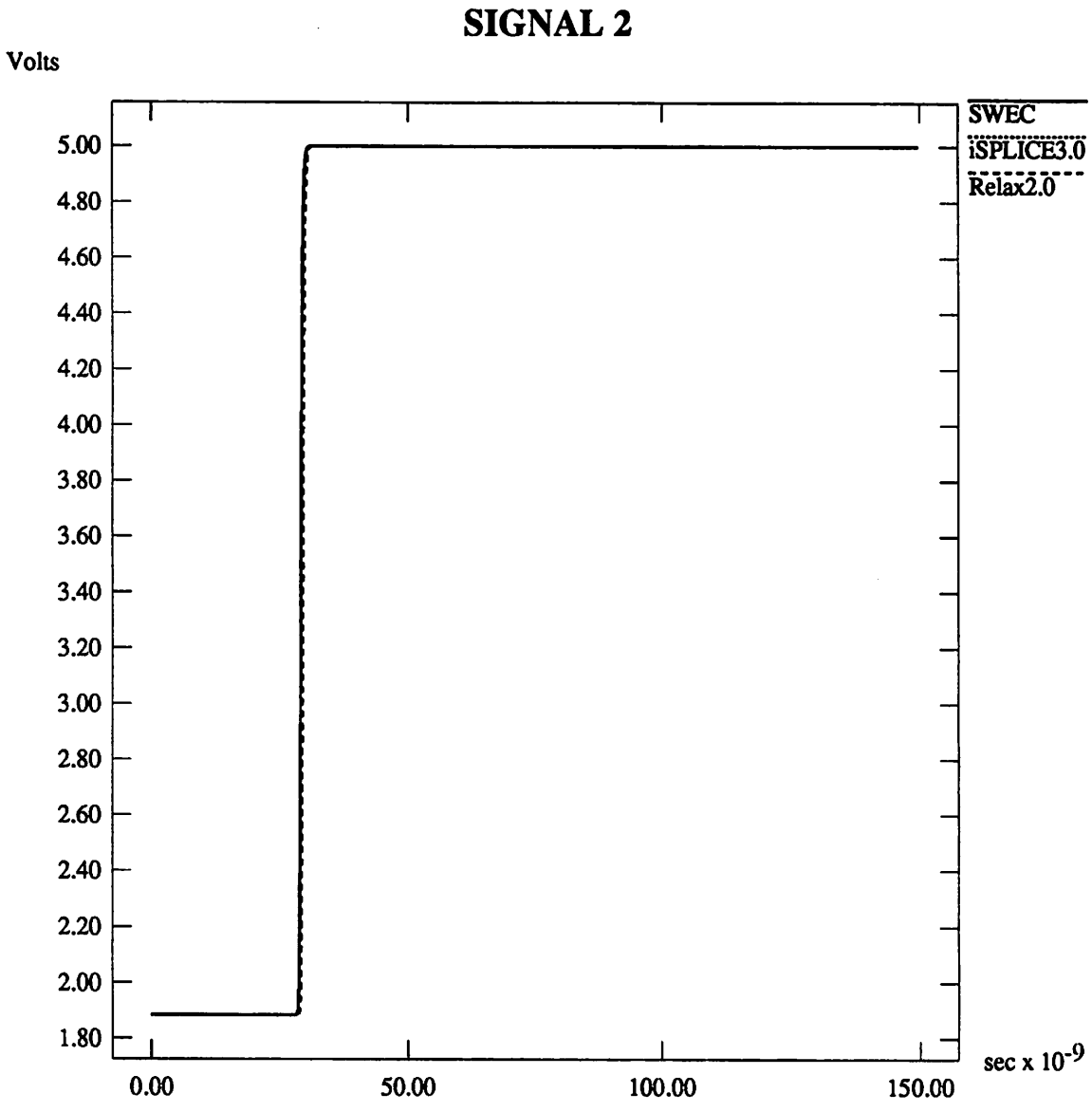


Figure 4.9: Simulation of a Micro-Processor (b).

# Chapter 5

# Transient Simulation of Lossy Interconnects Based on the Recursive Convolution Formulation

A new approach for transient simulation of lossy interconnects terminated in arbitrary nonlinear elements will be presented in this chapter. The approach is based on convolution simulation. By making use of the Pade approximation (or the reduced moment method), we can determine the impulse response of an arbitrary transfer function. Furthermore, by employing the particular form of the impulse response, we develop a recursive convolution formulation, which greatly reduces the computations used to perform convolution integrations. The approach can handle frequency-varying effects, such as skin effects,

and general coupling situations. To handle general coupling situations, a fitted polynomial decoupling approach is presented. We give mathematical analyses of the errors introduced by the Pade approximation and propose a scheme to determine the necessary order of the approximation.

We have incorporated the proposed approach into the Stepwise Equivalent Conductance timing simulator, SWEC, for digital CMOS circuits. Hence, Newton Raphson iterations are not needed for the implicit integration of a circuit even with lossy lines terminated in nonlinear elements. The comparisons with SPICE3.e [11] indicate that SWEC, which gives accurate results, can be one to two orders-of-magnitude faster.

The Pade approximation has been used in the literatures [24], [41], [42], and [43] for the delay estimation of linear $RLC$ networks. The transfer function of the output (or outputs) of the network is expanded into a Maclaurin series of $s$, the Laplace transform variable, around $s = 0$. The series is then truncated to a necessary order. The first order Elmore delay will be the first moment of the series. An approximated time domain output waveform can also be determined by matching the truncated transfer function. Our approach is different from their approaches in the following two aspects:

1. In our approach, the transfer function is expanded into a series of $\frac{1}{s}$ around $s = \infty$ to consider the correct initial condition and the high frequency responses.
2. In the above moment matching approaches, the transfer functions of the outputs of a distributed network will have an infinite number of poles (an infinite number of natural frequencies), whereas the transfer functions that are approximated by the Pade approximations in our approach have only very few poles located on

the left half of the $s$ plane.

The first point will be explained in more detail when we analyze the errors introduced by the Pade approximation in section 5.5. The second point will be discussed in section 5.1.

This chapter is organized as follows. In section 5.1, we give a brief introduction to the background of convolution simulation. Section 5.2 describes our Pade approximations and recursive convolution formulation. In section 5.3, the approach for the transient simulation of lossy coupled lines will be presented. Section 5.4 describes the implementation of our lossy interconnect simulation approach into the Stepwise Equivalent Conductance Timing Simulator, SWEC. In section 5.5, we analyze the approximation errors. In section 5.6, we present the experimental results along with comparisons with SPICE3.e [11].

## 5.1 Background of Convolution Simulation.

### 5.1.1 Convolution Simulation of Simple *RLGC*-lines.

The Telegrapher Equations for a *RLGC*-line are:

$$\frac{\partial v}{\partial x} = -(L\frac{\partial i}{\partial t} + Ri) \tag{5.1}$$

$$\frac{\partial i}{\partial x} = -(C\frac{\partial v}{\partial t} + Gv) \tag{5.2}$$

The transmission line as shown in Fig. 5.1 stretches from 0 to $l$ on the $x$ coordinate, where $l$ is the length of the line; $v(x,t)$ is the voltage at point $x$ at time $t$; $i(x,t)$ is the current in the $+x$ direction at point $x$ at time $t$. The boundary conditions for Eq.(5.1) and Eq.(5.2) are:

$$v(0,t) = v_1(t), \qquad v(l,t) = v_2(t) \tag{5.3}$$

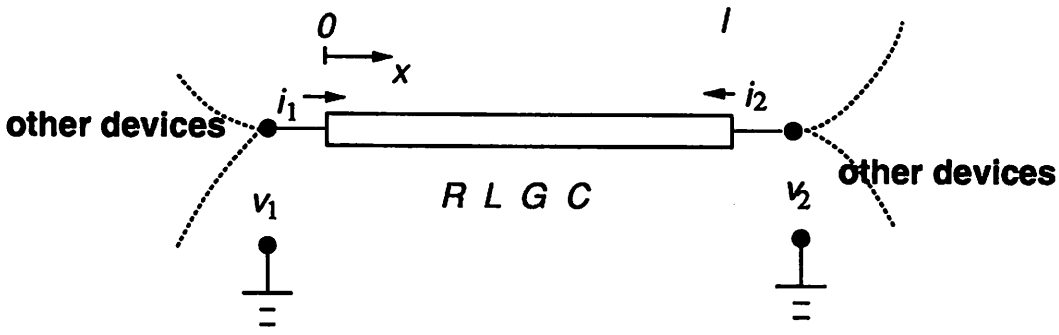


Figure 5.1: A transmission line.

$$i(0,t) = i_1(t), \qquad i(l,t) = -i_2(t) \tag{5.4}$$

In this notation, we use the usual 2-port convention that both $i_1(t)$ and $i_2(t)$ have positive reference direction flowing into the line. By taking the Laplace transforms on Eq.(5.1) and Eq.(5.2), we get

$$\frac{\partial V}{\partial x} = -(sL+R)I \tag{5.5}$$

$$\frac{\partial I}{\partial x} = -(sC+G)V, \tag{5.6}$$

or

$$\frac{\partial^2 V}{\partial x^2} = (sL+R)(sC+G)V \tag{5.7}$$

$$\frac{\partial^2 I}{\partial x^2} = (sL+R)(sC+G)I, \tag{5.8}$$

where $V$ and $I$ denote $V(x,s)$ and $I(x,s)$, the Laplace transformations of $v(x,t)$ and $i(x,t)$.

The solution of Eq.(5.7) and Eq.(5.8) will be of the general form below:

$$V(x,s) = K_1(s)e^{\lambda x} + K_2(s)e^{-\lambda x} \tag{5.9}$$

$$I(x,s) = K_3(s)e^{\lambda x} + K_4(s)e^{-\lambda x}, \tag{5.10}$$

where $\lambda = \lambda(s) \equiv \sqrt{(sL+R)(sC+G)}$. After $K_1$, $K_2$, $K_3$, and $K_4$ are expressed in terms of $V_1$, $V_2$, $I_1$, and $I_2$, the Laplace transforms of $v_1$, $v_2$, $i_1$, and $i_2$, respectively, we get

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{Y_0(s)}{e^{\lambda l} - e^{-\lambda l}} \begin{bmatrix} e^{\lambda l} + e^{-\lambda l} & -2 \\ -2 & e^{\lambda l} + e^{-\lambda l} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}, \tag{5.11}$$

where

$$Y_0(s) \equiv \sqrt{\frac{sC+G}{sL+R}}. \tag{5.12}$$

We call $e^{-\lambda(s)l}$ and $Y_0(s)$ the exponential propagation function and the characteristic admittance function of the line, respectively, which are transcendental functions with finite number of poles.

Each entry of the 2x2 admittance matrix on the right hand side of Eq.(5.11) has an infinite number of poles, the $s$ such that $e^{\lambda(s)l} = e^{-\lambda(s)l}$, which makes it difficult to approximate any entry of that matrix by a finite degree Pade rational function. Papers [41] and [43] expand the entries around $s = 0$ to perform the delay and waveform estimation. We believe this will incur inaccuracy in their results.

To overcome this difficulty, Eq.(5.11) is rearranged [11] to get the following form:

$$Y_0 V_1 - I_1 = e^{-\lambda l}(Y_0 V_2 + I_2) \tag{5.13}$$

$$Y_0 V_2 - I_2 = e^{-\lambda l}(Y_0 V_1 + I_1), \tag{5.14}$$

where the $Y_0$ and the $e^{-\lambda l}$ have no poles on the right half of the $s$ plane; therefore, we can apply the Pade approximation to these two frequency-domain functions, which will be introduced in section 5.2.

The inverse Laplace transforms of Eq.(5.13) and Eq.(5.14) lead to the following

equations for the transmission line in the time domain:

$$v_1(t) * h_1(t) - i_1(t) = v_2(t) * h_3(t) + i_2(t) * h_2(t) \tag{5.15}$$

$$v_2(t) * h_1(t) - i_2(t) = v_1(t) * h_3(t) + i_1(t) * h_2(t), \tag{5.16}$$

where $*$ stands for the convolution operator, and the impulse response functions

$$h_1(t) \equiv \mathcal{L}^{-1}\{Y_0(s)\} \tag{5.17}$$

$$h_2(t) \equiv \mathcal{L}^{-1}\{e^{-\lambda(s)l}\} \tag{5.18}$$

$$h_3(t) \equiv \mathcal{L}^{-1}\{Y_0(s)e^{-\lambda(s)l}\} \tag{5.19}$$

In [11], the authors derived the analytical solutions of $h_1(t)$, $h_2(t)$, and $h_3(t)$ explicitly[1].

At each time point, the integration of a circuit with *RLGC*-lines will involve the convolution of Eq.(5.15) and Eq.(5.16) for each line, where the $v_1$, $v_2$, $i_1$, and $i_2$ at that time point are the only unknown variables to be determined. The circuit equations are composed of two parts: one specified by the KCL equations for each node of the circuit, and the other specified by each line's Eq.(5.15) and Eq.(5.16). The modified nodal analysis is used, whose variables include node voltages, and the $i_1$ and $i_2$ of each line.

Since the convolution integration is time-consuming, we develop our special recursive convolution formulation in section 5.2.

### 5.1.2 Convolution Simulation of Lossy Coupled Lines.

The Telegrapher Equations for a lossy multiconductor system of $N$ lines are:

$$\frac{\partial \mathbf{v}}{\partial x} = -(\boldsymbol{L}\frac{\partial \mathbf{i}}{\partial t} + \boldsymbol{R}\mathbf{i}) \tag{5.20}$$

[1]Note that $Y_0(s)$ stretches to $s = \infty$ and is not periodic; therefore, it is difficult to solve $h_1(t)$ by using inverse Fourier transform.

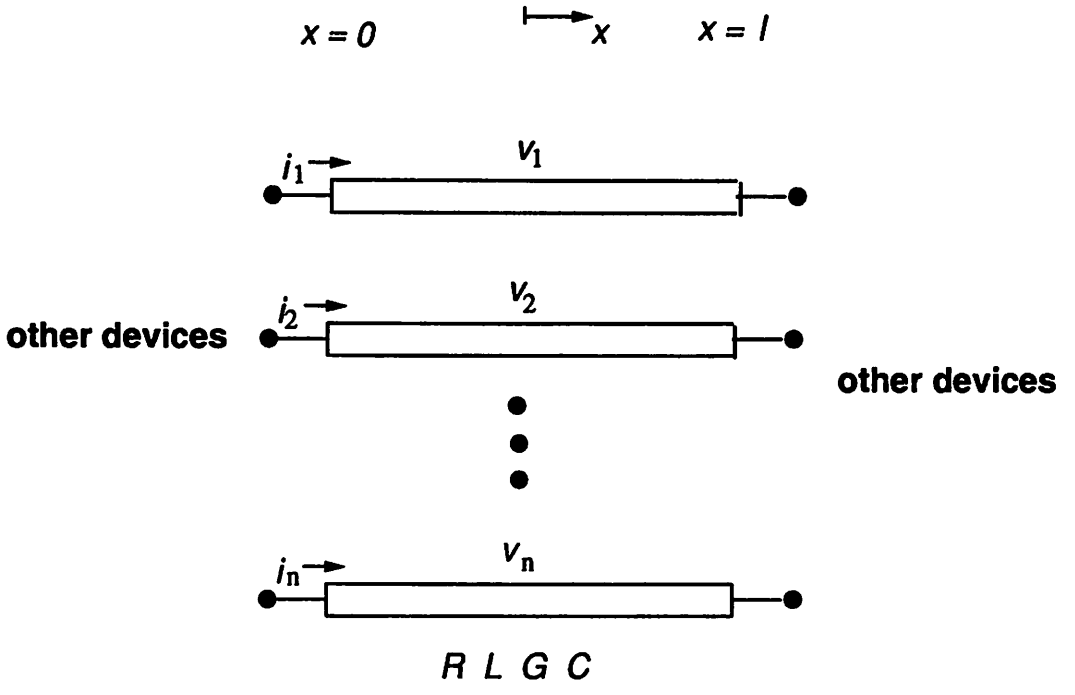


Figure 5.2: An $n$-coupled lossy line system.

$$\frac{\partial \mathbf{i}}{\partial x} = -(\boldsymbol{C}\frac{\partial \mathbf{v}}{\partial t} + \boldsymbol{G}\mathbf{v}) \tag{5.21}$$

The multiconductor lines as shown in Fig. 5.2 stretch from 0 to $l$ on the $x$ coordinate, where $l$ is the length of the lines; each entry of the column vector $\mathbf{v}(x,t)$ gives the voltage of a line at point $x$ at time $t$. Similarly, each entry of the vector $\mathbf{i}(x,t)$ gives the current of a line in the $+x$ direction at point $x$ at time $t$. $\boldsymbol{L}$ and $\boldsymbol{C}$ are the symmetric inductance and capacitance matrices, respectively, of the multiconductor system; their off diagonal components represent the couplings between lines. $\boldsymbol{R}$ and $\boldsymbol{G}$ are diagonal matrices representing loss in the lines.

The boundary conditions for Eq.(5.20) and Eq.(5.21) are:

$$\mathbf{v}(0,t) = \mathbf{v}_1(t), \qquad \mathbf{v}(l,t) = \mathbf{v}_2(t) \tag{5.22}$$

$$\mathbf{i}(0,t) = \mathbf{i}_1(t), \qquad \mathbf{i}(l,t) = -\mathbf{i}_2(t) \tag{5.23}$$

By taking the Laplace transforms of Eq.(5.20) and Eq.(5.21), we get

$$\frac{\partial \mathbf{V}}{\partial x} = -(s\boldsymbol{L} + \boldsymbol{R})\mathbf{I} \tag{5.24}$$

$$\frac{\partial \mathbf{I}}{\partial x} = -(s\boldsymbol{C} + \boldsymbol{G})\mathbf{V}. \tag{5.25}$$

From the above, we obtain the partial differential equations for $\mathbf{V}$ and $\mathbf{I}$.

$$\frac{\partial^2 \mathbf{V}}{\partial x^2} = \boldsymbol{Z}\boldsymbol{Y}\mathbf{V} \tag{5.26}$$

$$\frac{\partial^2 \mathbf{I}}{\partial x^2} = \boldsymbol{Y}\boldsymbol{Z}\mathbf{I}, \tag{5.27}$$

where $\mathbf{V}$ and $\mathbf{I}$ denote $\mathbf{V}(x,s)$ and $\mathbf{I}(x,s)$, the Laplace transformations of $\mathbf{v}(x,t)$ and $\mathbf{i}(x,t)$, respectively. We designate

$$\boldsymbol{Z} = s\boldsymbol{L} + \boldsymbol{R} \tag{5.28}$$

$$\boldsymbol{Y} = s\boldsymbol{C} + \boldsymbol{G}. \tag{5.29}$$

Note that in general $\boldsymbol{Z}\boldsymbol{Y}$ are not equal to $\boldsymbol{Y}\boldsymbol{Z}$ even though the matrices $\boldsymbol{R}$ $\boldsymbol{L}$ $\boldsymbol{G}$ $\boldsymbol{C}$ are all symmetric.

Let us consider the eigenvalue problem below:

$$det\{\gamma_m^2 \boldsymbol{U} - \boldsymbol{Z}\boldsymbol{Y}\} = 0, \tag{5.30}$$

where $\boldsymbol{U}$ is an identity matrix. There will be $N$ eigenvalues $\gamma_m^2, m = 1..N$ with the associated eigenvectors $\mathbf{s}_m, m = 1..N$. Let $\boldsymbol{\Gamma}$ be the diagonal matrix $\{\gamma_1, \gamma_2, \cdots, \gamma_N\}$, which characterizes the wave propagation, and let $\boldsymbol{S}_v$ be the square matrix with the eigenvectors $\mathbf{s}_m$ in the m-th columns. Note $\boldsymbol{\Gamma}$ and $\boldsymbol{S}_v$ are matrix functions of $s$. Let $\boldsymbol{E}(x,s)$ be the diagonal matrix $\{exp(-\gamma_1 x), exp(-\gamma_2 x), \cdots, exp(-\gamma_N x)\}$, which is called the modal function.

Then, the solutions for Eq.(5.26) can be written in terms of the linear combinations of the forward and backward modal functions as

$$\mathbf{V}(x,s) = \boldsymbol{S}_v(s)\boldsymbol{E}(x,s)\mathbf{K_1}(s) + \boldsymbol{S}_v(s)\boldsymbol{E}(x,s)^{-1}\mathbf{K_2}(s). \tag{5.31}$$

By substituting $\mathbf{V}(x,s)$ into Eq.(5.24), we have

$$\mathbf{I}(x,s) = \boldsymbol{S}_i(s)\boldsymbol{E}(x,s)\mathbf{K_1}(s) - \boldsymbol{S}_i(s)\boldsymbol{E}(x,s)^{-1}\mathbf{K_2}(s), \tag{5.32}$$

where

$$\boldsymbol{S}_i(s) = \boldsymbol{Z}^{-1}\boldsymbol{S}_v(s)\boldsymbol{\Gamma}(s). \tag{5.33}$$

After $\mathbf{K_1}$ and $\mathbf{K_2}$ are expressed in terms of $\mathbf{V_1}$, $\mathbf{V_2}$, $\mathbf{I_1}$, and $\mathbf{I_2}$, the Laplace transforms of $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{i_1}$, and $\mathbf{i_2}$, respectively, we get

$$\begin{aligned}\begin{bmatrix}\mathbf{I_1}\\ \mathbf{I_2}\end{bmatrix} &= \boldsymbol{\mathcal{Y}}\begin{bmatrix}\mathbf{V_1}\\ \mathbf{V_2}\end{bmatrix}\\ &= \begin{bmatrix}\boldsymbol{S}_i\boldsymbol{E}_1\boldsymbol{S}_v^{-1} & \boldsymbol{S}_i\boldsymbol{E}_2\boldsymbol{S}_v^{-1}\\ \boldsymbol{S}_i\boldsymbol{E}_2\boldsymbol{S}_v^{-1} & \boldsymbol{S}_i\boldsymbol{E}_1\boldsymbol{S}_v^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{V_1}\\ \mathbf{V_2}\end{bmatrix},\end{aligned} \tag{5.34}$$

where $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ are diagonal matrices:

$$\boldsymbol{E}_1 = diagonal\{\frac{exp(\gamma_m l) + exp(-\gamma_m l)}{exp(\gamma_m l) - exp(-\gamma_m l)}\}, \quad m = 1..N \tag{5.35}$$

$$\boldsymbol{E}_2 = diagonal\{\frac{-2}{exp(\gamma_m l) - exp(-\gamma_m l)}\}, \quad m = 1..N. \tag{5.36}$$

The decoupling technique (or eigenanalysis) above helps us to determine the admittance matrix, $\boldsymbol{\mathcal{Y}}$, of the $N$-port system. The inverse Laplace transform of Eq.(5.34) leads to the following equation for the multiconductor lines in the time domain:

$$\begin{bmatrix}\mathbf{i_1}\\ \mathbf{i_2}\end{bmatrix} = \mathcal{L}^{-1}\{\boldsymbol{\mathcal{Y}}\} * \begin{bmatrix}\mathbf{v_1}\\ \mathbf{v_2}\end{bmatrix}. \tag{5.37}$$

Therefore, at each time point, the integration of a circuit with multiconductor lines will involve the convolution of Eq.(5.37) for the multiconductor lines, where the $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{i_1}$, and $\mathbf{i_2}$ at that time instance are the only unknown variables to be determined by using the modified nodal analysis.

The determination of the inverse Laplace transform of $\mathcal{Y}$ is far from straight forward because of two reasons:

1. Since each entry of the matrices $\boldsymbol{S}_i$, $\boldsymbol{S}_v^{-1}$, $\boldsymbol{E}_1$, and $\boldsymbol{E}_2$ in Eq.(5.34) cannot be expressed as a closed form of $s$, it is impossible to obtain the inverse Laplace transforms of the matrices analytically.
2. Since each entry of the matrix $\mathcal{Y}$ is a transcendental function of $s$ having an infinite number of poles, it is impossible to obtain the inverse Laplace transform of any entry by making use of the inverse Fast Fourier transform.

In contrast, if coupling exists only between adjacent lines and all lines are identical and equally spaced as assumed in [11], $\boldsymbol{S}_i$ and $\boldsymbol{S}_v^{-1}$ will be constant matrices, and each $\gamma_m$ will be a linear function of $s$. Under these assumptions, the authors of [44, 45] derived the analytical solution for the inverse Laplace transform of $\mathcal{Y}$ explicitly.

The approach of [46] developed the scattering parameter technique to evaluate the function values of $\mathcal{Y}$ for $s$ over the whole spectrum and applied the inverse Fast Fourier transform on the data to obtain $\mathcal{L}^{-1}\{\mathcal{Y}\}$. In order to achieve desired accuracy, tremendous amount of data needs to be evaluated, which makes the approach impractical.

In our approach, by employing the Pade approximation, we derive $\mathcal{L}^{-1}\{\mathcal{Y}\}$ without going through inverse Fourier transforms. We rearrange Eq.(5.34) to avoid the problem of

infinite-number-of-poles. These will be discussed in section 5.3, after the introduction of the Pade approximation in the next subsection.

## 5.2 Recursive Convolution Simulation of simple *RLGC*-lines.

### 5.2.1 The Pade approximations of $Y_0(s)$ and $e^{-\lambda(s)l}$.

$$\begin{aligned} Y_0(s) &= \sqrt{\frac{sC+G}{sL+R}} \\ &= \sqrt{\frac{C}{L}}\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}} \end{aligned} \tag{5.38}$$

where $y$ is used to denote $\frac{1}{s}$. $\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}}$ can be expanded into a Maclaurin series of $y$ around $y = 0$, or $s = \infty$. Therefore,

$$\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}} = 1 + m_1 y + m_2 y^2 + \cdots + m_k y^k + \cdots, \tag{5.39}$$

where the $k$-th moment is

$$m_k = \frac{\frac{d^k \sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}}}{dy^k}}{k!}. \tag{5.40}$$

The differentiation of Eq.(5.40) can be evaluated exactly by using symbolic differentiations.

Let us focus for now on the case that the first $2n-1$ moments of $\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}}$ are matched to a Pade approximation with both the numerator polynomial and the denominator polynomial of the same degree $n$, that is

$$\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}} \approx \frac{a_n y^n + a_{n-1} y^{n-1} + \cdots + a_1 y + 1}{b_n y^n + b_{n-1} y^{n-1} + \cdots + b_1 y + 1} \tag{5.41}$$

The reason for that the numerator polynomial and the denominator polynomial are of the same degree is as $y \to \infty$, the value of $\sqrt{\frac{1+\frac{G}{C}y}{1+\frac{R}{L}y}}$ approaches to a finite nonzero number, $\sqrt{\frac{LG}{RC}}$, which also leads to the constraint that $\frac{a_n}{b_n} = \sqrt{\frac{LG}{RC}}$.

$b_i$ can be computed by solving Eq.(5.42) below

$$\begin{bmatrix} 1-\frac{a_n}{b_n} & m_1 & m_2 & \cdots & m_{n-1} \\ m_1 & m_2 & m_3 & \cdots & m_n \\ m_2 & m_3 & m_4 & \cdots & m_{n+1} \\ & & \cdots & & \\ m_{n-1} & m_n & m_{n+1} & \cdots & m_{2n-2} \end{bmatrix} \begin{bmatrix} b_n \\ b_{n-1} \\ b_{n-2} \\ \vdots \\ b_1 \end{bmatrix} = - \begin{bmatrix} m_n \\ m_{n+1} \\ m_{n+2} \\ \vdots \\ m_{2n-1} \end{bmatrix} \tag{5.42}$$

Then, $a_i$ can be computed according to

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \\ m_1 & 1 & 0 & \cdots & \\ m_2 & m_1 & 1 & \cdots & \\ m_3 & m_2 & m_1 & 1 & \cdots \\ \vdots & \vdots & & & \\ m_{n-2} & m_{n-3} & m_{n-4} & \cdots & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \end{bmatrix} + \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-1} \end{bmatrix} \tag{5.43}$$

After considering $y = \frac{1}{s}$, we have

$$\begin{aligned} Y_0(s) &= \sqrt{\frac{sC+G}{sL+R}} \\ &\approx \sqrt{\frac{C}{L}} \frac{a_n y^n + a_{n-1} y^{n-1} + \cdots + a_1 y + 1}{b_n y^n + b_{n-1} y^{n-1} + \cdots + b_1 y + 1} \\ &= \sqrt{\frac{C}{L}} \frac{s^n + a_1 s^{n-1} + a_2 s^{n-2} \cdots + a_n}{s^n + b_1 s^{n-1} + b_2 s^{n-2} \cdots + b_n} \end{aligned} \tag{5.44}$$

Let $P(s)$ denote $s^n + b_1 s^{n-1} + b_2 s^{n-2} \cdots + b_n$ and $Q(s)$ denote $s^n + a_1 s^{n-1} + a_2 s^{n-2} \cdots + a_n$.

Eq.(5.44) is then decomposed by applying the Heaviside's theorem,

$$\begin{aligned} Y_0(s) &\approx \sqrt{\frac{C}{L}}\frac{Q(s)}{P(s)} \\ &= \sqrt{\frac{C}{L}}(1+\sum_{i=1}^{n}\frac{q_i}{s-p_i}), \end{aligned} \tag{5.45}$$

where $p_i's$ are the roots of $P(s)=0$, and

$$q_i = \frac{Q(p_i)-P(p_i)}{\dot{P}(p_i)} \tag{5.46}$$

The Pade approximation helps us to find the inverse Laplace transform of $Y_0(s)$,

$$h_1(t) = \mathcal{L}^{-1}\{Y_0(s)\} \approx \sqrt{\frac{C}{L}}(\delta(t)+\sum_{i=1}^{n} q_i e^{p_i t}), \tag{5.47}$$

where $\delta(t)$ is the Dirac function. The derivation of our recursive convolution formulation is based on this particular form and will be introduced in the next subsection.

The Pade approximation of $e^{-\lambda(s)l}$ is similar.

$$\begin{aligned} \lambda(s) &= \sqrt{(sL+R)(sC+G)} \\ &= s\sqrt{LC}+s\sqrt{LC}\left[\sqrt{(1+\frac{R}{sL})(1+\frac{G}{sC})}-1\right] \\ &= s\sqrt{LC}+s\sqrt{LC}[1+\sum_{i=1}^{\infty}\frac{m_i}{s^i}-1] \end{aligned} \tag{5.48}$$

The moments $m_i$ can be found out by iteratively applying symbolic differentiations on $\sqrt{(1+\frac{R}{L}y)(1+\frac{G}{C}y)}$ with respect to $y$, that is

$$m_k = \frac{\frac{d^k\sqrt{(1+\frac{G}{C}y)(1+\frac{R}{L}y)}}{dy^k}}{k!}. \tag{5.49}$$

From Eq.(5.48) and by letting $\alpha_i$ denote $-\sqrt{LC}m_{i+1}$, we have

$$e^{-\lambda(s)l} = e^{-s\sqrt{LC}l}e^{-m_1\sqrt{LC}l}e^{\sum_{i=1}^{\infty}-\sqrt{LC}l\frac{m_{i+1}}{s^i}} \tag{5.50}$$

Let

$$e^{\sum_{i=1}^{\infty}\frac{\alpha_i}{s^i}} = 1+\sum_{i=1}^{\infty}\frac{\beta_i}{s^i} \tag{5.51}$$

Then,

$$\begin{aligned}
\beta_1 &= \alpha_1 \\
\beta_2 &= \frac{1}{2}(2\alpha_2+\alpha_1\beta_1) = \frac{1}{2}(\alpha_1, 2\alpha_2)(\beta_1, 1) \\
\beta_3 &= \frac{1}{3}(\alpha_1, 2\alpha_2, 3\alpha_3)(\beta_2, \beta_1, 1) \\
&\cdots \\
\beta_k &= \frac{1}{k}(\alpha_1, 2\alpha_2, \cdots, k\alpha_k)(\beta_{k-1}, \beta_{k-2}, \cdots, 1)
\end{aligned} \tag{5.52}$$

Eq.(5.51) is then approximated by a Pade approximation,

$$\begin{aligned}
e^{\sum_{i=1}^{\infty}-\sqrt{LC}\frac{m_{i+1}}{s^i}} &= 1+\sum_{i=1}^{\infty}\frac{\beta_i}{s^i} \\
&\approx \frac{s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_n}{s^n+b_1s^{n-1}+b_2s^{n-2}+\cdots+b_n} \\
&= 1+\sum_{i=1}^{n}\frac{q_i}{s-p_i}.
\end{aligned} \tag{5.53}$$

Here, $\frac{a_n}{b_n}$ is equal to $e^{m_1\sqrt{LC}l}e^{-\sqrt{GR}l}$ by matching the point at $s=0$[2].

Therefore,

$$e^{-\lambda(s)l} = e^{-s\sqrt{LC}l}e^{-m_1\sqrt{LC}l}(1+\sum_{i=1}^{n}\frac{q_i}{s-p_i}). \tag{5.54}$$

The time domain effect of $e^{-s\sqrt{LC}l}$ is the constant delay of $\sqrt{LC}l$. The inverse Laplace transform of $1+\sum_{i=1}^{n}\frac{q_i}{s-p_i}$ is the form that we mentioned, a Dirac function plus the sum of $n$ exponential functions of $t$, based on which we derive our recursive convolution formulation.

[2]The $n$ here may be different from that of Eq.(5.41) for $Y_0(s)$.

The Pade approximation of $Y_0(s)e^{-\lambda(s)l}$ can be computed by multiplying Eq.(5.45) and Eq.(5.54). Since $\frac{q_i}{s-p_i}\frac{q_i'}{s-p_i'} = \frac{q_iq_i'}{p_i-p_i'}(\frac{1}{s-p_i} - \frac{1}{s-p_i'})$, the $p_i's$ of the multiplication result are composed of the $p_i's$ for $Y_0(s)$ and those for $e^{-\lambda(s)l}$. The leading delay term is still $e^{-s\sqrt{LC}l}$.

### 5.2.2 Recursive Convolution Formulation.

Let us look at the convolution integration below at time $t_{n+1}$

$$\begin{aligned} v_1(t) * h_1(t)|_{t=t_{n+1}} &= \int_0^{t_{n+1}} v_1(\tau)h_1(t_{n+1}-\tau)d\tau \\ &= \sqrt{\frac{C}{L}}\int_0^{t_{n+1}} v_1(\tau)[\sum_{i=1}^{n} q_i e^{p_i(t_{n+1}-\tau)} + \delta(t_{n+1}-\tau)]d\tau \\ &= \sqrt{\frac{C}{L}}v_1(t_{n+1}) + \sqrt{\frac{C}{L}}\sum_{i=1}^{n}\int_0^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau \end{aligned} \quad (5.55)$$

Since

$$\begin{aligned} &\int_0^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau \\ =\ &\int_0^{t_n} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau + \int_{t_n}^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau \\ =\ &e^{p_i(t_{n+1}-t_n)}\int_0^{t_n} q_i v_1(\tau)e^{p_i(t_n-\tau)}d\tau + \int_{t_n}^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau \end{aligned} \quad (5.56)$$

we have the recursive relation to obtain $\int_0^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau$ if we have computed $\int_0^{t_n} q_i v_1(\tau)e^{p_i(t_n-\tau)}d\tau$, which is true because the convolution $v_1 * h_1(t_n)$ is evaluated before $t_{n+1}$. Therefore, the convolution integration does not need to expand over all the past history.

By using the Trapezoidal rule for the integration of Eq.(5.56), we have

$$\begin{aligned} &\int_0^{t_{n+1}} q_i v_1(\tau)e^{p_i(t_{n+1}-\tau)}d\tau \\ =\ &e^{p_i h_n}\int_0^{t_n} q_i v_1(\tau)e^{p_i(t_n-\tau)}d\tau + \frac{h_n}{2}q_i[v_1(t_n)e^{p_i h_n} + v_1(t_{n+1})], \end{aligned} \quad (5.57)$$

where $h_n = t_{n+1} - t_n$. Therefore, the only unknown variable left when we compute $v_1 * h_1(t_{n+1})$ is $v_1(t_{n+1})$, which will be determined after we integrate the differential circuit equations from $t_n$ to $t_{n+1}$.

Let us look at the convolution $i_2 * h_2$ of Eq.(5.15), where $h_2(t)$ is $\mathcal{L}^{-1}\{e^{-\lambda(s)l}\}$. Because of the $\sqrt{LC}l$ delay introduced by the leading term $e^{-s\sqrt{LC}l}$ of Eq.(5.54),

$$\begin{aligned} & i_2(t) * h_2(t)|_{t=t_{n+1}} \\ = \; & e^{-m_1\sqrt{LC}l} \int_0^{t_{n+1}} i_2(\tau - \sqrt{LC}l)[\sum_{i=1}^{n} q_i e^{p_i(t_{n+1}-\tau)} + \delta(t_{n+1} - \tau)]d\tau \end{aligned} \tag{5.58}$$

The recursive formulation to compute Eq.(5.58) follows a similar procedure as that of computing $v_1 * h_1(t_{n+1})$, where $v_1(t)$ is replaced by $i_2(t - \sqrt{LC}l)$. $i_2(t_n - \sqrt{LC}l)$ and $i_2(t_{n+1} - \sqrt{LC}l)$ are determined by employing interpolations between previous $i_2$ points. If $t_{n+1} - \sqrt{LC}l$ is less than $t_n$, then both $i_2(t_n - \sqrt{LC}l)$ and $i_2(t_{n+1} - \sqrt{LC}l)$ can be determined before the integration from $t_n$ to $t_{n+1}$; otherwise, $i_2(t_{n+1} - \sqrt{LC}l)$ is represented as an interpolating point between $i_2(t_n)$ and $i_2(t_{n+1})$, and $i_2(t_{n+1})$ is left as an unknown variable to be determined through the integration. A queue is implemented to store the previous $i_2$ data. The data points preceding $t_n - \sqrt{LC}l$ can be deleted from the queue.

As we mentioned in the last subsection $Y_0(s)e^{-\lambda(s)l}$ has the same form as $e^{-\lambda(s)l}$, the convolution of $v_2 * h_3$ of Eq.(5.15) can be computed similarly as we compute $i_2 * h_2$, where $i_2$ is replaced by $v_2$.

Therefore, we can compute the three convolutions of Eq.(5.15) recursively. According to the same discussion, the recursive convolution of Eq.(5.16) follows.

### 5.2.3 Frequency Varying Lines.

Besides *RLGC*-lines, our proposed approach can handle a frequency varying line as long as the line's $Y_0(s)$ and $e^{-\lambda(s)l}$ can be defined or measured. For example, to consider skin effects, the $Y_0(s)$ and $\lambda(s)$ of a line can be modeled by [47]

$$Y_0(s) = \sqrt{\frac{sC}{R + sL + K\sqrt{s}}} \tag{5.59}$$

$$\lambda(s) = \sqrt{sC(R + sL + K\sqrt{s})} \tag{5.60}$$

When the information of Eq.(5.59) and Eq.(5.60) are given, we can proceed with our approach: symbolic differentiations on Eq.(5.59) and Eq.(5.60) to get the moments, Pade approximations, Heaviside's decompositions, and recursive convolutions.

If the equations of a line's $Y_0(s)$ and $e^{-\lambda(s)l}$ are not available, we can measure them from experiments and then use polynomials to fit the measured data.

## 5.3 Recursive Convolution Simulation of Lossy Coupled Lines.

### 5.3.1 Computing Time-Domain Response Using Pade Approximation and Polynomial Fit.

Since each entry of the admittance matrix of Eq.(5.34) is a transcendental function with an infinite number of poles. We rearrange Eq.(5.34) in the following form:

$$\boldsymbol{S_i}\boldsymbol{S_v}^{-1}\mathbf{V_1} - \mathbf{I_1} = \boldsymbol{S_i}\boldsymbol{E}(l)\boldsymbol{S_v}^{-1}\mathbf{V_2} + \boldsymbol{S_i}\boldsymbol{E}(l)\boldsymbol{S_i}^{-1}\mathbf{I_2} \tag{5.61}$$

$$\boldsymbol{S_i}\boldsymbol{S_v}^{-1}\mathbf{V_2} - \mathbf{I_2} = \boldsymbol{S_i}\boldsymbol{E}(l)\boldsymbol{S_v}^{-1}\mathbf{V_1} + \boldsymbol{S_i}\boldsymbol{E}(l)\boldsymbol{S_i}^{-1}\mathbf{I_1}, \tag{5.62}$$

where[3] $\boldsymbol{S}_i$, $\boldsymbol{E}(l)$, and $\boldsymbol{S}_v^{-1}$ have finite numbers of poles, which will be in the left half of the $s$ plane; therefore, we can apply the Pade approximation to the entries of these three frequency-domain matrices.

The $k$-th order Pade approximation of $\boldsymbol{S}_i$, $\boldsymbol{E}(l)$, or $\boldsymbol{S}_v^{-1}$ will match the original matrix at $k+1$ different $s$ near $s=\infty$; therefore, to determine the Pade approximations of $\boldsymbol{S}_i$, $\boldsymbol{E}(l)$, and $\boldsymbol{S}_v^{-1}$ we only need to evaluate them at $k+1$ different large $s$ points. From the $k+1$ points, a fitted polynomial for each entry of the matrices can be determined[4]. A Pade approximation is then applied to the polynomial. The whole procedure is as follows.

Let $y$ denote $\frac{1}{s}$. Then,

$$\begin{aligned} & (s\boldsymbol{L}+\boldsymbol{R})(s\boldsymbol{C}+\boldsymbol{G}) \\ = \; & s^2(\boldsymbol{L}+y\boldsymbol{R})(\boldsymbol{C}+y\boldsymbol{G}) \\ = \; & s^2\boldsymbol{K}(y), \end{aligned} \tag{5.63}$$

where the matrix $\boldsymbol{K}(y)$ denotes the product of $\boldsymbol{L}+y\boldsymbol{R}$ and $\boldsymbol{C}+y\boldsymbol{G}$. For each $y$, we can diagonalize it[5] as

$$\boldsymbol{K}(y) = \boldsymbol{S}_v(y)\boldsymbol{\Lambda}^2(y)\boldsymbol{S}_v(y)^{-1}. \tag{5.64}$$

And, similar to Eq.(5.33) $\boldsymbol{S}_i(y)$ can be computed according to

$$\boldsymbol{S}_i(y) = (\boldsymbol{L}+y\boldsymbol{R})^{-1}\boldsymbol{S}_v(y)\boldsymbol{\Lambda}(y). \tag{5.65}$$

The fitted polynomial around $y=0$ will be the fitted polynomial around $s=\infty$. Hence, the first $\boldsymbol{S}_i$, $\boldsymbol{E}(l)$, and $\boldsymbol{S}_v^{-1}$ point to be computed should be at $y=0$. The rest $k$

[3] The $E(l)$ is the modal function $E(l,s)$ of Eq.(5.12) and Eq.(5.13).

[4] Here, we made the assumption that the eigenvectors will be continuous with respect to $s$. This assumption is true for multiconductor line systems and will be discussed in Appendix G.

[5] $K(y)$ is diagonalizable, which will be shown in Appendix G.

points will be at the $y$ equal to the multiples of a fraction of the smallest entry of $\boldsymbol{\Lambda}(0)$. Practically, the smallest entry of $\boldsymbol{\Lambda}(0)$ is a small enough value within the neighborhood of $y = 0$. We construct polynomials to fit the set of $(y, \boldsymbol{S}_i(y))$ points, and, similarly, for $(y, \boldsymbol{\Lambda}^2(y))$ and $(y, \boldsymbol{S}_v{}^{-1}(y))$ points. All the entries of $\boldsymbol{S}_i$, $\boldsymbol{\Lambda}^2$ and $\boldsymbol{S}_v{}^{-1}$ will be $k$-th degree polynomials of $y$. We can perform $\boldsymbol{S}_i\boldsymbol{S}_v{}^{-1}$ by using polynomial arithmetics (polynomial multiplication and addition). Each entry of the product $\boldsymbol{S}_i\boldsymbol{S}_v{}^{-1}$ will be a $2k$-th degree polynomial of $y$ (or $\frac{1}{s}$). To perform the $\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_v{}^{-1}$ and $\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_i{}^{-1}$ of Eq.(5.61) and Eq.(5.62), we need to first solve the diagonal matrix $\boldsymbol{E}(l)$.

The $m$-th diagonal entry of $\boldsymbol{\Lambda}$ will be of the form[6][7]:

$$\begin{aligned} \lambda_m &= \sqrt{c_0 + c_1 y + \cdots + c_k y^k} \\ &= m_0 + m_1 y + m_2 y^2 + \cdots + m_k y^k + \cdots, \end{aligned} \tag{5.66}$$

where the $i$-th moment is

$$m_i = \frac{\frac{d^i \lambda_m}{dy^i}}{i!}. \tag{5.67}$$

The differentiation of Eq.(5.67) can be evaluated exactly by using symbolic differentiations.

In Eq.(5.63) $s^2$ has been extracted out front, so

$$\begin{aligned} \gamma_m &= s\lambda_m \\ &= m_0 s + m_1 + m_2 y + \cdots + m_k y^{k-1} + m_{k+1} y^k + \cdots \end{aligned} \tag{5.68}$$

By letting $\alpha_i$ denote $-m_{i+1} l$, we have the $m$-th entry of $\boldsymbol{E}(l)$

$$e^{-\gamma_m(s)l} = e^{-sm_0 l} e^{-m_1 l} e^{\sum_{i=1}^{\infty} -m_{i+1} l y^i}$$

[6]Note that each entry of $\Lambda^2$ is a $k$-th degree polynomial of $y$.

[7]The entries of $\Lambda^2(0)$ (the $c_0$ of Eq.(5.66)) will be positive. This property is important because the $\lambda_m$ of Eq.(5.66) will be expanded around $y = 0$. If $c_0$ is not positive, we can not perform this expansion. We will show the property in Appendix G.

$$= e^{-sm_0 l} e^{-m_1 l} e^{\sum_{i=1}^{\infty} \alpha_i y^i}. \tag{5.69}$$

Let

$$e^{\sum_{i=1}^{\infty} \alpha_i y^i} = 1 + \sum_{i=1}^{\infty} \beta_i y^i. \tag{5.70}$$

The $\beta's$ can be determined as before given in Eq.(5.52).

After truncating the summation of Eq.(5.70) to the $k$-th term, we have

$$e^{-\gamma_m(s)l} = e^{-sm_0 l} e^{-m_1 l}(1 + \beta_1 y + \beta_2 y^2 + \cdots + \beta_k y^k); \tag{5.71}$$

therefore, each diagonal entry of $\boldsymbol{E}(l)$ will be a $k$-th degree polynomial of $y$ (or $\frac{1}{s}$) times an exponential function.

We are able to perform $\boldsymbol{S_i E}(l)\boldsymbol{S_v}^{-1}$ and $\boldsymbol{S_i E}(l)\boldsymbol{S_i}^{-1}$ by using polynomial multiplications and additions. Unlike $\boldsymbol{S_i}$ or $\boldsymbol{S_v}^{-1}$, there are exponential terms in $\boldsymbol{E}(l)$. The polynomials multiplied by different exponential terms cannot be added together. Therefore, each entry of $\boldsymbol{S_i E}(l)\boldsymbol{S_v}^{-1}$ or $\boldsymbol{S_i E}(l)\boldsymbol{S_i}^{-1}$ will be a sum of $N$ exponential-times-polynomial terms. Each exponential is associated with one of the $N$ modes. The polynomials are then approximated by Pade approximations. After the approximation, each entry of $\boldsymbol{S_i E}(l)\boldsymbol{S_v}^{-1}$ and $\boldsymbol{S_i E}(l)\boldsymbol{S_i}^{-1}$ will be of the form below

$$\sum_{j=1}^{N} e^{-sm_0(j)l} K_j (1 + \sum_{i=1}^{n} \frac{q_i(j)}{s - p_i(j)}), \tag{5.72}$$

where $(j)$ is the index of the $j$-th mode, $m_0(j)$ is the $m_0$ of the $j$-th mode, and $K_j$ is a constant. The time domain effect of $e^{-sm_0(j)l}$ is the constant delay of $m_0(j)l$. The inverse Laplace transform of $1 + \sum_{i=1}^{n} \frac{q_i(j)}{s-p_i(j)}$ is of the form that we mentioned in section 5.2, a Dirac function plus the sum of $n$ exponential functions of $t$.

### 5.3.2 Recursive Convolution of Multiconductor Lines.

The $i$-th entry of the convolution of $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{S}_v{}^{-1}\} * \mathbf{v}$ is equal to

$$\sum_{j=1}^{N} \mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{S}_v{}^{-1}\}_{ij} * v_j, \tag{5.73}$$

where $\mathbf{v}$ can be either $\mathbf{v_1}$ as for Eq.(5.61) or $\mathbf{v_2}$ as for Eq.(5.62). The subscript $ij$ is to denote the the $ij$-th entry of the matrix and the subscript $j$ is to denote the $j$-th entry of the vector. Since every entry of $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{S}_v{}^{-1}\}$ is of the same form, a Dirac function plus the sum of $n$ exponential functions of $t$, let us only consider the $ij$-th one, denoted by $h(t)$.

At time $t_{n+1}$ the convolution integration

$$\begin{aligned} v_j(t) * h(t)|_{t=t_{n+1}} &= \int_0^{t_{n+1}} v_j(\tau)h(t_{n+1}-\tau)d\tau \\ &= \int_0^{t_{n+1}} v_j(\tau)K[\sum_{i=1}^{n} q_i e^{p_i(t_{n+1}-\tau)} + \delta(t_{n+1}-\tau)]d\tau \\ &= Kv_1(t_{n+1}) + K\sum_{i=1}^{n}\int_0^{t_{n+1}} q_i v_j(\tau)e^{p_i(t_{n+1}-\tau)}d\tau, \end{aligned} \tag{5.74}$$

where $K$ is the leading constant of $h(t)$, can be computed recursively following the discussion of section 5.2.2.

For the convolution of $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_v{}^{-1}\}_{ij} * v_j$, since $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_v{}^{-1}\}_{ij}$ is a summation of $N$ similar terms as described in Eq.(5.72), let us consider only one of them, denoted by $h(t)$.

Because of the $m_0 l$ delay introduced by the exponential term of Eq.(5.72)

$$\begin{aligned} & v_j(t) * h(t)|_{t=t_{n+1}} \\ = \; & K_j \int_0^{t_{n+1}} v_j(\tau - m_0 l)[\sum_{i=1}^{n} q_i e^{p_i(t_{n+1}-\tau)} + \delta(t_{n+1}-\tau)]d\tau \end{aligned} \tag{5.75}$$

The recursive formulation to compute Eq.(5.75) follows a similar procedure as that of Eq.(5.58).

The computation of the convolution $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_i^{-1}\} * \mathbf{i}$ is the same as that of $\mathcal{L}^{-1}\{\boldsymbol{S}_i\boldsymbol{E}(l)\boldsymbol{S}_v^{-1}\} * \mathbf{v}$; therefore, we can compute the convolutions of Eq.(5.61) and Eq.(5.62) recursively.

### 5.3.3 Frequency-Varying Coupled Lines.

To consider the frequency-varying effects, the $\boldsymbol{R}$ $\boldsymbol{L}$ $\boldsymbol{G}$ $\boldsymbol{C}$ matrices of the lossy multiconductor lines will be $\boldsymbol{R}$(s) $\boldsymbol{L}$(s) $\boldsymbol{G}$(s) $\boldsymbol{C}$(s), functions of frequency. In this case, when we evaluate the $\boldsymbol{K}(y)$ matrix of Eq.(5.63), we need to use the $\boldsymbol{R}$ $\boldsymbol{L}$ $\boldsymbol{G}$ $\boldsymbol{C}$ evaluated at the frequency $s = \frac{1}{y}$. The rest follows exactly the same derivations as discussed in subsection 5.3.1. For frequency-varying simple lines, the approach introduced in subsection 5.2.3 is preferred because that approach has better numerical stability. The moments of the suggested model can be evaluated exactly using symbolic differentiations, which guarantees all the poles will be on the left half of the $s$ plane.

## 5.4 Implementation in a Stepwise Equivalent Conductance Circuit Simulator.

In chapter 3, we have discussed that nonlinear circuits can be treated as linear circuits composed of time-varying conductors, and for the integration of a time step, say from $t_n$ to $t_{n+1}$, the time-varying conductances can be replaced by their middle values within $(t_n, t_{n+1})$ interval to achieve a second order of accuracy. The middle conductance is equal to the conductance at $t_n$ plus $\frac{t_{n+1}-t_n}{2}$ times the time derivative of the conductance at $t_n$. After the substitutions of the effective constant conductances, the original nonlinear

circuit is transformed into a linear time-invariant circuit; therefore, the implicit integration does not involve solving any nonlinear equation.

When simulating nonlinear circuits with lossy multiconductor transmission lines, we employ the same technique. The nonlinear circuit is treated as a linear time-varying circuit. For each implicit integration, we replace every time-varying conductor by a time-invariant conductor of its effective conductance; therefore, we end up integrating a linear circuit with transmission lines. A modified nodal analysis is used to solve the voltage at each node and the current of each line. We have incorporated the transmission line simulation into SWEC [4, 5]. In section 5.6, we will show the experimental results.

### 5.4.1 Circuit Partition.

In previous simulation works, very often transmission lines were either neglected or modeled as lump *RLC* elements when accuracy was traded for efficiency. By using our approach, which considers the transmission line, we gain another advantage in circuit partitioning. Furthermore, we can achieve a considerable speedup on the overall simulation. A circuit can be partitioned at the transmission lines; therefore, a large circuit can be decomposed into small subcircuits.

Let us consider the situation that the node 1 and the node 2 of Eq.(5.15) or Eq.(5.16) belong to different subcircuits. Then, Eq.(5.15) is the convolution equation which will be considered when we integrate the subcircuit containing node 1, and Eq.(5.16) will be considered when the subcircuit containing node 2 is integrated. Because of the $\sqrt{LC}l$ delay introduced by $h_2(t)$ and $h_3(t)$, the subcircuit containing node 1 can be integrated, say from its $t_n$ to its $t_{n+1}$, independently from the subcircuit containing node 2 if the

subcircuit containing node 2 has already been evaluated beyond $t_{n+1} - \sqrt{LC}l$, which means $v_2(t_{n+1}-\sqrt{LC}l)$ and $i_2(t_{n+1}-\sqrt{LC}l)$ were available. It is similar to the subcircuit containing node 2. In order to do the integration on each subcircuit independently, we need to place the upper bound, $\sqrt{LC}l$, on the time steps of the two subcircuits. Therefore, whenever we make a cut on a transmission line, we place the upper bound of the line's $\sqrt{LC}l$ value on the time steps of the two subcircuits connected by the line. This technique is applicable to both digital and analog circuits, including MOS and bipolar circuits. Since some lines would have very small $\sqrt{LC}l$ values, causing very small time steps and degrading the efficiency, we should avoid making cuts on those lines.

Similarly, we can partition the circuit on each multiconductor lines and integrate the subcircuit containing the group of nodes in side 1 of Eq.(5.61) or Eq.(5.62) independently of the the subcircuit containing the group of nodes in side 2. The upper bound of the time steps placed on the two subcircuits will be the smallest modal frequency of $\boldsymbol{\Lambda}(0)$.

## 5.5 Error Analysis of the Pade Approximation.

In this section, we are going to analyze the errors introduced by these approximations and to determine the necessary order for the approximations.

Let us consider a frequency domain function $H(s)$ which has no poles on the right half plane or at $s = 0$. $H(s)$ represents any transfer function, such as $Y_0(s)$. It can be expanded into a series of $\frac{1}{s}$ around $s = \infty$, which is

$$H(s) = m_0 + m_1\frac{1}{s} + m_2\frac{1}{s^2} + \cdots + m_k\frac{1}{s^k} + m_{k+1}\frac{1}{s^{k+1}} + m_{k+2}\frac{1}{s^{k+2}} + \cdots \qquad (5.76)$$

Let $H_P(s)$ be the Pade approximation of $H(s)$ which, when expanded into a series of $\frac{1}{s}$,

matches the $m_i's$ of $H(s)$ up to $m_k$. This means

$$H_P(s) = m_0 + m_1\frac{1}{s} + m_2\frac{1}{s^2} + \cdots + m_k\frac{1}{s^k} + m'_{k+1}\frac{1}{s^{k+1}} + m'_{k+2}\frac{1}{s^{k+2}} + \cdots \tag{5.77}$$

We say $H_P(s)$ is a $k$-th order approximation of $H(s)$ at $s = \infty$[8]. Let $E(s)$ be the difference of the two, which is the frequency domain approximation error. So

$$\begin{aligned} E(s) &= H(s) - H_P(s) \\ &= (m_{k+1} - m'_{k+1})\frac{1}{s^{k+1}} + (m_{k+2} - m'_{k+2})\frac{1}{s^{k+2}} + \cdots \end{aligned} \tag{5.78}$$

Let

$$h(t) \equiv \mathcal{L}^{-1}\{H(s)\} \tag{5.79}$$

$$h_P(t) \equiv \mathcal{L}^{-1}\{H_P(s)\} \tag{5.80}$$

$$e(t) \equiv \mathcal{L}^{-1}\{E(s)\} = h(t) - h_P(t) \tag{5.81}$$

From the initial value theorem and Eq.(5.78),

$$\lim_{t\to 0} e(t) = \lim_{s\to\infty} sE(s) = 0 \tag{5.82}$$

Furthermore, since the Laplace transform of the $i$-th derivative of $e(t)$, $e^{(i)}(t)$, is $s^iE(s)$, we have

$$\lim_{t\to 0} e^{(i)}(t) = \lim_{s\to\infty} s^{i+1}E(s) = 0 \qquad \forall\, i = 1..k-1. \tag{5.83}$$

Therefore, around $t = 0$, $h_P(t)$ matches $h(t)$ up to the $k-1$-th derivative. This property is important to assure the accuracy when $h_P(t)$ is used to evaluate the convolution integration

[8]Note that for our convolution approach, $k$ should be an odd number. The degree of the numerator polynomial and that of the denominator polynomial are both equal to $\frac{k+1}{2}$.

which originally involves $h(t)$. Let us consider how we compute the convolution integration $h * v(t_n)$: (1) $h(t)$ is flipped around $t = 0$ and, then, shifted to $t_n$, and (2) we integrate from $t = 0$ to $t = t_n$ the values of $v(t)$ times the flipped and shifted $h$. Hence, the values of $h$ around $t = 0$ are crucial; therefore, $h_P(t)$ should match $h(t)$ around $t = 0$ in order to achieve accurate convolution results. Since the first $k$ derivatives of $e(t)$ vanished at $t = 0$, we can expect that $h_P(t)$ matches $h(t)$ for an interval beyond $t = 0$. As $k$ increases, the interval will increase. If the simulation time falls into this interval, no error on the convolution results will be introduced by the Pade approximation. If the simulation time is small, a small $k$ will guarantee the accuracy when Pade approximations are applied to the transfer function. Hence, the necessary order for the approximation depends on the simulation time.

The above discussion justifies our approach to expand $H(s)$ into a series of $\frac{1}{s}$ around $s = \infty$ instead of a series of $s$ around $s = 0$. The Pade approximation $H_P(s)$ should match the $\frac{1}{s}$ series in order to achieve accurate convolution results. This is different from the work in [24], [41], and [42], which is based on the expansion at $s = 0$. In order to highlight this point further, in the next section, we will also show the simulation results based on the Pade approximations of the $s$ expansions of $Y_0(s)$ and $e^{-\lambda(s)l}$ in the convolutions. Note that there is no need to approx $H(s)$ over the whole spectrum. The correct approximation of $H(s)$ around $s = 0$ will guarantee the correct steady state result of the approximate impulse response; however, we will never use that part of the impulse response during the simulation.

Since the $H_P(s)$ in our Pade approximation also matches the original $H(s)$ at

$s = 0$, from[9] the final value theorem we have

$$\lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = 0 \tag{5.84}$$

$$\lim_{t\to\infty} \int_0^t e(\tau)d\tau = \lim_{s\to 0} s\frac{E(s)}{s} = 0. \tag{5.85}$$

Eq.(5.84) and Eq.(5.85) will have the effects that the absolute value of $e(t)$ will not increase very fast beyond the initial zero interval, which is important for the situation where we cannot afford to pursue higher order approximations. Most importantly, the match of $H(s)$ at $s = 0$ makes it possible to use large time steps whenever the simulation is in steady states. This can be shown as follows. If a large time step $h_n = t_{n+1} - t_n$ is used for integration, then the $e^{p_i(t_{n+1}-t_n)} \int_0^{t_n} q_i v_1(\tau) e^{p_i(t_n-\tau)} d\tau$ term of Eq.(5.56) will vanish because $p_i h_n$ has a large negative real part. Since the simulation is in a steady state, i.e. $v_1(\tau)$ of Eq.(5.56) is a constant, say $v_1^0$, the integration of Eq.(5.56) becomes

$$\begin{aligned}
&\int_0^{t_{n+1}} q_i v_1(\tau) e^{p_i(t_{n+1}-\tau)} d\tau \\
&= \int_{t_n}^{t_{n+1}} q_i v_1(\tau) e^{p_i(t_{n+1}-\tau)} d\tau \\
&= v_1^0 \int_0^{h_n} q_i e^{p_i\tau} d\tau \\
&= -v_1^0 \frac{q_i}{p_i} \\
&= v_1^0 \frac{q_i}{s - p_i}|_{s=0}.
\end{aligned} \tag{5.86}$$

From Eq.(5.86) and due to the match $H_P(s = 0) = H(s = 0)$, we have accurate convolution results of Eq.(5.15) and Eq.(5.16) at $t_{n+1}$ while using the Pade approximations and large time steps.

[9] Refer the paragraph before Eq.(5.42).

Even though $H(s)$ does not have poles in the right half plane or the origin, the Pade approximation may lead to right half plane poles. In that case we need to increase the order of the approximation. Pade approximations do not guarantee all the poles will be on the left half of the s plane; however, as the approximation becomes accurate, the poles will be pushed to the left half plane. The other fact in determining the necessary order for a Pade approximation is that the maximum errors in the time domain, i.e. $h(t) - h_P(t)$, should be less than a pre-specified error bound. After neglecting higher order terms, we have

$$E(s) \approx (m_{k+1} - m'_{k+1}) \frac{1}{s^{k+1}}, \tag{5.87}$$

and,

$$e(t) \approx (m_{k+1} - m'_{k+1}) \frac{t^k}{k!}. \tag{5.88}$$

Therefore, the absolute value of $e(t)$ is a monotonic function of $t$, which will have its maximum $|(m_{k+1} - m'_{k+1}) \frac{T_s^k}{k!}|$ at $T_s$, the ending simulation time. We want it to be less than an upper bound, say $\epsilon$, that is

$$(m_{k+1} - m'_{k+1}) \frac{T_s^k}{k!} \leq \epsilon \tag{5.89}$$

We need to increase the order $k$ until the condition of Eq.(5.89) is satisfied.

For our convolution approach, there are two $\epsilon's$, one for the approximation of $Y_0(s)$ and one for that of $e^{-\lambda(s)l}$. Those two $\epsilon's$ can be tuned to achieve the best trade-off between accuracy and efficiency.

Singhal and Vlach have developed the error analysis for their numerical inverse Laplace transform approach based on the Pade approximation of $e^{st}$ around $s = 0$ [48]. Our

error analysis shows the expansion of $H(s)$ around $s = \infty$ in addition to the match at $s = 0$ will guarantee the correct convolution results, which is the major difference.

## 5.6 Experimental Results.

### 5.6.1 Lossy Simple Lines.

We compared the simulation results of SWEC with the results of SPICE3.e, which implemented the work of [11]. [11] solves the inverse Laplace transforms of $Y_0(s)$ and $e^{-\lambda(s)l}$ analytically; therefore, we can assume the simulation results of SPICE3.e is correct. Since SPICE3.e can not allow leakage conductances, $G$ is set to zero for all the examples. The first test circuit is a *RLGC*-line driven by a CMOS inverter and terminated in a capacitor. Fig. 5.3 shows the result at the near end. The first two rows of Table 1 summarize the CPU times for the case with 3 input vectors and that with 24 input vectors. The 24-vector case simply repeats the 3-vector case eight times. The CPU time increase for SWEC to finish the two was four times, while 30 times for SPICE3.e. Fig. 5.4 shows the far end result of the 24-vector case. The waveforms from the two simulators at larger simulation times were departing from each other, which confirms the discussion of the previous section. We did not observe much accuracy degradation although the simulation time increased dramatically. A fifth order Pade approximation was used for the simulation.

The second circuit was a line driven by a CMOS 2 input NAND gate. The inputs to the NAND gate were used to create a glitch on purpose at the output; therefore, we could observe more transient information. Fig. 5.5 shows the result at the near end. The third row of Table 1 summarizes the CPU times.

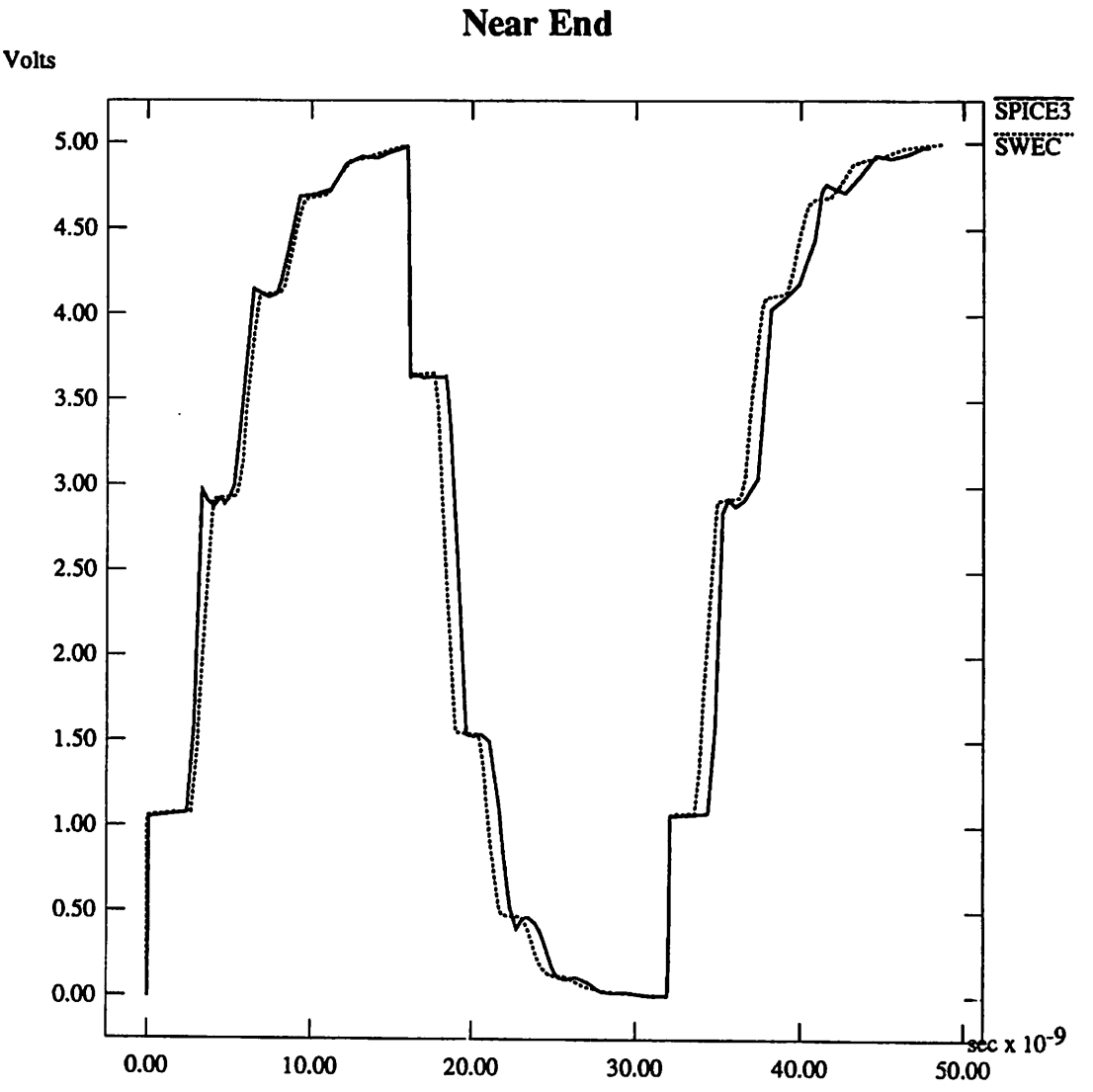


Figure 5.3: A line driven by an inverter (3 input vectors).

| Experiment | SPICE3.e | SWEC |
|---|---|---|
| INV with 3 vec. | 2.7 s | 0.3 s |
| INV with 24 vec. | 79.5 s | 1.2 s |
| NAND | 4.0 s | 0.4 s |

Table 5.1: CPU times for Single Line Simulations.

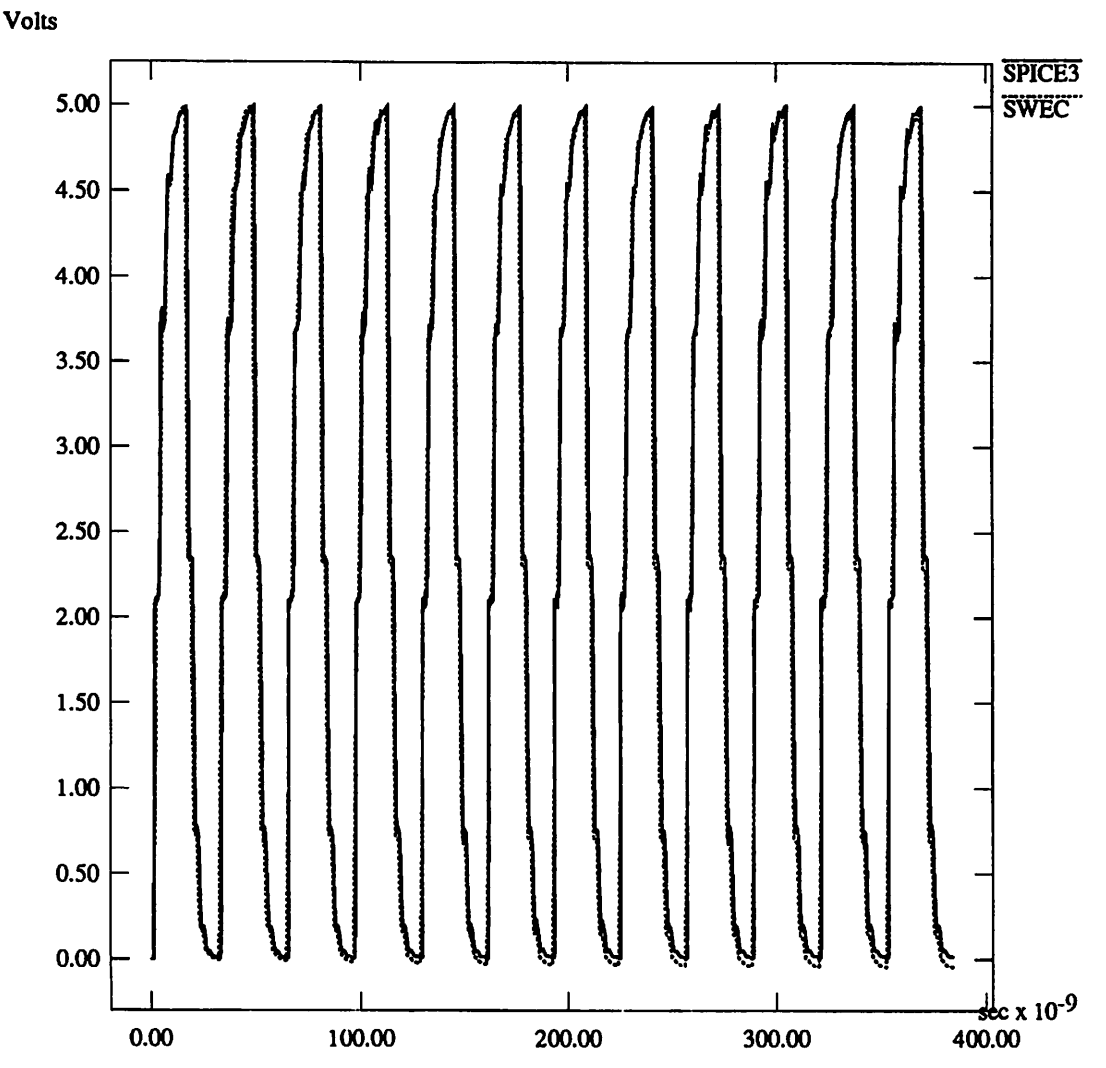


Figure 5.4: A line driven by an inverter (24 input vectors).

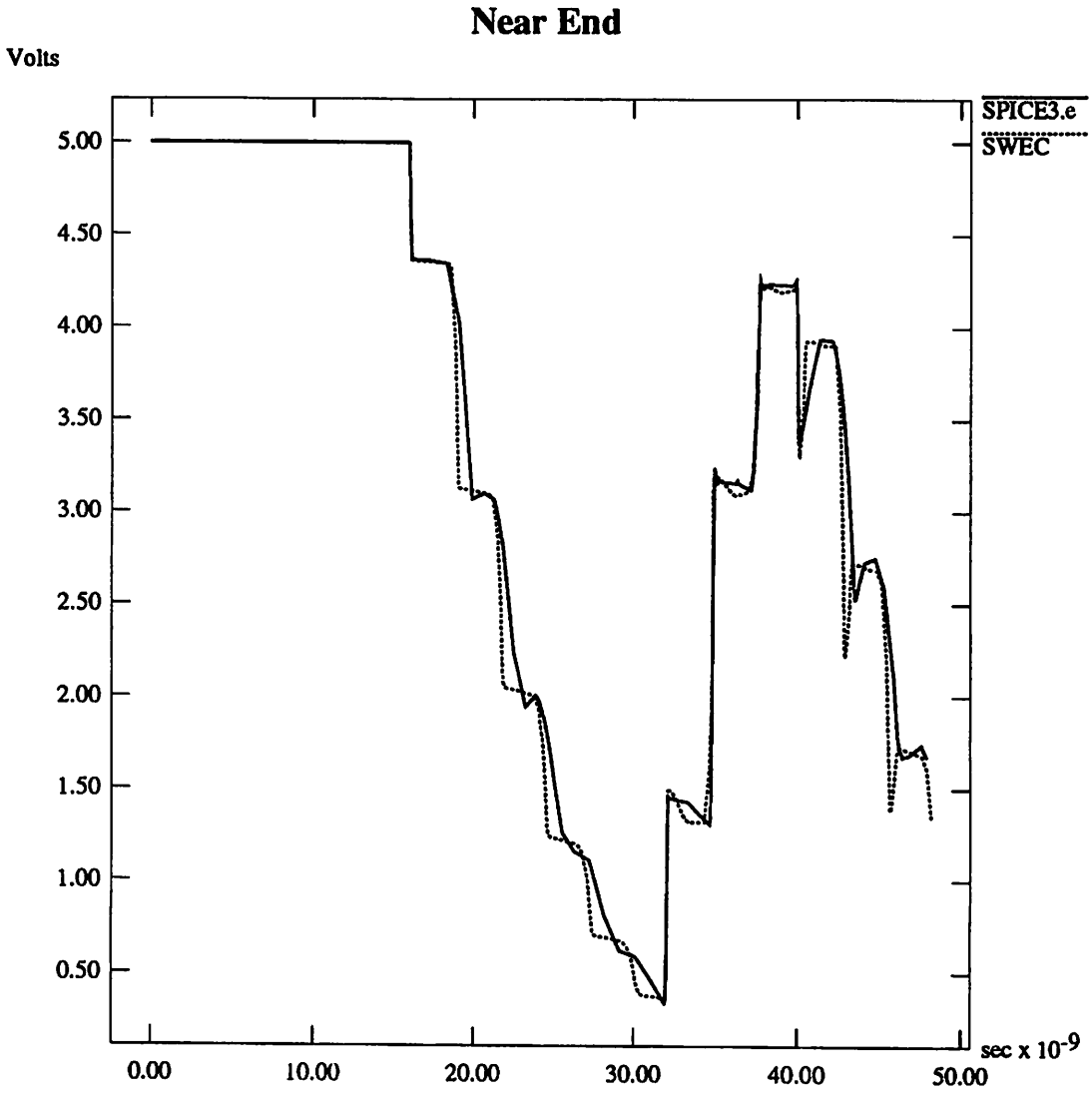


Figure 5.5: A line driven by a NAND.

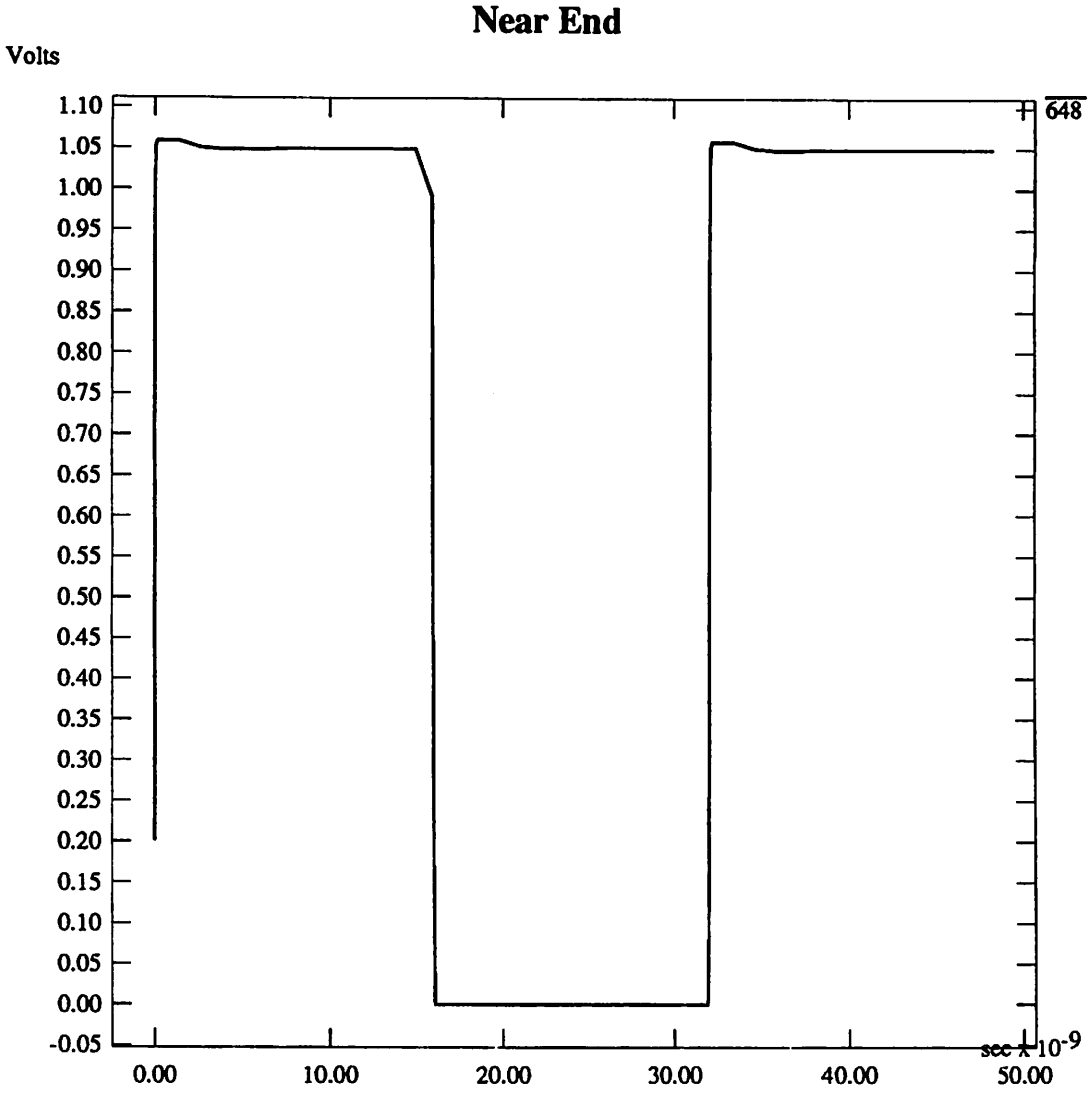


Figure 5.6: Expansion in $s$.

To highlight that the expansions of $Y_0(s)$ and $e^{-\lambda(s)l}$ into series of $s$ are not appropriate for simulation, we also implement a similar simulation program as SWEC except it uses the $s$ expansions rather than $\frac{1}{s}$. We simulated the inverter circuit using this simulator. Fig. 5.6 shows the near end result, which, in comparison with Fig. 5.3, does not contain any high frequency information.

### 5.6.2 Lossy Coupled Lines.

We compared the simulation results of SWEC with the results of SPICE3.e, which implemented the work of [11]. Since SPICE3.e can only consider the multiconductor lines (1) whose couplings exist only between adjacent lines, (2) which are identical and equally spaced, and (3) which have no leakage conductances, the simulated circuit is picked according to these assumptions. The simulated circuit, shown in Fig. 5.7, has multiconductor four lines, whose parameters are:

$$\boldsymbol{R}=\begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0.3 \end{bmatrix} \quad \boldsymbol{L}=\begin{bmatrix} 9e-9 & 5.4e-9 & 0 & 0 \\ 5.4e-9 & 9e-9 & 5.4e-9 & 0 \\ 0 & 5.4e-9 & 9e-9 & 5.4e-9 \\ 0 & 0 & 5.4e-9 & 9e-9 \end{bmatrix}$$

$$\boldsymbol{G}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{C}=\begin{bmatrix} 3.5e-13 & -3e-14 & 0 & 0 \\ -3e-14 & 3.5e-13 & -3e-14 & 0 \\ 0 & -3e-14 & 3.5e-13 & -3e-14 \\ 0 & 0 & -3e-14 & 3.5e-13 \end{bmatrix}. \tag{5.90}$$

Fig. 5.8 shows the result at node 5 and Fig. 5.9 shows the results at node 11. The rings on Fig. 5.9 are caused by the crosstalk, since node 3 has been kept to 5 volts all the time, the waveform at node 11 would be zero all the time if there is no coupling. We can observe that the waveforms from the two simulators matched perfectly. For the simulation, there are three input vectors at node 2 and node 4, respectively. The first row of Table 2 summarizes the CPU times for this simulation. The simulations were run on a DEC station 5000. SWEC is about 40 times faster. For the same circuit, we also ran the simulation

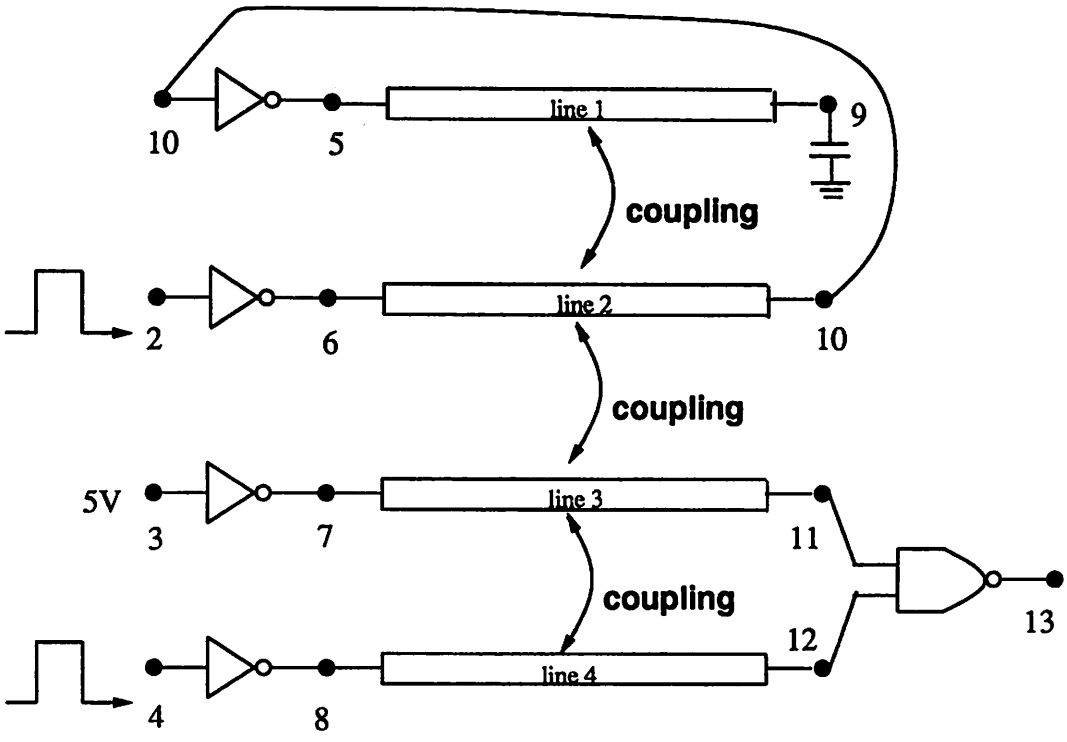


Figure 5.7: A test circuit.

| No. of vec. | SPICE3.e | SWEC |
|---|---|---|
| 3 | 207 s | 4.8 s |
| 24 | 9110 s | 50.4 s |

Table 5.2: CPU times for Coupled Line Simulations.

which has 24 input vectors. The speed gain of SWEC was even more significant, about 180 times, because the convolution integrations of SPICE3.e expanded over all the past history.

The experimental results are very encouraging. We can achieve one to two order-of-magnitude speed gain compared with SPICE3.e.

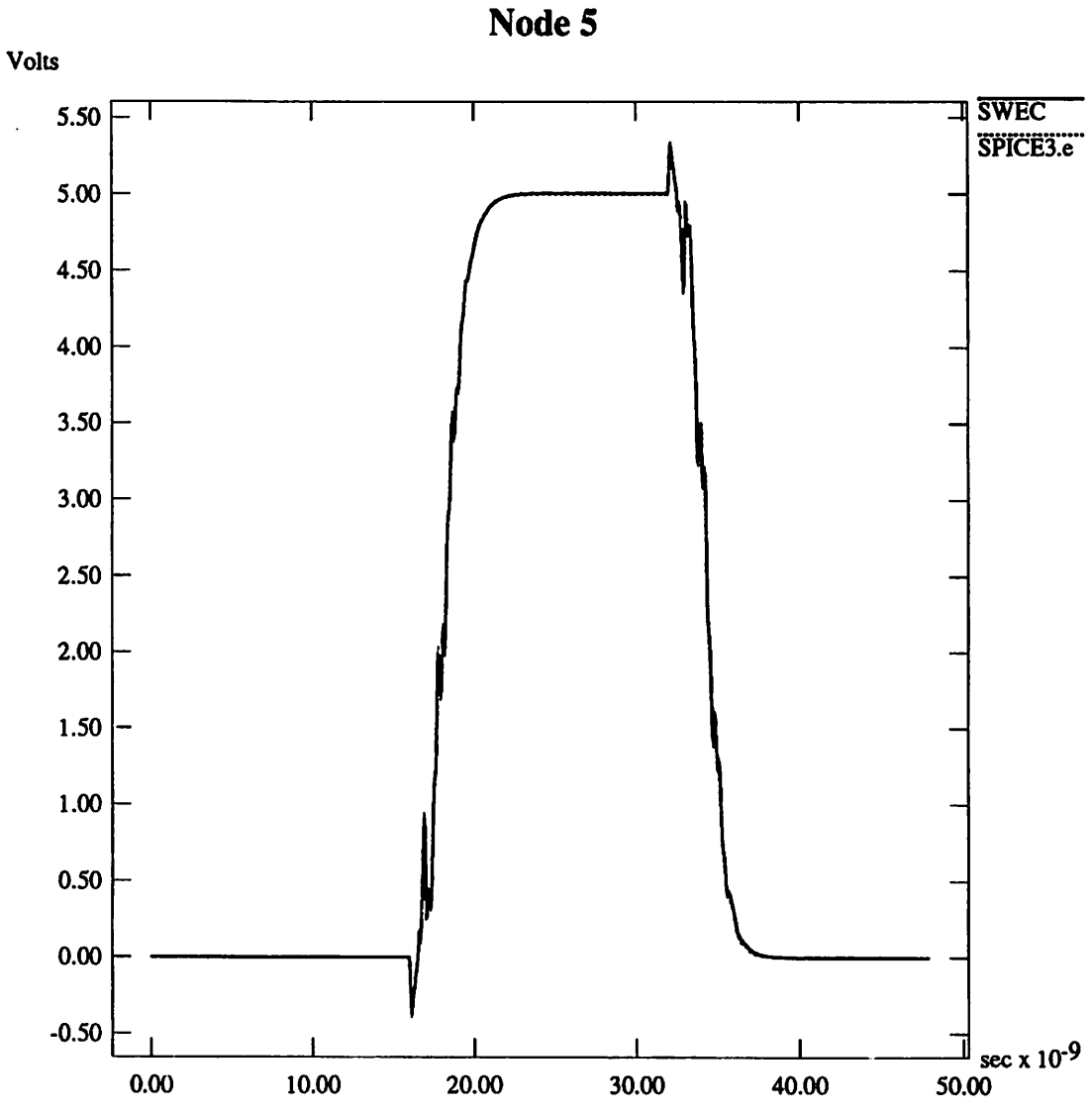


Figure 5.8: The Waveform at Node 5.

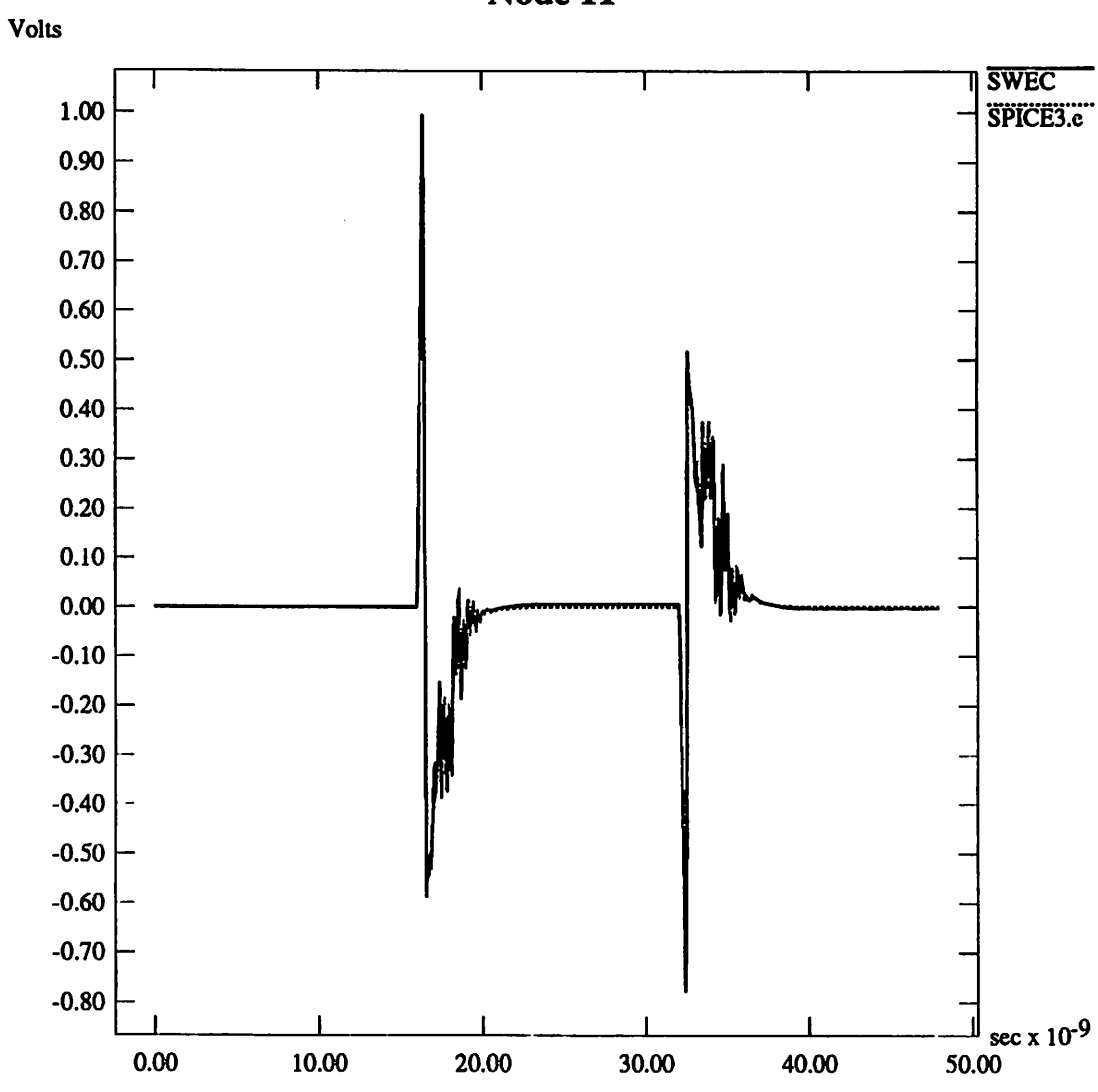


Figure 5.9: The Waveform at Node 11.

# Chapter 6

# Future Work: Recursive Convolution Simulation of Arbitrary Distributed Networks.

The multiconductor Telegrapher equation model introduced in subsection 5.1.2 is not adequate to model general distributed networks, such as nonuniform lines or lines connected in arbitrary three-dimensional geometry. The assumptions introduced by the Telegrapher model are that all the lines of the multiconductor system should be of equal length and the $\boldsymbol{R}$ $\boldsymbol{L}$ $\boldsymbol{G}$ $\boldsymbol{C}$ parameters should be uniform over the lines. These two assumptions restrict the model from handling the general interconnects of today's high speed microelectronic systems. Tape automated bonding (TAB) and tapered etches on chip carriers are example of nonuniform lines. In addition, the implementation of thin-film multilayer interconnects has caused the emergence of irregular geometries such as vias and multilevel

crossing metallic signal strips in orthogonal multilayer configurations. These general configurations and irregularities have created difficulties for the Telegrapher model. We may need to use many Telegrapher multiconductor systems to model the interconnects[1] between a small number of terminals, which makes the simulation impractical.

To handle nonuniform lines, previously [49] has proposed the approach of decomposing the nonuniform multiconductor system into many sections and treating each section as a uniform multiconductor system. Convolution simulation is then applied to each uniform multiconductor section. However, to maintain the desired accuracy a large number of sections may be needed for long lines, which increase the computations. Furthermore, this approach still cannot handle interconnect with general configurations.

In [37], the scattering parameter model was used to model an arbitrary distributed network. The network is treated as a black box because the internal nodes of the network are hidden from the simulation. The terminals of the box are the nodes of the network connected with other nonlinear devices. To characterize the network, the network is assumed to be terminated in its characteristic impedance; therefore, the scattering parameters can fully characterize the electrical behavior of the distributed network. The scattering parameters represent the transfer function of the reflected voltage waves (outputs) versus the incident voltage waves (inputs) of the black box. Convolution can be employed to determine the outputs given the information on the inputs. For the purpose of simulation, to compensate the effects of the added characteristic impedance, in between each terminal and the characteristic impedance, an impedance of the negative impedance value is assumed to be

[1] A section of parallel lines can be treated as a multiconductor system.

connected in series. This approach offers good computational efficiency because the internal electrical information of the network is not evaluated. Usually, the number of internal nodes of a network is much larger than the number of its external nodes.

We propose to apply the recursive convolution to the above scattering parameter approach and employ the Pade approximation to synthesize the scattering parameters. Therefore, (1) the recursive convolution technique can be used to greatly speed up the simulation, (2) there is no need to estimate the scattering parameters and the characteristic impedance over the whole spectrum, and (3) FFT is not needed.

This chapter is organized as follows. In section 6.1, we introduce the concept of scattering parameters. In section 6.2, we introduce the convolution simulation using the scattering parameter models. In section 6.3, we present our recursive convolution simulation with scattering parameters.

## 6.1 Introduction of the Scattering Parameter Model.

The frequency-domain scattering parameters $\boldsymbol{S}(w)$ of a transmission line system describe the relative amplitude and phase of the forward and backward traveling waves, at each port and at each frequency. The transmission line system shown in Fig. 6.1 has the characteristic impedance $Z_m$. The distributed network of the figure represent interconnects of general configurations and with irregularities. Here $V_j^+$ and $V_j^-$ are the forward and backward traveling waves, respectively, at the $j$th port. The scattering parameter is the ratio of the backward traveling wave to the forward traveling wave $S_{ij}(w) = V_i^-(w)/V_j^+(w)$ [50].

Then, each frequency-domain scattering parameter can be determined by $S_{ij}(w) = V_i(w)/V_j(w)$ with all ports except port $j$ terminated in $Z_m$, where $V_i(w)$ is the voltage at port $i$. Let port $j$ be connected to a sinusoidal-wave voltage generator $V_j(w)$.

The scattering parameters $\boldsymbol{S}(w)$ can also be determined by performing the short-circuit admittance measurement. That is

$$\boldsymbol{S}(w) = (\boldsymbol{Y}_m(w) + \boldsymbol{Y}_0(w))^{-1}(\boldsymbol{Y}_m(w) - \boldsymbol{Y}_0(w)), \tag{6.1}$$

where $\boldsymbol{Y}_m(w)$ is the characteristic admittance and $\boldsymbol{Y}_0(w)$ is the short-circuit admittance.

We will use $s_{ij}(t)$ to denote the time-domain scattering parameter, which is the inverse Fourier transform of $S_{ij}(w)$.

## 6.2 Simulation with Scattering Parameters.

Let us consider the $N$-port distributed network of Fig. 6.2 with ports terminated in Thevenin equivalent sources each with characteristic impedance $Z_m$. Thus, the scattering parameter given in the previous section can fully characterize the network in the figure. Therefore,

$$\begin{aligned} v_i(t) &= \sum_{j=1}^{N} \int_0^t g_{ij}(t-\tau) e_j(\tau) d\tau \\ &= \sum_{j=1}^{N} g_{ij}(t) * e_j(t), \end{aligned} \tag{6.2}$$

where $g_{ij}(t)$ is the inverse Fourier transform of the Green's function $G_{ij}(w)$ where

$$\begin{aligned} G_{ij}(w) &= \frac{1 + S_{ij}(w)}{2}, \qquad i = j \\ &= \frac{S_{ij}(w)}{2}, \qquad i \neq j \end{aligned} \tag{6.3}$$

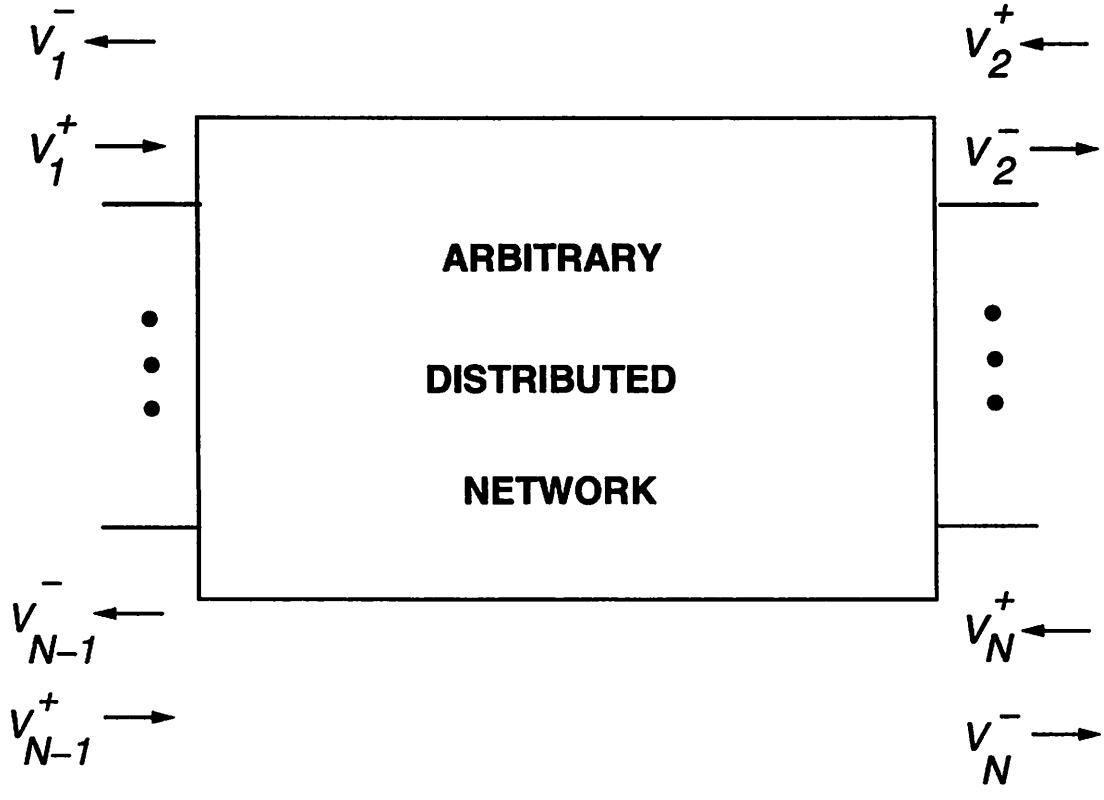


Figure 6.1: $N$-port distributed network.

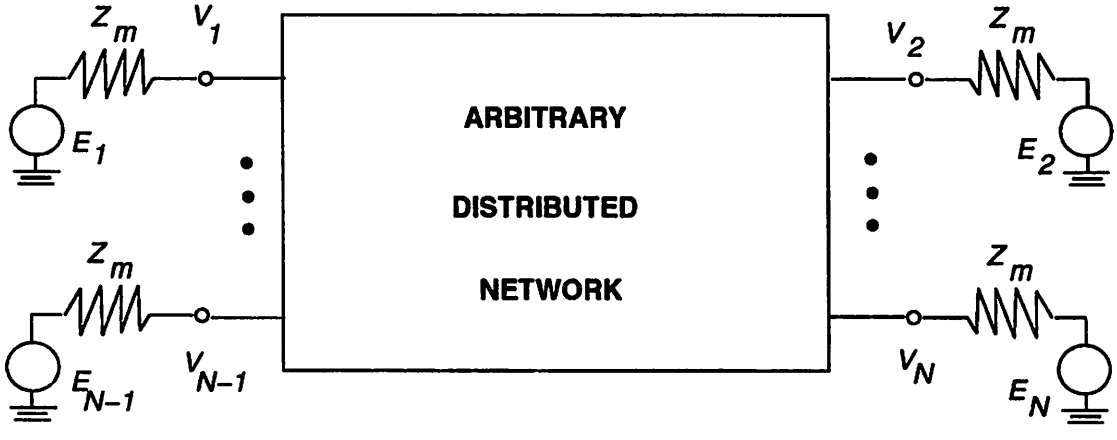


Figure 6.2: $N$-port distributed network with the characteristic impedance termination and arbitrary sources.

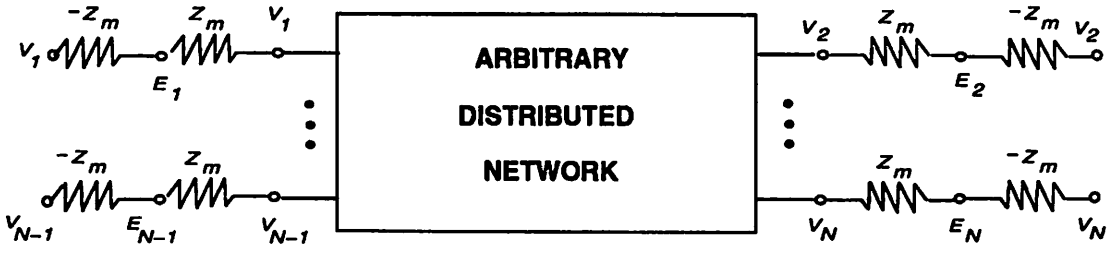


Figure 6.3: $N$-port distributed network with the characteristic impedance removed.

Following the approach in [34] and [37], the effect of the characteristic impedance is removed by inserting a series impedance $-Z_m$ at each termination so that a virtual short circuit is created between the load and the distributed network as shown in Fig. 6.3. Hence, we have one extra branch equation for each port $i$

$$v_i(t) = e_i(t) - i_i(t)Z_m. \tag{6.4}$$

Beside the original nodal equations of the circuit, for each port $i$, we create two extra variables $e_i(t)$ and $i_i(t)$. Since we also have two extra equations Eq.(6.2) and Eq.(6.4) for each port, it is possible to solve all the variables at each integration step.

## 6.3 Recursive Convolution Simulation with Scattering Parameter Models.

In [37], the authors used FFT to derive $g_{ij}(t)$ from $G_{ij}(w)$. Since $G_{ij}(w)$ expands to very high frequency, windowing the spectrum is necessary, which will introduce errors. Furthermore, they need to measure the scattering parameters over the whole spectrum of interest before performing FFT, which is a difficult job. Their approach also suffers from the quadratic complexity of the convolution simulation.

To cope with the first two difficulties of [37], we propose employing the Pade approximation to derive $g_{ij}(t)$, hence avoiding performing FFT and measuring a large number of scattering parameter data. From a few measured data of $G_{ij}(w)$ at high frequency, a fitted complex polynomial can be determined. Then, the Pade approximation can be applied to the polynomial to solve for $g_{ij}(t)$. $g_{ij}(t)$ will assume the particular form, a Dirac function plus a sum of exponential functions. Therefore, recursive convolution can be used to save computations. The method is similar to that described in Chapter 5.

# Chapter 7

# Conclusions.

Electrical transient analysis programs are very important CAD tools for improving the operating speed, reducing the silicon area and power consumption, and detecting the possibility of timing errors in a digital system. The direct approach simulators such as SPICE and ASTAP have been the bread and butter of circuit designers for over two decades. However, these simulators will not be feasible when applied to large submicron systems and are not capable of simulating lossy interconnects in multi-chip modules. Although many modified versions have been introduced to improve those programs' efficiency and usability, there is a sore need for a different approach to handle large and tightly coupled VLSI circuits.

In this dissertation we have accomplished the following: (1) to improve the efficiency for the simulation of tightly coupled circuits, we presented the Stepwise Equivalent Conductance implicit integration; (2) to further speed up the simulation of CMOS digital circuits, we presented the Piecewise-Linear Waveform Event-Driven simulation; and (3) to

solve the simulation of lossy interconnects, we presented the Recursive Convolution simulation based on the Pade approximation. The Stepwise Equivalent Conductance implicit integration avoids solving nonlinear equations in the implicit integration of nonlinear circuits. Computationally expensive Newton Raphson iterations are not needed. Therefore, we can achieve efficiency and stability at the same time. The piecewise linearity on voltage waveforms are exploited by our event-driven approach to decrease the number of event rescheduling, to maximize the integration time steps, and to handle the feedbacks in the circuits. Our recursive convolution approach avoids the difficulty of expanding the convolution integration over the whole past history and speeds up the simulation. The approach is by means of the Pade approximation of transcendental functions in the frequency domain; therefore, it can handle very general coupling situations and frequency varying lines. We also proposed the approach of applying the recursive convolution to arbitrary distributed networks modeled by the scattering parameters.

The experimental results are very encouraging. Our digital CMOS timing simulator, SWEC, based on these techniques outperforms Relax2.3, SPLICE3.0, XPsim, and SPECS2 in both efficiency and accuracy. For the simulation of the circuit with 31 thousand transistors, we could achieve one to two orders-of-magnitude speedups than SPLICE3.0 and Relax2.3. SWEC, while giving accurate results, can be one to two orders-of-magnitude faster than SPICE3.e on the simulation of lossy interconnects. We believe that these techniques can make the multi-chip module simulation possible.

# Appendix A

# Circuits with Inductors, Nonlinear Capacitors and Nonlinear Inductors.

If there are linear inductors in the simulated circuit, we can use either the state equations or the modified nodal equations. For the later the circuit equation will be of the form below

$$\tilde{\mathcal{F}}(\mathbf{X}(t)) + \boldsymbol{H}\dot{\mathbf{X}}(t) = \tilde{\mathbf{I}}_{\mathbf{s}}(t). \tag{A.1}$$

Eq.(A.1) is composed of the nodal equation for each node, which is the Eq.(3.1) mentioned earlier, and the inductor equation for each inductor, which specifies that the voltage drop across the inductor is equal to the inductance of the inductor times the time derivative of the current through the inductor. The variable $\mathbf{X}$ is composed of the node voltages and the currents through the inductors.

Eq.(A.1) can also be transformed into a linear time-varying circuit equation

$$\boldsymbol{A}(t)\mathbf{X}(t) + \boldsymbol{H}\dot{\mathbf{X}}(t) = \tilde{\mathbf{I}}_s(t). \tag{A.2}$$

$\boldsymbol{A}(t)$ is composed of the instantaneous equivalent conductance matrix at time $t$ and submatrices with their entries being equal to 1, 0, or -1 to specify the inductor equations. $\boldsymbol{A}(t)$ satisfies the relation below

$$\boldsymbol{A}(t)\mathbf{X}(t) = \tilde{\mathcal{F}}(\mathbf{X}(t)). \tag{A.3}$$

Eq.(A.2) is of the same form as Eq.(3.2). Therefore, all the discussions regarding Eq.(3.2) in this paper can directly be extended to Eq.(A.2). The sufficient condition of the exactness of the transformation from Eq.(A.1) to Eq.(A.2) is that $\tilde{\mathcal{F}}$ is continuously differentiable.

If there are nonlinear capacitors or nonlinear inductors in the simulated circuit, then the charge of each nonlinear capacitor or the flux of each nonlinear inductor can be thought of as a time-varying function. The charges or the fluxes will also be considered as variables. Their time derivatives will be either the branch currents through the nonlinear capacitors or the voltage differences across the nonlinear inductors. The circuit equations will still be of the same form as Eq.(3.2). Therefore, the discussions in this paper can also be extended to cover these kinds of circuits.

# Appendix B

# Approximation Errors.

In this Appendix, the numerical error introduced from $t_n$ to $t_{n+1}$ by using the solution of Eq.(3.8) as the approximated solution of Eq.(3.2) will be derived. Let us denote this approximation error $\tau_n$. Then, the local truncation error for the integration of Eq.(3.2) from $t_n$ to $t_{n+1}$ will be the sum of $\tau_n$ and the error introduced by the integration scheme that we use to integrate Eq.(3.8) for the time step.

Let us rewrite Eq.(3.2) below:

$$\boldsymbol{G}(t)\mathbf{V}(t) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I}_s(t). \tag{B.1}$$

Given $\mathbf{V}(t_n)$, we are trying to solve for $\mathbf{V}(t_{n+1})$ of Eq.(B.1).

The $\boldsymbol{G}(t)$ can be expanded into Taylor series around the neighborhood of $t_n$:

$$(\boldsymbol{G}(t_n) + \dot{\boldsymbol{G}}(t_n)(t-t_n) + \frac{1}{2}\ddot{\boldsymbol{G}}(t_n)(t-t_n)^2 + \cdots)\mathbf{V}(t) + \boldsymbol{C}\dot{\mathbf{V}}(t) = \mathbf{I}_s(t). \tag{B.2}$$

Let us consider the first two terms of the Taylor series,

$$(\boldsymbol{G}(t_n) + \dot{\boldsymbol{G}}(t_n)(t-t_n))\mathbf{V}'(t) + \boldsymbol{C}\dot{\mathbf{V}}'(t) = \mathbf{I}_s(t), \tag{B.3}$$

with $\mathbf{V}'(t_n) = \mathbf{V}(t_n)$. Then, $\mathbf{V}'(t_{n+1})$ will be the approximated solution of $\mathbf{V}(t_{n+1})$. We use a column vector $\tau_n^1$ to denote the errors, i.e. $\tau_n^1 = \mathbf{V}(t_{n+1}) - \mathbf{V}'(t_{n+1})$.

To solve $\mathbf{V}'(t_{n+1})$ of Eq.(B.3), we further approximate Eq.(B.3) by the following equation:

$$\mathcal{G}\mathbf{V}"(t) + C\dot{\mathbf{V}}"(t) = \mathbf{I}_s(t), \tag{B.4}$$

with $\mathcal{G} = G(t_n) + \frac{h_n}{2}\dot{G}(t_n)$, where $h_n = t_{n+1} - t_n$, and $\mathbf{V}"(t_n) = \mathbf{V}'(t_n) = \mathbf{V}(t_n)$. $\mathbf{V}"(t_{n+1})$ is then the approximated solution of $\mathbf{V}'(t_{n+1})$, and let us denote the errors due to this by $\tau_n^2 = \mathbf{V}'(t_{n+1}) - \mathbf{V}"(t_{n+1})$.

Then,

$$\tau_n = \tau_n^1 + \tau_n^2. \tag{B.5}$$

## B.1 The analysis of $\tau_n^2$.

The solutions for $\mathbf{V}'(t)$ for Eq.(B.3) are composed of two parts: one is the homogeneous solution with $\mathbf{I}_s(t) = \mathbf{0}$, denoted by $\mathbf{V}'_c(t)$, and the other is the zero initial state solution, denoted by $\mathbf{V}'_*(t)$, where $\mathbf{V}'_*(t_n) = \mathbf{0}$. Therefore, $\mathbf{V}'(t) = \mathbf{V}'_c(t) + \mathbf{V}'_*(t)$. Similarly, $\mathbf{V}"(t)$, $\mathbf{V}"(t) = \mathbf{V}"_c(t) + \mathbf{V}"_*(t)$.

By setting $\mathbf{I}_s(t) = \mathbf{0}$ in Eq.(B.3), we get

$$(G(t_n) + \dot{G}(t_n)(t - t_n))\mathbf{V}'_c(t) + C\dot{\mathbf{V}}'_c(t) = \mathbf{0} \tag{B.6}$$

Eq.(B.6) can be solved exactly with the solution

$$\mathbf{V}'_c(t) = e^{-C^{-1}(G(t_n)(t-t_n)+\frac{1}{2}\dot{G}(t_n)(t-t_n)^2)}\mathbf{V}^0, \tag{B.7}$$

where $\mathbf{V^0} = \mathbf{V}'_{\mathbf{c}}(t_n) = \mathbf{V}"_{\mathbf{c}}(t_n)$. Hence,

$$\mathbf{V}'_{\mathbf{c}}(t_{n+1}) = e^{-\boldsymbol{C}^{-1}(\boldsymbol{G}(t_n)h_n + \frac{1}{2}\dot{\boldsymbol{G}}(t_n)h_n^2)}\mathbf{V^0}. \tag{B.8}$$

On the other hand, by setting $\mathbf{I}_{\mathbf{s}}(t) = \mathbf{0}$ in Eq.(B.4), we get

$$\mathbf{V}"_{\mathbf{c}}(t) = e^{-\boldsymbol{C}^{-1}\boldsymbol{\mathcal{G}}(t-t_n)}\mathbf{V^0}. \tag{B.9}$$

Hence,

$$\mathbf{V}"_{\mathbf{c}}(t_{n+1}) = e^{-\boldsymbol{C}^{-1}\boldsymbol{\mathcal{G}}h_n}\mathbf{V^0}. \tag{B.10}$$

By using the fact $\boldsymbol{\mathcal{G}} = \boldsymbol{G}(t_n) + \frac{h_n}{2}\dot{\boldsymbol{G}}(t_n)$, we have $\mathbf{V}'_{\mathbf{c}}(t_{n+1}) = \mathbf{V}"_{\mathbf{c}}(t_{n+1})$. Therefore, $\tau_n^2$ is not caused by the zero input solution part.

Let us next look at the situation with $\mathbf{V^0} = \mathbf{0}$. By subtracting Eq.(B.4) from Eq.(B.3), we get

$$\boldsymbol{G}(t_n)(\mathbf{V}'_*(t) - \mathbf{V}"_*(t)) + \boldsymbol{C}(\dot{\mathbf{V}}'_*(t) - \dot{\mathbf{V}}"_*(t)) = \dot{\boldsymbol{G}}(t_n)(\frac{h_n}{2}\mathbf{V}"_*(t) - (t-t_n)\mathbf{V}'_*(t)),$$

or

$$\boldsymbol{G}(t_n)\mathbf{e}(t) + \boldsymbol{C}\dot{\mathbf{e}}(t) = \dot{\boldsymbol{G}}(t_n)(\frac{h_n}{2}\mathbf{V}"_*(t) - (t-t_n)\mathbf{V}'_*(t)), \tag{B.11}$$

where $\mathbf{e}(t) = \mathbf{V}'_*(t) - \mathbf{V}"_*(t)$.

$\mathbf{V}'_*(t)$ and $\mathbf{V}"_*(t)$ on the right hand side of Eq.(B.11) can be expanded into Taylor series around $t_n$. By neglecting the higher order terms and by using the fact $\mathbf{V}'_*(t_n) = \mathbf{V}"_*(t_n) = \mathbf{0}$, we have

$$\boldsymbol{G}(t_n)\mathbf{e}(t) + \boldsymbol{C}\dot{\mathbf{e}}(t) = \dot{\boldsymbol{G}}(t_n)(\frac{h_n}{2}(t-t_n)\dot{\mathbf{V}}"_*(t_n) - (t-t_n)^2\dot{\mathbf{V}}'_*(t_n)). \tag{B.12}$$

By replacing $t$ of Eq.(B.3) with $t_n$, we get $\dot{\mathbf{V}}'_*(t_n) = \boldsymbol{C}^{-1}\mathbf{I}_s(t_n)$ because $\mathbf{V}'_*(t)$ is the solution of Eq.(21) with $\mathbf{V}'_*(t_n) = \mathbf{0}$. Similarly, by replacing the $t$ of Eq.(B.4) with $t_n$, we get $\dot{\mathbf{V}}''_*(t_n) = \boldsymbol{C}^{-1}\mathbf{I}_s(t_n)$. Hence, $\dot{\mathbf{V}}'_*(t_n) = \dot{\mathbf{V}}''_*(t_n)$. Note, $\boldsymbol{C}^{-1}\mathbf{I}_s(t_n) = \boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + \dot{\mathbf{V}}(t_n)$. We can use either expression for $\dot{\mathbf{V}}'_*(t_n)$ depending on which information is available.

Therefore,

$$\boldsymbol{G}(t_n)\mathbf{e}(t) + \boldsymbol{C}\dot{\mathbf{e}}(t) = \dot{\boldsymbol{G}}(t_n)\boldsymbol{C}^{-1}\mathbf{I}_s(t_n)(\frac{h_n}{2}(t-t_n) - (t-t_n)^2). \tag{B.13}$$

We can use the Laplace transform method to solve for $\mathbf{e}(t_{n+1})$, which is given by the convolution integral,

$$\mathbf{e}(t_{n+1}) = \int_{t_n}^{t_n+h_n} e^{-\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)(t_n+h_n-s)}\boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)\boldsymbol{C}^{-1}\mathbf{I}_s(t_n)(\frac{h_n}{2}(s-t_n) - (s-t_n)^2)ds. \tag{B.14}$$

Since $h$ is small, $\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)(t_n + h - s)$ is approximately a zero matrix, during the integral interval, $e^{-\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)(t_n+h-s)} \approx \mathbf{1}$, the identity matrix. Therefore,

$$\begin{aligned}\mathbf{e}(t_n + h_n) &= \int_{t_n}^{t_n+h_n} \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)\boldsymbol{C}^{-1}\mathbf{I}_s(t_n)(\frac{h_n}{2}(s-t_n) - (s-t_n)^2)ds \\ &= \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)\boldsymbol{C}^{-1}\mathbf{I}_s(t_n)(\frac{-h_n^3}{12}).\end{aligned} \tag{B.15}$$

Because $\tau_n^2 = \mathbf{e}(t_n + h_n)$,

$$\tau_n^2 = \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)\boldsymbol{C}^{-1}\mathbf{I}_s(t_n)(\frac{-h_n^3}{12}),$$

or

$$= \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)(\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + \dot{\mathbf{V}}(t_n))(\frac{-h_n^3}{12}). \tag{B.16}$$

## B.2 The analysis of $\tau_n^1$.

Subtracting Eq.(B.3) from Eq.(B.2), we get:

$$(\boldsymbol{G}(t_n) + \dot{\boldsymbol{G}}(t_n)(t-t_n))\mathbf{e}'(t) + \boldsymbol{C}\dot{\mathbf{e}}'(t) = -(\frac{1}{2}\ddot{\boldsymbol{G}}(t_n)(t-t_n)^2 + \cdots)\mathbf{V}(t), \qquad \text{(B.17)}$$

where $\mathbf{e}'(t) = \mathbf{V}(t) - \mathbf{V}'(t)$, and $\mathbf{e}'(t_n) = \mathbf{0}$. $\mathbf{e}'(t)$ represents the error introduced by using $\mathbf{V}'(t)$ as the solution of $\mathbf{V}(t)$. We next solve for $\mathbf{e}'(t_n + h_n)$.

Eq.(B.17) is of the form of Eq.(B.3) with $\mathbf{I_s}(t)$ equal to

$$-(\frac{1}{2}\ddot{\boldsymbol{G}}(t_n)(t-t_n)^2 + \cdots)\mathbf{V}(t). \qquad \text{(B.18)}$$

As mentioned in the previous subappendix, we can solve the equations below

$$\boldsymbol{\mathcal{G}}\mathbf{e}"(t) + \boldsymbol{C}\dot{\mathbf{e}}"(t) = -(\frac{1}{2}\ddot{\boldsymbol{G}}(t_n)(t-t_n)^2 + \cdots)\mathbf{V}(t), \qquad \text{(B.19)}$$

with $\boldsymbol{\mathcal{G}} = \boldsymbol{G}(t_n) + \frac{h_n}{2}\dot{\boldsymbol{G}}(t_n)$ and $\mathbf{e}"(t_n) = \mathbf{0}$, and use $\mathbf{e}"(t_n + h_n)$ as the approximated solution for $\mathbf{e}'(t_n + h_n)$. This approximation will introduce a $\tau^2$ type of error as explained in the previous subappendix. However, when we calculate Eq.(B.18) with $t = t_n$, we have zero currents. This means that $\dot{\mathbf{e}}'_*(t_n) = \dot{\mathbf{e}}"_*(t_n) = \mathbf{0}$ because both are equal to $\boldsymbol{C}^{-1}\mathbf{I_s}(t_n)$. Therefore, we need to consider the second order terms for the Taylor expansions of $\mathbf{e}'(t)$ and $\mathbf{e}"(t)$, respectively. In this way, we will have $\dot{\boldsymbol{G}}(t_n)(\frac{h_n}{2}(t-t_n)^2\ddot{\mathbf{e}}"_*(t_n) - (t_n)^3\ddot{\mathbf{e}}'_*(t_n))$ on the right hand side of Eq.(B.12). Since the power of $(t-t_n)$ is one more than that of Eq.(B.12), we get the $\tau^2$ type of error by using $\mathbf{e}"(t_{n+1})$ as $\mathbf{e}'(t_{n+1})$ is of the order of $o(h^4)$, which is much smaller than the $\tau_n^2$ derived previously. In fact, later in this subappendix, we will show $\tau_n^1$ is also of $o(h^3)$, so we can neglect the $o(h^4)$ errors. Therefore, we will assume that no error is introduced by using $\mathbf{e}"(t_{n+1})$ as $\mathbf{e}'(t_{n+1})$.

From Taylor's expansion of $\mathbf{V}(t)$ in Eq.(B.19) around $t_n$ and neglecting the higher order terms, we have

$$\mathcal{G}\mathbf{e}''(t) + C\dot{\mathbf{e}}''(t) = -\frac{(t-t_n)^2}{2}\ddot{G}(t_n)\mathbf{V}(t_n). \tag{B.20}$$

Again, we can use the Laplace transform method to solve for $\mathbf{e}''(t_{n+1})$, which is

$$\mathbf{e}''(t_n+h) = \int_{t_n}^{t_n+h} e^{-C^{-1}\mathcal{G}(t_n+h-s)}C^{-1}\ddot{G}(t_n)\mathbf{V}(t_n)(-\frac{(s-t_n)^2}{2})ds. \tag{B.21}$$

Since $h_n$ is a very small number, during the integral interval, $e^{-C^{-1}\mathcal{G}(t_n+h-s)} \approx \mathbf{1}$. Therefore,

$$\begin{aligned}\mathbf{e}(t_n+h) &= \int_{t_n}^{t_n+h} C^{-1}\ddot{G}(t_n)\mathbf{V}(t_n)(-\frac{(s-t_n)^2}{2})ds \\ &= C^{-1}\ddot{G}(t_n)\mathbf{V}(t_n)(\frac{-h^3}{6}).\end{aligned} \tag{B.22}$$

Hence,

$$\tau_n^1 = C^{-1}\ddot{G}(t_n)\mathbf{V}(t_n)(\frac{-h^3}{6}). \tag{B.23}$$

Combining the $\tau_n^2$ derived previously, we have

$$\begin{aligned}\tau_n &= \tau_n^1 + \tau_n^2 \\ &= -C^{-1}\ddot{G}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) - C^{-1}\dot{G}(t_n)(C^{-1}G(t_n)\mathbf{V}(t_n) + \dot{\mathbf{V}}(t_n))(\frac{h_n^3}{12}).\end{aligned} \tag{B.24}$$

# Appendix C

# Time Step Constraints.

In this Appendix, the upper bound on the infinity norm of Eq.(3.9) under the constraints of $h_n$ on Eq.(3.11) is derived. The infinity norm of a column vector is the largest absolute value of the entries of the vector. Since the column vector of Eq.(3.9) represents the approximation errors on the node voltages, the infinity norm of Eq.(3.9) can give the measure of the the error introduced.

To make the notations simple, we follow the notations used in the previous Appendix. The column vector of Eq.(3.9) is denoted by $\tau_n$, which is equal to $\tau_n^1 + \tau_n^2$. We will first derive the upper bound on $\| \tau_n^1 \|$ under Eq.(3.11), and then the upper bound on $\| \tau_n^2 \|$ under Eq.(3.11). Then, $\| \tau_n \| \leq \| \tau_n^1 \| + \| \tau_n^2 \|$. We will show that the upper bound on $\| \tau_n \|$ is approximately $\frac{\epsilon}{3}\Delta V$.

For the purpose of the derivation, the following two theorems have been developed.

**Theorem 2** *If*

$$(|\frac{\ddot{G}_i(t_n)h_n^2}{G_i(t_n)}|) \leq \epsilon \quad \forall \text{ device } i, \tag{C.1}$$

*then*

$$\| (h_n^2 \ddot{\boldsymbol{G}}(t_n)) \boldsymbol{G}(t_n)^{-1} \| \leq \epsilon \tag{C.2}$$

**Proof:** Since

$$1 = \| \boldsymbol{G}(t_n) \boldsymbol{G}(t_n)^{-1} \| \leq \| \boldsymbol{G}(t_n) \| \| \boldsymbol{G}(t_n)^{-1} \|, \tag{C.3}$$

and $\boldsymbol{G}(t_n)$ is diagonally dominant,

$$\frac{1}{2d_{max}} \leq \| \boldsymbol{G}(t_n)^{-1} \|, \tag{C.4}$$

where $d_{max}$ represents the largest diagonal element of $\boldsymbol{G}(t_n)$. Note that

$$\| \boldsymbol{G}(t_n) \|_\infty = MAX_i(\sum_{j=1} |(\boldsymbol{G}(t_n))_{ij}|) \leq 2d_{max}. \tag{C.5}$$

Therefore,

$$\| (h_n^2 \ddot{\boldsymbol{G}}(t_n)) \boldsymbol{G}(t_n)^{-1} \| \leq \| h_n^2 \ddot{\boldsymbol{G}}(t_n) \| \| \boldsymbol{G}(t_n)^{-1} \| \leq \epsilon \, 2d_{max} \frac{1}{2d_{max}} \leq \epsilon. \tag{C.6}$$

■

**Theorem 3** *If*

$$(|\frac{\dot{G}_i(t_n) h_n}{G_i(t_n)}|) \leq \epsilon \quad \forall \text{ device } i, \tag{C.7}$$

*then*

$$\| (h_n \dot{\boldsymbol{G}}(t_n)) \boldsymbol{G}(t_n)^{-1} \| \leq \epsilon \tag{C.8}$$

The proof for Theorem 3 is similar to that of Theorem 2, so the details are omitted.

$$\begin{aligned} \| \tau_n^1 \| &= \| \boldsymbol{C}^{-1}\ddot{\boldsymbol{G}}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) \| \\ &= \| \boldsymbol{C}^{-1}\ddot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) \| \\ &\approx \| \ddot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) \| \end{aligned} \tag{C.9}$$

Since the matrix $\boldsymbol{C}$ is diagonally dominant and so is $\boldsymbol{C}^{-1}$, the error introduced on the norm by commuting $\boldsymbol{C}^{-1}$ is small. Note that if $\boldsymbol{C}$ is a diagonal matrix, then the commutation of $\boldsymbol{C}^{-1}$ in Eq.(C.9) will not introduce any approximation.

Under the constraints of Eq.(3.11), we have

$$\begin{aligned} \| \tau_n^1 \| &\approx \| \ddot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n)(\frac{h_n^3}{6}) \| \\ &\leq \| \frac{h_n^2}{6}\ddot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1} \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) \| \\ &\leq \frac{\epsilon}{6} \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) \| \\ &\approx \frac{\epsilon}{6} \| h_n\dot{\mathbf{V}}(t_n) \| \\ &\leq \frac{\epsilon}{6}\Delta V \end{aligned} \tag{C.10}$$

In Eq.(C.10), we use the approximation $\| \dot{\mathbf{V}}(t_n) \| \approx \| \boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) \|$. If in Eq.(3.2) $\mathbf{I_s}(t_n) = \mathbf{0}$, then there will be no approximation introduced. For the case where $\mathbf{I_s}(t_n) \neq \mathbf{0}$, if $\Delta V$ is very small, the approximation incurred will be very small.

On the other hand,

$$\begin{aligned} \| \tau_n^2 \| &= \| \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)(\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + \dot{\mathbf{V}}(t_n))(\frac{h^3}{12}) \| \\ &= \| \boldsymbol{C}^{-1}\dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{G}(t_n)(h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + h_n\dot{\mathbf{V}}(t_n))(\frac{h_n^2}{12}) \| \end{aligned}$$

$$\approx \| \dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)(h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + h_n\dot{\mathbf{V}}(t_n))(\frac{h_n^2}{12}) \| \quad \text{(C.11)}$$

Again, the approximate commutation of the matrix $\boldsymbol{C}^{-1}$ is used in Eq.(C.11).

Under the constraints of Eq.(11), we have

$$\begin{aligned}
\| \tau_n^2 \| &\approx \| \dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1}\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)(h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + h_n\dot{\mathbf{V}}(t_n))(\frac{h_n^2}{12}) \| \\
&\leq \| \frac{h_n}{12}\dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1} \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) + h_n\dot{\mathbf{V}}(t_n) \| \\
&\leq \| \frac{h_n}{12}\dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1} \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| (\| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) \| + \| h_n\dot{\mathbf{V}}(t_n) \|) \\
&\approx \| \frac{h_n}{12}\dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1} \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| (\| h_n\dot{\mathbf{V}}(t_n) \| + \| h_n\dot{\mathbf{V}}(t_n) \|) \\
&\leq \| \frac{h_n}{12}\dot{\boldsymbol{G}}(t_n)\boldsymbol{G}(t_n)^{-1} \| \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| (\Delta V + \Delta V) \\
&\leq \| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| \epsilon\frac{\Delta V}{6} \\
&\leq \epsilon\frac{\Delta V}{6} \qquad \text{(C.12)}
\end{aligned}$$

In Eq.(C.12), the approximation $\| \dot{\mathbf{V}}(t_n) \| \approx \| \boldsymbol{C}^{-1}\boldsymbol{G}(t_n)\mathbf{V}(t_n) \|$ is used again. For the last two lines of Eq.(C.12), the reason why $\| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| \epsilon\frac{\Delta V}{6} \leq \epsilon\frac{\Delta V}{6}$ is because $\| h_n\boldsymbol{C}^{-1}\boldsymbol{G}(t_n) \| \leq 1$, which is the basic requirement for the integration time step $h_n$, otherwise $h_n$ can not be kept to the first order accuracy.

Combining Eq.(C.12) and Eq.(C.10),

$$\begin{aligned}
\| \tau_n \| &\leq \| \tau_n^1 \| + \| \tau_n^2 \| \\
&\leq \epsilon\frac{\Delta V}{6} + \epsilon\frac{\Delta V}{6} \\
&= \epsilon\frac{\Delta V}{3} \qquad \text{(C.13)}
\end{aligned}$$

# Appendix D

# Time Steps Selection for an Inverter.

Our event driven approach works for circuits with feedback loops because of the assumption that the integration time step selected according to Eq.(4.3) for each subcircuit will guarantee that the subcircuit's fanouts will not change slopes within the time step (*time step criterion*). Therefore, a loop of event rescheduling can be avoided, which is explained in section 4.4.

In this Appendix, we show that the time step criterion will hold for a CMOS inverter if its time steps are selected according to Eq.(4.3). Since inverters are the most basic CMOS gates, we assume the validity of Eq.(4.3) applies to all CMOS gates.

First assume that the inverter is in a pull-down situation and its pMOS is almost OFF. Therefore, the pMOS part can be neglected. (We will remove this restriction later in this Appendix.) We have an RC equivalent circuit as in Fig. D.1. $C_L$ is the capacitive load

at the output node of the inverter, and $R$ is the reciprocal of the equivalent conductance of the nMOS.

The input voltage to the inverter, denoted by $V_i$, is equal to $V_{GS}$ of the nMOS, and the output voltage, denoted by $V_o$ will be equal to $V_{DS}$ of the nMOS. Since the input waveform is piecewise linear and given, we know the breakpoints beforehand. Therefore, at every time point, the input can be looked at as a straight line with a slope $\alpha$.

There will be no constraint on the time step if the nMOS stays OFF. The next time point will be the earlier one between the following breakpoint and the time when the nMOS is turned ON. Hence, for the constraints on time steps, we only analyze the following two cases:

**Case (i):** $V_{DS} \geq V_{GS} - V_{th} \geq 0$

In this case, $I_{DS} = \beta(V_{GS} - V_{th})^2$. And,

$$\frac{dV_i(t)}{dt} = \frac{dV_{GS}}{dt} = \frac{d}{dt}(V_{GS} - V_{th}) = \alpha. \tag{D.1}$$

Let, at the time $t_n$,

$$V_{GS} - V_{th} = V_i^0, \quad V_o(t_n) = V_o^0, \quad G_n = G^0. \tag{D.2}$$

Then, at the time $t_n + h$, $V_{GS} - V_{th} = V_i^0 + \alpha h$. The true output change during this time step $h$, $\Delta V_o(t_n, t_n + h)$ is

$$\begin{aligned} \Delta V_o(t_n, t_n + h) &= \frac{1}{C_L}\int_0^h \beta(\alpha t + V_i^0)^2 dt \\ &= \frac{\beta}{C_L}(V_i^{0^2} h + \alpha V_i^0 h^2 + \frac{\alpha^2}{3} h^3). \end{aligned} \tag{D.3}$$

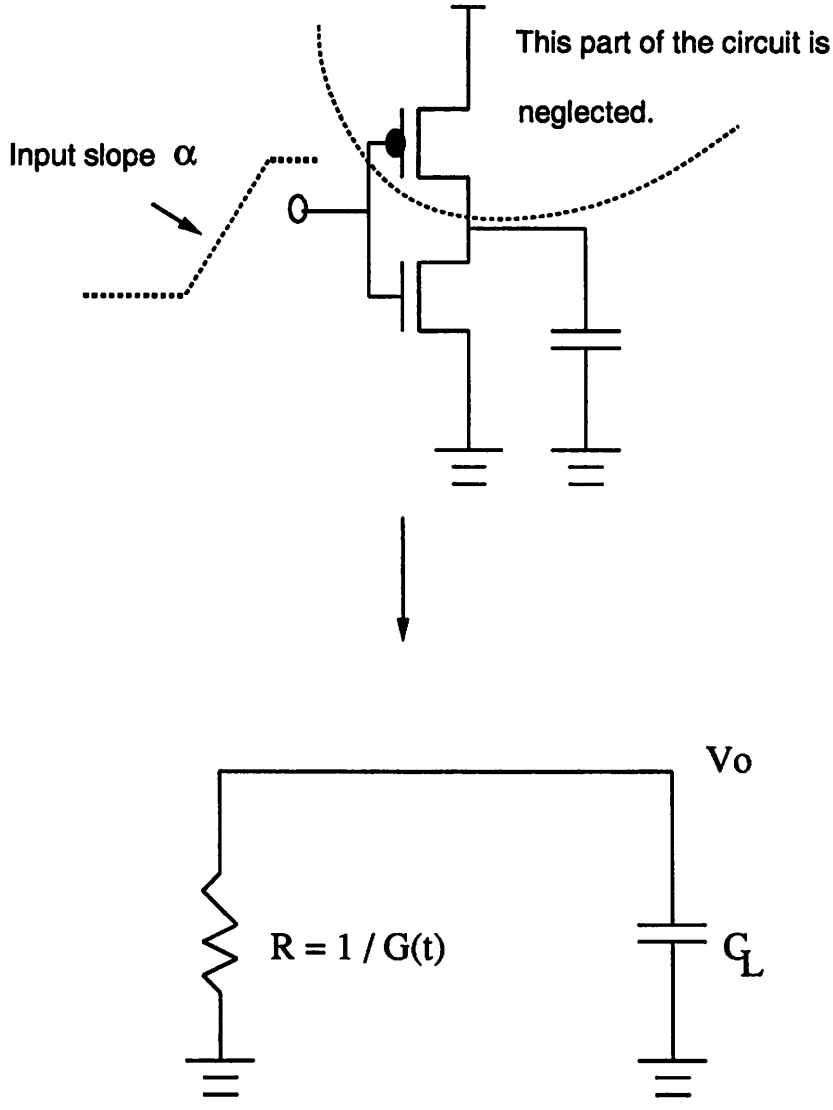


Figure D.1: Equivalent Circuit for the Inverter.

Eq.(D.3) is a polynomial in $\frac{\alpha h}{V_i^0}$. Assume

$$\frac{\alpha h}{3V_i^0} \leq \epsilon, \tag{D.4}$$

where $\epsilon$ is the local error bound. Then we get[1]

$$\Delta V_o(t_n, t_n + h) \approx \frac{\beta}{C_L}(V_i^{0^2} h + \alpha V_i^0 h^2). \tag{D.5}$$

Based on Eq.(4.2),

$$\mathcal{G}_n = G^0 + \frac{h}{2} 2\beta \frac{V_{GS} - V_{th}}{V_{DS}} \dot{V}_{GS}|_{t_n} = \frac{\beta}{V_o^0}(V_i^{0^2} + V_i^0 \alpha h). \tag{D.6}$$

Let $\hat{V}_o$ be the estimated output after time step $h$ calculated using the backward Euler integration, which is

$$\hat{V}_o = \frac{C_L V_o^0}{\mathcal{G}_n h + C_L}. \tag{D.7}$$

Then,

$$\Delta V_{o_n} = \hat{V}_o - V_o^0 = \frac{h\mathcal{G}_n}{h\mathcal{G}_n + C_L} V_o^0 \approx \frac{h\mathcal{G}_n}{C_L} V_o^0, \tag{D.8}$$

where the approximation holds for

$$\frac{\mathcal{G}_n h}{C_L} \leq \epsilon. \tag{D.9}$$

And, under this approximation,

$$\Delta V_{o_n} = \frac{h\mathcal{G}_n}{C_L} V_o^0 = \frac{\beta}{C_L}(h V_i^{0^2} + V_i^0 \alpha h^2). \tag{D.10}$$

[1]Note the neglected third term is associted with the $\ddot{G}(t)$ of Eq.(3.11), which means Eq.(3.11b) implies Eq.(3.11a) for a CMOS inverter.

Eq.(D.9) can be further approximated to be

$$\frac{G^0 h}{C_L} \leq \epsilon, \tag{D.11}$$

because the second term of Eq.(D.6) is the difference between $\mathcal{G}_n$ and $G^0$, and compared with the first term, it is very small when the condition of Eq.(D.4) holds.

From Eq.(D.5) and Eq.(D.10), we see that $\Delta V_o(t_n, t_n+h)$ is equal to $\Delta V_{o_n}$ provided that Eq.(D.4) and Eq.(D.11) hold. Therefore, the % local error will be less than $\epsilon$ if

$$h \leq \frac{3V_i^0}{\alpha}\epsilon, \quad h \leq \frac{C_L}{G^0}\epsilon. \tag{D.12}$$

**Case (ii):** $V_{GS} - V_{th} \geq V_{DS} \geq 0$

In this case, $I_{DS} = \beta(2(V_{GS}-V_{th})V_{DS}-V_{DS}{}^2)$. The equivalent conductance, $G(t)$, of the nMOS at the time $t$ is

$$G(t) = \frac{I_{DS}}{V_{DS}}|_t = \beta(2(V_{GS} - V_{th}) - V_{DS})|_t. \tag{D.13}$$

Assume that $V_{DS}$ is linear with a constant slope during the estimated time step. (The feasibility of this assumption will be checked later.) Then, $G(t)$ will be a linear function of $t$ since both $V_{GS}$ and $V_{DS}$ are linear. Let $G(t) = G^0 + \gamma(t - t_n)$, where $G^0$ is the equivalent conductance at $t_n$ being equal to $\beta(2V_i^0 - V_o^0)$ , and $\gamma = \frac{dG}{dt} = \beta(2\frac{dV_{GS}}{dt} - \frac{dV_{DS}}{dt})$ evaluated at $t_n$. Given the output voltage at $t_n$, say $V_o^0$, we have at $t_n + h$

$$V_o(t_n + h) = V_o^0 e^{-(G^0 h + \frac{\gamma}{2}h^2)/C_L}. \tag{D.14}$$

According to Eq.(4.2) $\mathcal{G}_n$ is equal to $G^0 + \frac{\gamma h}{2}$. Hence, by setting this value to be the conductance of the resistor of Fig. D.1, the output voltage at $t_n + h$ will be

$$\hat{V_o} = V_o^0 e^{-\frac{h}{C_L}(G^0 + \frac{\gamma h}{2})} = V_o^0 e^{-(G^0 h + \frac{\gamma}{2}h^2)/C_L}, \tag{D.15}$$

which is exactly the same as Eq.(D.14).

Now we need to check the validity of the assumption made regarding $V_{DS}$. We assumed that $V_{DS}$ is linear during this time step $h$. The change of the output voltage during the time step $h$ is

$$\Delta V_o = V_o^0(1 - e^{-(G^0 h + \frac{\gamma}{2}h^2)/C_L}) \approx V_o^0(\frac{1}{C_L}(G^0 h + \frac{\gamma}{2}h^2)), \tag{D.16}$$

under the condition

$$\frac{1}{C_L}(G^0 h + \frac{\gamma}{2}h^2) \leq \epsilon. \tag{D.17}$$

We need to have $\Delta V_o$ linear in terms of $h$. From Eq.(D.16), we know this is true only if

$$\frac{\frac{\gamma}{2}h^2}{G^0 h} \leq \epsilon. \tag{D.18}$$

By using the condition of Eq.(D.18), Eq.(D.17) can be simplified to

$$h \leq \epsilon \frac{C_L}{G^0}. \tag{D.19}$$

Eq.(D.18) is therefore

$$h \leq 2\epsilon \frac{G^0}{\gamma} = 2\epsilon \frac{\beta(2V_i^0 - V_o^0)}{\beta(2\alpha - \frac{dV_{DS}}{dt}|_{t_n})} \leq 2\epsilon \frac{V_i^0}{\alpha}. \tag{D.20}$$

The above is true because $V_o^0 \geq 0$, and $\frac{dV_{DS}}{dt} \leq 0$ (opposite sign with $\alpha$).

From equations (D.12), (D.19), and (D.20), we have the following constraints on the time step $h_n$ for an inverter circuit with the pMOS being OFF

$$h_n \leq \epsilon \frac{C_L}{G(t_n)}$$

and

$$h_n \leq 2\epsilon \frac{V_{GS} - V_{th}}{|\alpha|}|_{t_n}, \tag{D.21}$$

which applies to both situations: $V_{DS} \geq V_{GS} - V_{th} \geq 0$ and $V_{GS} - V_{th} \geq V_{DS} \geq 0$. If $h_n$ fulfills the inequalities in Eq.(D.21), then the % local error is less than $\epsilon$.

If the pMOS is not OFF then it will have its own time step constraint of the same form as Eq.(D.21). Hence, it is reasonable to assume the next time step to be such an $h$ which satisfies the Eq.(D.21) for the nMOS and the Eq.(D.21) for the pMOS.

We get the piecewise linearity on the output waveforms and on the MOS I-V characteristic by controlling the sizes of time steps. The only assumption required is **the piecewise linearity on the input waveforms.**

Eq.(D.21) is the time step constraint for one inverter or the constraint associted with the node connected to only one device. For general circuit configurations, Eq.(D.21) will be generalized to be Eq.(4.3).

# Appendix E

# Piecewise-Linear Waveform Approximation Algorithm.

Fig. E.1a shows the output points for a typical simulation which preserves the approximate piecewise linearity. However, in order to exploit this linearity in our simulation approach, the waveform should be transformed to that shown in Fig. E.1b which has its break points specified explicitly and is linear between adjacent break points. In this Appendix, we will elaborate the procedure of determining those break points.

The piecewise linear approximation of a signal at the inputs of a gate causes errors at its output. We first choose what is a tolerable error bound at the output and then, we determine the breakpoints of an input waveform, such that the error at the output does not exceed the chosen bound.

Let $\{x_n\} = \{(t_n, v_n)\}$ be the ordered samples of the original waveforms. Our goal is to find such a subset of $\{x_n\}$ that after the input is replaced with piecewise linear curve

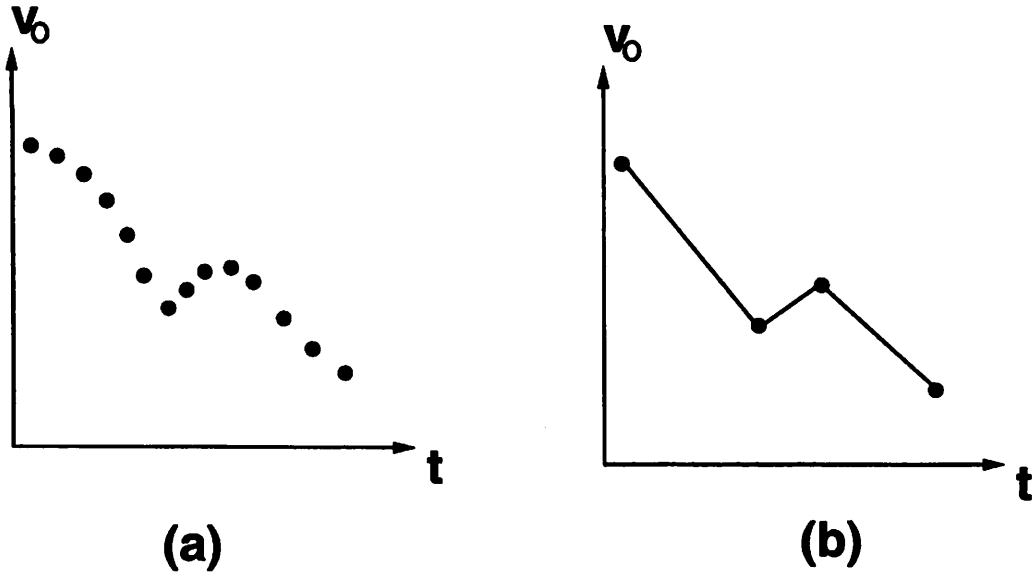


Figure E.1: Simulation Waveforms.

with the chosen subset as breakpoints, the output of the inverter will be accurate with the chosen error bound. For the sake of simplicity, we only consider local errors. In the worst case, the global error will be an accumulated effect of all the local errors. We define the term *% local error* at $t_n$ as $\frac{|\Delta V_o(t_n)-\Delta \hat{V}_o(t_n)|}{\Delta V_o(t_n)}$, where $\Delta V_o(t_n)$ is the true output change from $t_{n-1}$ to $t_n$ if no approximation has been applied. $\Delta \hat{V}_o(t_n)$ is the output change from $t_{n-1}$ to $t_n$ by using $\{x_i\}_{i=1}^{i=n-1}$, and, the extrapolated $\hat{x_n}$ from $x_{n-2}$ and $x_{n-1}$, as inputs. We call the % local error bound $\epsilon$, and assume it is a known value, set up explicitly.

In MOS circuits, it is convenient to define two threshold voltages: the *high threshold voltage*, $V_{th}^H$, being 80% of $Vdd$ and the *low threshold voltage*, $V_{th}^L$, being 20% of $Vdd$. An input signal which is above the $V_{th}^H$ can almost fully turn ON every nMOS transistor, and turn OFF every pMOS transistor. An input signal which is below the $V_{th}^L$ can fully turn ON the pMOS transistor but turn OFF the nMOS transistor. Hence, we only need to represent waveforms very accurately between these two threshold voltages. For a sequence of $\{x_n\}$

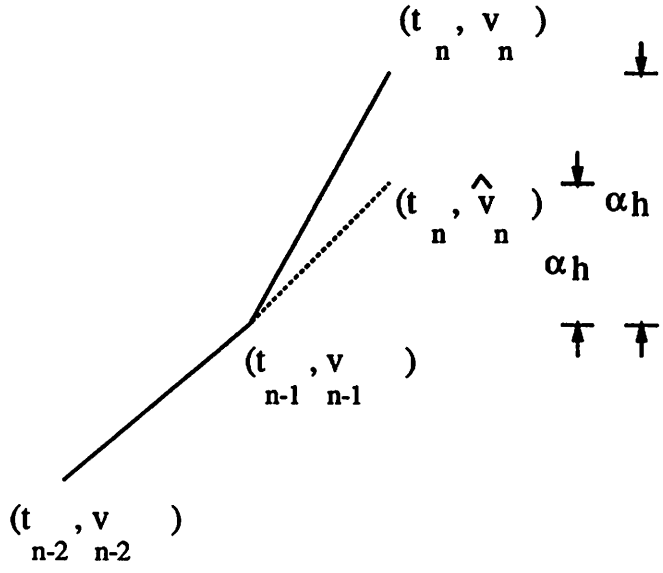


Figure E.2: Piecewise-Linear Approximation.

above $V_{th}^H$ or below $V_{th}^L$, we only pick the starting point, the ending point, and the extreme point, which is the point with the largest voltage if $\{x_n\}$ is above $V_{th}^H$ or the point with the smallest voltage if $\{x_n\}$ is below $V_{th}^L$, as the breakpoints. A sequence of $\{x_n\}$ between those threshold voltages can drive a MOS transistor in the saturated region or in the active region depending on which among $V_{GS} - V_{th}$ and $V_{DS}$ of that transistor is larger.

(i) $V_{DS} \geq V_{GS} - V_{th} \geq 0$ :

Let us consider again Eq.(D.5):

$$\Delta V_o \approx \frac{\beta}{C_L}(V_i^{0^2} h + \alpha V_i^0 h^2). \tag{E.1}$$

Note that $V_i^0$ denotes $V_{GS} - V_{th}$ evaluated at $t_{n-1}$. $\Delta V_o$ is the output change between $t_{n-1}$ and $t_n$, and $\alpha$ is the slope of the input signal from $t_{n-1}$ to $t_n$. Suppose that $\alpha$ is approximated by some value $\hat{\alpha}$. The % local error introduced by this approximation by

using Eq.(E.1) will be

$$\frac{|\Delta V_o(t_n) - \Delta \hat{V}_o(t_n)|}{\Delta V_o(t_n)} = \frac{|\alpha h - \hat{\alpha} h|}{V_i^0}. \tag{E.2}$$

Since $V_i^0$ is at most about 4 Volts and at least above 0 Volt, we use the average, 2 Volts, to represent it.

The difference between $\alpha h$ and $\hat{\alpha} h$ is exactly the same as the difference between $v_n$ and $\hat{v_n}$, which follows from the definition of a slope and is depicted in Fig. E.2. Therefore, whenever $\frac{|v_n - \hat{v_n}|}{2} > \epsilon$, i.e. the $\hat{x_n}$ based on the extrapolation of $x_{n-2}$ and $x_{n-1}$ is no longer accurate enough, we choose a breakpoint $x_{n-1} = (t_{n-1}, v_{n-1})$.

(ii) $V_{GS} - V_{th} \geq V_{DS} \geq 0$ :

We look again at Eq.(D.16):

$$\Delta V_o \approx \frac{V_o^0}{C_L}(G^0 h + \frac{\gamma}{2} h^2). \tag{E.3}$$

$\gamma$ is $\frac{dG}{dt}$ from $t_{n-1}$ to $t_n$, which is equal to $\beta(2\frac{dV_{GS}}{dt} - \dot{V}_{DS})$ from $t_{n-1}$ to $t_n$, and $G^0 = \beta(2V_i^0 - V_o^0)$. The discrepancy in $\alpha$ will introduce errors in $\gamma$. The approximated value of $\gamma$ is denoted by $\hat{\gamma}$. The % local error by using Eq.(E.3) will be

$$\begin{aligned} \frac{|\Delta V_o(t_n) - \Delta \hat{V}_o(t_n)|}{\Delta V_o(t_n)} &= \frac{|\gamma h - \hat{\gamma} h|}{2G^0} \\ &= \frac{|\alpha h - \hat{\alpha} h|}{V_i^0 + (V_i^0 - V_o^0)} \\ &\leq \frac{|\alpha h - \hat{\alpha} h|}{V_i^0}. \end{aligned} \tag{E.4}$$

The % local error is smaller than that of Eq.(E.2), which means the constraint of Eq.(E.2) is tighter. Hence, whenever $\frac{|v_n - \hat{v_n}|}{2} > \epsilon$, we pick a breakpoint $x_{n-1} = (t_{n-1}, v_{n-1})$. This

expression is independent of the type of transistor that $\{x_n\}$ is driving and can be extended to a general CMOS gate.

Let us consider a general CMOS gate, with a given a sequence of samples $\{x_n\}$ having values between $V_{th}^L$ and $V_{th}^H$. Our algorithm starts at the beginning of the sequence and examines consecutive three samples. Let $x_{i-2}$, $x_{i-1}$, and $x_i$ be the three samples we are checking now. If

$$|v_i - v_{i-1} - (t_i - t_{i-1})\frac{v_{i-1} - v_{i-2}}{t_{i-1} - t_{i-2}}| > 2\epsilon, \tag{E.5}$$

then $x_{i-1}$ is kept as a breakpoint, and in the next step we examine the samples: $x_{i-1}$, $x_i$, and $x_{i+1}$. Otherwise, $x_{i-1}$ is not a breakpoint. The next triplet to be examined is: $x_{i-1}$, $x_i$, and $x_{i+1}$.

Therefore, we have a linear complexity and a real time[1] algorithm to determine the break points. For every three points we check the condition specified in Eq.(E.5) (or Eq.(4.4)) to determine whether the middle point is a break point.

[1]This means that we don't need to gather all the data points before we proceed our algorithm.

# Appendix F

# Remarks on Recursive Convolution Formulation.

Professor Omar Wing of Columbia University recently pointed out that the recursive convolution formulation was proposed in [51] in 1975. The authors of [51] observed that if a system's time domain unit step response behaves like an exponential function, with the asymptotic waveform, then it is possible to approximate that unit step response by one or two exponential terms, which will be its primary and second primary components; then, later the convolution can be performed recursively.

Their approach would fail if the exponents are complex (the unit step response is no longer asymptotic), and have no control of the errors introduced by the exponential fitting. It may have been the reason why after 17 years their work was still not applied to solve any practical problem.

The contributions of our work are as follows:

1. We combine the recursive convolution formulation (independent of their work) and the Pade approximation to solve the lossy transmission line simulation. The approach does not matter if the poles of the Pade approximation are complex or not and no approximation is introduced from the frequency domain Pade rational polynomial to its time domain exponential impulse response.
2. We develop the error analysis for the Pade approximation, which is very important.

# Appendix G

# Remarks on Computing Time-Domain Response of Lossy Multiconductor Line System.

## G.1 The eigenvector matrix $S_v(s)$ is continuous.

It has been proved that if the eigenvalues are distinct then the eigenvectors associated to them will be continuous [52]. We assume that this distinct-eigenvalue property holds for the interval of $s$ that we are evaluating. This assumption is valid for most coupled line systems except for the rare situation where the coupling across $k$ lines is equal to the coupling across $j$ lines, for some $k \neq j$. This exceptional case happens only when there exists hardly any coupling. In that case, the simple-line approach will be appropriate.

## G.2 The $K(y)$ of Eq.(5.63) is diagonalizable.

$$\boldsymbol{K}(y) = (\boldsymbol{L} + y\boldsymbol{R})(\boldsymbol{C} + y\boldsymbol{G}). \tag{G.1}$$

The matrices $\boldsymbol{L}+y\boldsymbol{R}$ and $\boldsymbol{C}+y\boldsymbol{G}$, denoted by $\boldsymbol{Z}$ and $\boldsymbol{Y}$, respectively, are symmetric, hence diagonalizable. Let us consider the following similarity transformation on $\boldsymbol{K}(y)$

$$\begin{aligned} & \boldsymbol{Z}^{-\frac{1}{2}}\boldsymbol{K}(y)\boldsymbol{Z}^{\frac{1}{2}} \\ = \; & \boldsymbol{Z}^{-\frac{1}{2}}(\boldsymbol{Z}\boldsymbol{Y})\boldsymbol{Z}^{\frac{1}{2}} \\ = \; & \boldsymbol{Z}^{\frac{1}{2}}\boldsymbol{Y}\boldsymbol{Z}^{\frac{1}{2}}. \end{aligned} \tag{G.2}$$

Since $\boldsymbol{K}(y)$ is similar to $\boldsymbol{Z}^{\frac{1}{2}}\boldsymbol{Y}\boldsymbol{Z}^{\frac{1}{2}}$ and the later is is symmetric, $\boldsymbol{K}(y)$ is always diagonalizable.

## G.3 The entries of $\boldsymbol{\Lambda}^2(0)$ will all be positive.

$\boldsymbol{\Lambda}^2(0)$ are the eigenvalues of the product $\boldsymbol{L}\boldsymbol{C}$. Due to the positive energy reasoning, both $\boldsymbol{L}$ and $\boldsymbol{C}$ are positive definite [50]. Let us consider the similarity transformation on $\boldsymbol{L}\boldsymbol{C}$ below

$$\begin{aligned} & \boldsymbol{L}^{-\frac{1}{2}}(\boldsymbol{L}\boldsymbol{C})\boldsymbol{L}^{\frac{1}{2}} \\ = \; & \boldsymbol{L}^{\frac{1}{2}}\boldsymbol{C}\boldsymbol{L}^{\frac{1}{2}}. \end{aligned} \tag{G.3}$$

Since $\boldsymbol{L}^{\frac{1}{2}}\boldsymbol{C}\boldsymbol{L}^{\frac{1}{2}}$ will have all its eigenvalues positive, the entries of $\boldsymbol{\Lambda}^2(0)$ will be positive.

# Bibliography

[1] R. E. Bryant. A switch-level model and simulator for mos digital systems. *IEEE Trans. on Computers*, C-33:160–177, February 1984.

[2] L. W. Nagel. Spice2: A computer program to simulate semiconductor circuits. In *Electronics Research Laboratory Rep. No. ERL-M520, University of California, Berkeley*, May 1975.

[3] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Mass.: Addison-Wesley, 1990.

[4] Shen Lin, M. Marek-Sadowska, and Ernest Kuh. Swec: A stepwise equivalent conductance timing simulator for cmos vlsi circuits. In *Proc. of EDAC*, pages 142–148, February 1991.

[5] Shen Lin and Ernest Kuh. A new approach to circuit simulator. In *Proc. of the 10th European Conference on Circuit Theory and Design*, pages 264–273, July 1991.

[6] J. White and A. Sangiovanni-Vincentelli. Relax2.1 : A waveform relaxation based circuit simulation program. In *Proc. Custom Integrated Circuits Conf.*, pages 232–236, 1984.

[7] R. A. Saleh and A. R. Newton. The exploitation of latency and multirate behavior using nonlinear relaxation for circuit simulation. *IEEE Trans. on Computer-Aided Design of ICAS*, pages 1286–1298, December 1989.

[8] R. L. Baner, A. Ng J. Fang, and R. K. Brayton. Xpsim : A mos vlsi simulator. In *Proc. of IEEE ICCAD*, pages 66–69, November 1988.

[9] C. Visweswariah and R. A. Rohrer. Piecewise approximate circuit simulation. *IEEE Trans. on Computer-Aided Design of ICAS*, pages 861–870, October 1991.

[10] Shen Lin, Ernest Kuh, and M. Marek-Sadowska. Stepwise equivalent conductance circuit simulation technique. *IEEE Trans. on Computer-Aided Design of ICAS.* to appear.

[11] J. S. Roychowdhury and D. O. Pederson. Efficient transient simulation of lossy interconnect. In *Proc. of the 28th Design Automation Conference*, pages 740–745, June 1991.

[12] R. A. Struble. *Nonlinear Differential Equations.* New York: McGraw-Hill, 1962.

[13] L. O. Chua and P. M. Lin. *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques.* Prentice-Hall, 1975.

[14] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods.* New York: Wiley, 1966.

[15] A. R. Newton and A. Sangiovanni-Vincentelli. Relaxation based electrical simulation. *IEEE Trans. on Computer-Aided Design of ICAS*, pages 308–330, October 1984.

[16] J. E. Kleckner, R. A. Saleh, and A. R. Newton. Electrical consistency in schematic simulation. In *Proc. Int. Conf. on Circuits and Computers*, October 1983.

[17] R. Saleh and A. R. Newton. *Mixed-Mode Simulation.* Kluwer Academic Publishers, 1990.

[18] E. Lelarasmee, A. E. Ruehli, and A. Sangiovanni-Vincentelli. The waveform relaxation method for the time-domain analysis of large scale integrated circuits. *IEEE Trans. on Computer-Aided Design of ICAS*, pages 131–145, July 1982.

[19] D. J. Erdman and D. J. Rose. Newton waveform relaxation technique for tightly coupled system. *IEEE Trans. on Computer-Aided Design of ICAS*, CAD-11, 1992.

[20] D. Overhauser and I. Hajj. Idsim2: An environment for mixed-mode simulation. In *Proc. of IEEE Custom Integrated Circuits Conf.*, pages 5.2.1–5.2.4, 1990.

[21] P. Debefve, H. Y. Hsieh, and A. E. Ruehli. Wave convergence algorithms for the waveform relaxation method. In *Proc. of ICCAD*, pages 33–35, November 1984.

[22] B. R. Chawla, H. K. Gummel, and P. Kozak. Motis-an mos timing simulator. *IEEE Trans. on Circuits and Systems*, pages 901–910, October 1975.

[23] B. D. Ackland and R. Clark. Event-emu: An event-driven timing simulator for mos vlsi circuit. In *Proc. of IEEE ICCAD*, pages 80–83, 1989.

[24] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. on Computer-Aided Design of ICAS*, CAD-9:352–366, April 1990.

[25] X. Huang, V. Raghavan, and R. A. Rohrer. Awesim: A program for the efficient analysis of linear(ized) circuits. In *Proc. of ICCAD*, pages 534–537, November 1990.

[26] J. K. Ousterhout. A switch-level timing verifier for digital mos vlsi. *IEEE Trans. on Computer-Aided Design of ICAS*, CAD-4:336–349, July 1985.

[27] N. P. Jouppi. Timing analysis and performance improvement of mos vlsi designs. *IEEE Trans. on Computer-Aided Design of ICAS*, CAD-6:650–665, July 1987.

[28] Christopher J. Terman. Rsim - a logic-level timing simulator. In *Proc. of IEEE ICCAD*, pages 437–440, November 1983.

[29] J. Rubinstein, Jr. P. Penfield, and M. A. Horowitz. Signal delay in rc tree networks. *IEEE Trans. on Computer-Aided Design of ICAS*, CAD-2:202–211, July 1983.

[30] J. Katzenelson. An algorithm for solving nonlinear resistive network. *Bell System Tech. J.*, 44:1605–1620, July 1965.

[31] M.T. van Stiphout, J.T.J. van Eijndhoven, and H.W. Buurman. Plato: A new piecewise linear simulation tool. In *Proc. of EDAC*, pages 235–239, March 1990.

[32] Y. H. Kim, S. H. Hwang, and A. R. Newton. Electrical-logic simulation and its application. *IEEE Trans. on Computer-Aided Design of ICAS*, pages 8–22, January 1989.

[33] F. Y. Chang. Waveform relaxation analysis of rlgc transmission lines. *IEEE Trans. on Circuits and Systems*, CAS-37:1394–1415, November 1992.

[34] A. R. Djordjevi'c, T. K. Sarkar, and R. F. Harrington. Analysis of lossy transmission lines with arbitrary nonlinear terminal networks. *IEEE Trans. on Microwave Theory Tech.*, MTT-34:660–666, June 1986.

[35] J. R. Griffith and M. S. Nakhla. Time-domain analysis of lossy coupled transmission lines. *IEEE Trans. on Microwave Theory Tech.*, MTT-38:1480–1487, October 1990.

[36] J. E. Schutt-Aine and R. Mittra. Scattering parameter transient analysis of transmission lines loaded with nonlinear terminations. *IEEE Trans. on Circuits and Systems*, MTT-36:529–536, April 1988.

[37] D. Winklestein, M. B. Steer, and R. Pomerleau. Simulation of arbitrary transmission line network with nonlinear terminations. *IEEE Trans. on Circuits and Systems*, MTT-38:418–422, April 1991.

[38] J. S. Roychowdhury, A. R. Newton, and D. O. Pederson. An impulse-response based linear time-complexity algorithm for lossy interconnect simulation. In *Proc. of IEEE ICCAD*, pages 62–65, November 1991.

[39] R. Wang and O. Wing. Analysis of vlsi multiconductor systems by bi-level waveform relaxation. In *Proc. of IEEE ICCAD*, pages 161–169, November 1991.

[40] J. White and A. Sangiovanni-Vincentelli. *Relaxation techniques for the simulation of VLSI Circuits.* Kluwer Academic Publishers, 1987.

[41] S. P. McCormick and J. Allen. Waveform moment methods for improved interconnection analysis. In *Proc. of the 27th Design Automation Conference*, pages 406–412, June 1990.

[42] T. K. Tang and M. S. Nakhla. Analysis of high-speed vlsi interconnects using the asymptotic waveform evaluation technique. In *Proc. of IEEE ICCAD*, pages 542–545, November 1990.

[43] M. M. Alaybeyi, J. E. Bracken, J. Y. Lee, V. Raghavan, R. J. Trihy, and R. A. Rohrer. Analysis of mcms using asymptotic waveform evaluation (awe). In *Proc. IEEE Multi-Chip Module Conference*, pages 48–51, March 1992.

[44] F. Romeo and M. Santomauro. Time-domain simulation of n coupled transmission lines. *IEEE Trans. on Microwave Theory Tech.*, MTT-36:131–136, February 1987.

[45] D. Gao, A. T. Yang, and S. M. Kang. Modeling and simulation of interconnection delays and crosstalks in high-speed integrated circuits. *IEEE Trans. on Circuits and Systems*, CAS-37:1–9, January 1990.

[46] J. E. Schutt-Aine and R. Mittra. Nonlinear transient analysis of coupled transmission lines. *IEEE Trans. on Microwave Theory Tech.*, MTT-36:959–967, June 1989.

[47] N. S. Nahman and D. R. Holt. Transient analysis of coaxial cables using the skin effect approximation $a+b\sqrt{s}$. *IEEE Trans. on Circuits and Systems*, CAS-19:443–451, April 1972.

[48] K. Singhal and J. Vlach. *Computer methods for circuit analysis and design.* New York : Van Nostrand Reinhold, 1983.

[49] J.E. Schutt-Aine. Transient analysis of nonuniform transmission lines. *IEEE Trans. on Circuits and Systems*, CAS-39, 1992.

[50] R. Plonsey and R. E. Collin. *Principles and Application of Electromagnetic Fields.* New York : McGraw-Hill, 1961.

[51] A. Semlyen and A. Dabuleanu. Fast and accurate switching transient calculation on transmission lines with ground return using recursive convolution. *IEEE Trans. on Power Apparatus and Systems*, PAS-94:561–571, 1975.

[52] R. L. Fox and M. P. Kapoor. Rates of change of eigenvalues and eigenvectors. *AIAA Journal*, 6:2426–2429, December 1968.