Wojciech Maly (M'80) received the M.Sc. degree in electrical engineering from the Technical University of Warsaw, Poland, in 1970, and Ph.D. degree from the Institute of Applied Cybernetics, Polish Academy of Sciences, in 1975.

From 1970 to 1973 he was with the Institute of Applied Cybernetics, Polish Academy of Sciences where he was involved in the development of the control theory applications. In 1973 he joined Technical University of Warsaw, where he was appointed Assistant Professor in 1975. In 1977 he spent 8 months on leave at the Research and Production Center of Semiconductor Devices, Warsaw, Poland. Since September 1979 till July 1981, he has been a Visiting Assistant Professor of Electrical Engineering Department at Carnegie-Mellon University, Pittsburgh, PA. His research interests are in computer aided design of integrated circuits and diagnosis, control and simulation.

Andrzej J. Strojwas was born in Poland, on August 20, 1952. He received the M.S. degree (with honors) in electronic engineering from the Technical University of Warsaw, Poland, in September 1976. In 1975 he received the first prize in the Copernicus Competition for the best student in the University. Since October 1976 he has been a member of the faculty of the Technical University of Warsaw. He is currently on leave from this university to pursue the Ph.D. degree in electrical engineering from Carnegie-Mellon University, Pittsburgh, PA. His present research interests include statistical design of integrated circuits, and IC fabrication process modeling and diagnosis.

*

The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits

EKACHAI LELARASMEE, STUDENT MEMBER, IEEE, ALBERT E. RUEHLI, SENIOR MEMBER, IEEE, AND ALBERTO L. SANGIOVANNI-VINCENTELLI, SENIOR MEMBER, IEEE

***Abstract*—The Waveform Relaxation (WR) method is an iterative method for analyzing nonlinear dynamical systems in the time domain. The method, at each iteration, decomposes the system into several dynamical subsystems each of which is analyzed for the entire given time interval. Sufficient conditions for convergence of the WR method are proposed and examples in MOS digital integrated circuits are given to show that these conditions are very mild in practice. Theoretical and computational studies show the method to be efficient and reliable.**

I. Introduction

THE spectacular growth in the scale of integrated circuits being designed in the VLSI era has generated the need for new methods of circuit simulation. "Standard" circuit simulators, such as SPICE2 [1] and ASTAP [2], simply take too much CPU time and too much storage to analyze a VLSI circuit. These standard circuit simulators are essentially based on three techniques:

(i) Stiffly stable implicit integration methods, such as Backward Euler formula, for obtaining a system of nonlinear algebraic equations from the original system of nonlinear algebraic-differential equations describing the behavior of the circuit.

(ii) Newton-Raphson iteration to linearize the system of nonlinear algebraic equations of (i).

(iii) Sparse Gaussian elimination to solve the system of linear algebraic equations of (ii).

New simulators, such as MOTIS [3], SPLICE [4], [5], DIANA [6], and MACRO [7], in their quest for speed have rejected one or more of the principal features of standard simulators. In particular MOTIS and the timing simulation part of the mixed-mode simulator SPLICE use Backward Euler integration and relaxation techniques to decompose and solve the system of nonlinear algebraic equations, eliminating the need for sparse Gaussian elimination. The decomposition achieved by relaxation allows the use of selective trace algo-

Manuscript received October 26, 1981. This work was supported in part by the Joint Services Electronics Program under Contract F49620-79-C-0178, the Defense Advanced Research Projects Agency (DOD) under Contract N00039-K-0251, and the National Science Foundation under Grant ECS-79-13148.

E. Lelarasmee and A. L. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Sciences, and the Electronics Research Laboratory, University of California, Berkeley, CA 94720.

A. E. Ruehli is with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

0278-0070/82/0700-0131$00.75

rithms for exploiting the "latency" of VLSI circuits, a fundamental feature of SPLICE. However, in MOTIS and SPLICE, the relaxation process is not carried out to convergence: only one "sweep" of relaxation is taken [3]-[5]. Therefore, the numerical properties, such as stability and accuracy, of the Backward Euler integration no longer hold. It turns out that timing simulation techniques tend to have poor numerical properties when dealing with MOS circuits containing floating capacitors[1] and/or trees of MOS pass transistors (transmission gates). Moreover, in some cases, the size of the timestep has to be chosen very small to avoid instability and inaccuracy of the solution. A mathematical study and characterization of this technique is given in [8], where it is also shown that a method proposed by Kahan [11] can be used for timing simulation of MOS circuits yielding better stability and accuracy properties.

The Waveform Relaxation (WR) method is introduced in this paper for the analysis of large-scale circuits. The basic idea here is to apply relaxation directly to the system of nonlinear algebraic-differential equations describing the circuit. As a result, the system is decomposed into decoupled subsystems of algebraic-differential equations corresponding to decoupled dynamical subcircuits. Each decoupled subcircuit is then analyzed for the entire simulation time interval by means of standard simulation techniques, i.e., stiffly stable integration method and Newton-Raphson iteration. The decomposition achieved allows latency to be exploited in the most natural way.

This paper is organized as follows. Sections II and III describe the WR method and its convergence theorem in a mathematical framework. The rest of the paper describes the method in the circuit simulation context. In Section IV, the method is specialized to the analysis of VLSI MOS circuits. Two WR algorithms are described and are shown to be convergent under very mild and realistic assumptions. In Section V, the computational aspects of WR are studied with the help of an experimental WR circuit simulator: RELAX, the results being compared to those obtained from a standard simulator: SPICE. Finally, Section VI discusses how to improve the computational efficiency of the WR method.

II. Mathematical Formulation and the WR Algorithm Model

We consider dynamical systems which can be described by a system of mixed implicit algebraic-differential equations of the form

$$F(\dot{y}, y, u) = 0 \tag{2.1a}$$

$$E(y(0) - y_0) = 0 \tag{2.1b}$$

where $y(t) \in \boldsymbol{R}^p$ is the vector of unknown variables at time t, $\dot{y}(t) \in \boldsymbol{R}^p$ is the time derivative of y at time t, $u(t) \in \boldsymbol{R}^r$ is the vector of input variables at time t, $y_0 \in \boldsymbol{R}^p$ is the given initial value of y, F: $\boldsymbol{R}^p \times \boldsymbol{R}^p \times \boldsymbol{R}^r \to \boldsymbol{R}^p$ is a continuous function, and $E \in \boldsymbol{R}^{n \times p}$, $n \leqslant p$ is a matrix of rank n such that $Ey(t)$ is the state of the system at time t.

[1] A floating capacitor is a capacitor between two unknown nodes of a circuit.

Note that (2.1b) is meant to supply the initial conditions for the state variables of (2.1a). We shall assume that y_0 is chosen so as to give $y(0) = y_0$, i.e., y_0 also satisfies all the algebraic relations embedded in (2.1a). For example, in circuit simulation y_0 is usually obtained from the so-called "dc" solution of the system, i.e., it satisfies

$$F(\dot{y}(0), y_0, u(0)) = 0, \quad \dot{y}(0) = 0. \tag{2.2}$$

The general structure of a WR algorithm for analyzing (2.1) in a given time interval $[0, T]$ consists of two major processes, namely the *assignment-partitioning* process and the *relaxation* process.

A. The Assignment-Partitioning Process

In the assignment-partitioning process, each unknown variable is assigned to an equation of (2.1a) in which it is involved. However, no two variables can be assigned to the same equation. Then (2.1a) is partitioned into m disjoint[2] subsystems of equations, each of which may have only differential equations or only algebraic equations or both. Then, without loss of generality, we can rewrite (2.1) after being processed by the assignment-partitioning process as follows:

$$\begin{bmatrix} F_1(\dot{y}_1, y_1, d_1, u) \\ \vdots \\ F_m(\dot{y}_m, y_m, d_m, u) \end{bmatrix} = 0 \tag{2.3a}$$

$$E(y(0) - y_0) = 0 \tag{2.3b}$$

where, for $i = 1, 2, \cdots, m$, $y_i \in \boldsymbol{R}^{p_i}$ is the subvector of unknown variables assigned to the ith partitioned subsystem, F_i: $\boldsymbol{R}^{p_i} \times \boldsymbol{R}^{p_i} \times \boldsymbol{R}^{2p-2p_i} \times \boldsymbol{R}^r \to \boldsymbol{R}^{p_i}$ is a continuous function, and

$$d_i = \text{col}\,(y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_m, \dot{y}_1, \cdots, \dot{y}_{i-1}, \dot{y}_{i+1}, \cdots, \dot{y}_m)^*. \tag{2.3c}$$

It is clear that if the vectors d_i, $i = 1, 2, \cdots, m$, are treated as inputs, then (2.3a) can be solved by solving m independent subsystems. Therefore, they are called the *decoupling* vectors of the subsystems. This gives rise to the notion of the decomposed system as given in the following definition.

Definition 2.1. The *decomposed system* associated with an assignment-partitioning process applied to (2.1) consists of m independent subsystems, called *decomposed subsystems*, each of which is described by

$$F_i(\dot{y}_i, y_i, \tilde{u}_i, u) = 0 \tag{2.4a}$$

$$E_i(y_i(0) - y_{0i}) = 0 \tag{2.4b}$$

where $y_{0i} \in \boldsymbol{R}^{p_i}$ is the subvector of the given initial vector y_0, $\tilde{u}_i \in \boldsymbol{R}^{2p-2p_i}$ is the vector of the decoupling inputs, and

[2] There are cases in which the algorithm has better convergence properties if the subsystems are nondisjoint. For such cases, we can consider the nondisjoint subsystems as being obtained from partitioning an augmented system of equations with an augmented set of unknown variables.

*col $(a, b) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$

$E_i \in \boldsymbol{R}^{n_i \times p_i}$, $n_i \leq p_i$ is a matrix of rank n_i such that $E_i y_i$ is a state vector of the ith decomposed subsystem.

B. The Relaxation Process

The relaxation process is an iterative process. For simplicity we shall consider two most commonly used types of relaxation namely the *Gauss-Seidel* (GS) [15] and the *Gauss-Jacobi* (GJ) [15] relaxation. The relaxation process starts with an initial guess of the waveform solution of the original dynamical equations (2.1) in order to initialize the approximate waveforms of the decoupling vectors. During each iteration, each decomposed subsystem is solved for its assigned variables in the given time interval $[0, T]$ by using the approximate waveform of its decoupling vector. For the GS relaxation, the waveform solution obtained by solving one decomposed subsystem is immediately used to update the approximate waveforms of the decoupling vectors of the other subsystems. For the GJ relaxation, all waveforms of the decoupling vectors are updated at the beginning of the next iteration. The relaxation process is carried out repeatedly until satisfactory convergence is achieved. ■

Let the superscript index k denote the iteration count. Then the general structure of a WR algorithm can be formally described as follows:

WR Algorithm Model 2.1

Step 0: (Assignment-partitioning process)

Assign the unknown variables to equations in (2.1) and partition (2.1) into m subsystems of equations as given by (2.3).

Step 1: (Initialization of the relaxation process)

Set $k = 1$ and guess an initial waveform $(y^0(t); t \in [0, T])$ such that $y^0(0) = y(0) = y_0$.

Step 2: (Analyzing the decomposed system at the kth iteration)

For each $i = 1, 2, \cdots, m$, set

$$d_i^k = \text{col}\,(y_1^k, \cdots, y_{i-1}^k, y_{i+1}^{k-1}, \cdots, y_m^{k-1}, \dot{y}_1^k, \cdots, \dot{y}_{i-i}^k, \dot{y}_{i+1}^{k-1}, \cdots, \dot{y}_m^{k-1})$$

for the GS relaxation, or

$$d_i^k = \text{col}\,(y_1^{k-1}, \cdots, y_{i-1}^{k-1}, y_{i+1}^{k-1}, \cdots, y_m^{k-1}, \dot{y}_1^{k-1}, \cdots, \dot{y}_{i-1}^{k-1}, \dot{y}_{i+1}^{k-1}, \cdots, \dot{y}_m^{k-1})$$

for the GJ relaxation, and solve for $(y_i^k(t); t \in [0, T])$ from

$$F_i(\dot{y}_i^k, y_i^k, d_i^k, u) = 0 \tag{2.5a}$$

$$E_i(y_i^k(0) - y_i(0)) = 0. \tag{2.5b}$$

Step 3: (Iteration)

Set $k = k + 1$ and go to Step 2.

Remarks.

1) A simple guess for $(y^0(t); t \in [0, T])$ is $y^0(t) = y(0)$ for all $t \in [0, T]$.

2) In the actual implementation, the relaxation process stops when the difference between $(y^k(t); t \in [0, T])$ and $(y^{k-1}(t); t \in [0, T])$, i.e., $\max_{t \in [0, T]} \|y^k(t) - y^{k-1}(t)\|$, is sufficiently small.

3) In analogy to the classical relaxation methods for solving linear or nonlinear algebraic equations (see [15] for examples), it is possible to modify a WR algorithm by using a relaxation parameter $\omega \in (0, 2)$. With ω, the iteration equation (2.5) is modified to yield

$$F_i(\dot{\tilde{y}}_i^k, \tilde{y}_i^k, d_i^k, u) = 0 \tag{2.6a}$$

$$E_i(\tilde{y}_i^k(0) - y_i(0)) = 0 \tag{2.6b}$$

$$y_i^k = y_i^{k-1} + \omega(\tilde{y}_i^k - y_i^{k-1}). \tag{2.6c}$$

4) Note the following two important characteristics of the WR Algorithm Model 2.1.

a) The analysis of the original system is decomposed into the independent analysis of m subsystems.

b) The relaxation process is carried out on the entire waveforms, i.e., during each iteration each sybsystem is individually analyzed for the entire given time interval $[0, T]$. ■

III. Convergence of WR Method

In this section we shall derive sufficient conditions to guarantee that the WR Algorithm Model 2.1 converges, i.e., it generates a converging sequence of iterated solutions whose limit satisfies the dynamical equations (2.1a) and the given initial conditions (2.1b). In general, results on the convergence of an iterative method are stated when the iteration equation is written in an explicit form, i.e., the iterated variables are written as functions of their previous values and other noniterated variables. Hence, we shall introduce an explicit form of the iteration equation (2.5) of the WR Algorithm Model 2.1 and refer to it as the canonical representation of the algorithm. Sufficient conditions to ensure the existence of the canonical WR algorithm will also be given. These conditions are basically related to how the assignment-partitioning process treats the state variables of the original dynamical system and give rise to the following definition of the consistency of an assignment-partitioning process.

Definition 3.1. An assignment-partitioning process is said to be *consistent with a given dynamical system* (2.1) if the choice of the state vector of its associated decomposed system (2.4) is also a valid choice of the state vector of the original system (2.1), i.e., there exist matrices $E_1, E_2, \cdots, E_m$ and E, as defined in (2.4) and (2.1), such that

$$\text{col}\,[E_1 y_1, E_2 y_2, \cdots, E_m y_m] = Ey.$$

A WR algorithm which uses a consistent assignment-partitioning process is called a *consistent WR algorithm*. ■

Note that in the WR Algorithm Model 2.1, only the state variables of the decomposed system are initialized at time $t = 0$ by the given initial conditions of the original system. The initial values of other variables of the decomposed system (except the inputs u) can vary from one iteration to the other. However, due to the definition of the state variables, for a consistent WR algorithm, when the sequence of iterated solutions converges, the sequence of initial values of the nonstate variables also converges to the values given by the initial conditions of the original system. On the other hand, for an inconsistent WR algorithm, it is possible that the sequence of iterated solutions converges to a limit which satisfies the origi-

nal dynamical equations with another set of initial conditions. An example of this phenomenon is given in the Appendix. The same example also indicates that inconsistent WR algorithms tend to have poor convergence properties. Therefore, throughout this paper, we consider only consistent WR algorithms.

In most engineering designs, the assignment-partitioning process can be guided by the physical interpretation of the state of the dynamical system. For example, the state variables of lumped electrical circuits are usually the voltages across the capacitors and the currents through the inductors. Based on this physical interpretation, a designer can select an assignment-partitioning process which is consistent with the system. In fact, as we shall see later, for large-scale integrated circuits, there are automatic procedures which guarantee the consistency of the WR algorithms. Details about how to verify consistency of an assignment-partitioning process in a general case are given in [16].

We now give the definition of the canonical form of a WR algorithm, followed by conditions which guarantee that the WR Algorithm Model 2.1 can be transformed into a canonical form. It should be noted that we do not have to perform the transformation explicitly in order to implement the WR Algorithm Model 2.1. Based on its canonical form, we can state sufficient convergence conditions of the WR Algorithm Model 2.1 in a simplified form as given in Theorem 4.1.

Definition 3.2. A *canonical* WR algorithm is characterized by the following iteration equations:

$$\dot{x}^k = f(x^k, x^{k-1}, \dot{x}^{k-1}, z^{k-1}, u) \tag{3.1a}$$

$$z^k = g(x^k, x^{k-1}, \dot{x}^{k-1}, z^{k-1}, u) \tag{3.1b}$$

where $x \in \boldsymbol{R}^n$, $z \in \boldsymbol{R}^l$, $u \in \boldsymbol{R}^r$, and f, g are continuous functions. ■

Lemma 3.1. Assume that for each decomposed subsystem described by (2.4), there exist functions $\tilde{f}_i$ and $\tilde{g}_i$ and a nonsingular matrix

$$\begin{bmatrix} E_i \\ D_i \end{bmatrix} \in \boldsymbol{R}^{p_i \times p_i}$$

such that (2.4) can be rewritten as

$$\dot{x}_i = \tilde{f}_i(x_i, \tilde{u}_i, u) \tag{3.2a}$$

$$z_i = \tilde{g}_i(x_i, \tilde{u}_i, u) \tag{3.2b}$$

$$x_i(0) = E_i(y_i(0)) \tag{3.2c}$$

$$\begin{bmatrix} x_i \\ z_i \end{bmatrix} = \begin{bmatrix} E_i \\ D_i \end{bmatrix} y_i \tag{3.2d}$$

where $x_i \in \boldsymbol{R}^{n_i}$, $z_i \in \boldsymbol{R}^{p_i - n_i}$, and $(\tilde{f}_i, \tilde{g}_i)$ are smooth[3] functions, i.e., each decomposed subsystem has a state-equation representation. If the WR Algorithm Model 2.1 is consistent, then it can be transformed into a canonical WR algorithm. ■

[3]By smooth we mean that the functions are r times continuously differentiable, where r is as large as required by the transformation identified in the proof of the lemma.

The proof of this lemma (given in [16]) is constructive and identifies such a transformation.

Theorem 3.1 (Convergence Theorem of WR Algorithms).

Consider the consistent WR Algorithm Model 2.1 which can be transformed into the following canonical form:

$$\dot{x}^k = f(x^k, x^{k-1}, \dot{x}^{k-1}, z^{k-1}, u) \tag{3.3a}$$

$$z^k = g(x^k, x^{k-1}, \dot{x}^{k-1}, z^{k-1}, u) \tag{3.3b}$$

$$x^k(0) = x(0) \tag{3.3c}$$

where $x \in \boldsymbol{R}^n$, $z \in \boldsymbol{R}^l$, and $u \in \boldsymbol{R}^r$. Assume that

a) $u(\cdot)$: $[0, T] \to \boldsymbol{R}^r$ is a given piecewise continuous[4] function;

b) there exist norms in $\boldsymbol{R}^n \times \boldsymbol{R}^l$ and $\boldsymbol{R}^n$, $\lambda_1 \geqslant 0$, $\lambda_2 \geqslant 0$, and $\gamma \in [0, 1)$ such that for any $a, b, s, \tilde{a}, \tilde{b}, \tilde{s} \in \boldsymbol{R}^n$, $v, \tilde{v} \in \boldsymbol{R}^l$, and $u \in \boldsymbol{R}^r$

$$\left\| \begin{matrix} f(a, b, s, v, u) - f(\tilde{a}, \tilde{b}, \tilde{s}, \tilde{v}, u) \\ g(a, b, s, v, u) - g(\tilde{a}, \tilde{b}, \tilde{s}, \tilde{v}, u) \end{matrix} \right\|$$

$$\leqslant \lambda_1 \|a - \tilde{a}\| + \lambda_2 \|b - \tilde{b}\| + \gamma \left\| \begin{matrix} s - \tilde{s} \\ v - \tilde{v} \end{matrix} \right\|$$

i.e., (f, g) is globally Lipschitz continuous with respect to x and globally contractive with respect to $(\dot{x}, z)$;

c) both f and g are continuous with respect to u.

Then, for any initial guess $(x^0(t), z^0(t); t \in [0, T])$ such that $\dot{x}^0(\cdot)$ and $z^0(\cdot)$ are piecewise continuous and $x^0(0) = x(0)$, the sequence $\{(\dot{x}^k(t), x^k(t), z^k(t); t \in [0, T])\}_{k=1}^{\infty}$ generated by the canonical WR algorithm (3.3) converges uniformly to $(\hat{x}(t), \hat{x}(t), \hat{z}(t); t \in [0, T])$ which satisfies

$$\dot{\hat{x}} = f(\hat{x}, \hat{x}, \dot{\hat{x}}, \hat{z}, u) \tag{3.4a}$$

$$\hat{z} = g(\hat{x}, \hat{x}, \dot{\hat{x}}, \hat{z}, u) \tag{3.4b}$$

$$\hat{x}(0) = x(0). \tag{3.4c}$$

■

Remark. We do not need condition (b) of Theorem 3.1 to hold for the entire spaces $\boldsymbol{R}^n \times \boldsymbol{R}^l$ and $\boldsymbol{R}^n$, i.e., global Lipschitz and global contractive properties of (f, g) are not necessary. The convergence is still guaranteed as long as the condition (b) holds for subsets of $\boldsymbol{R}^n \times \boldsymbol{R}^l$ and $\boldsymbol{R}^n$ containing the sequences $\{(\dot{x}^k(t), z^k(t); t \in [0, T])\}_{k=0}^{\infty}$ and $\{(x^k(t); t \in [0, T])\}_{k=0}^{\infty}$, respectively. ■

It is possible to justify intuitively the derivation of the convergence conditions given in Theorem 3.1 if one is familiar the contraction mapping theorem (see [15, p. 120]) and the Picard-Lindelof theorem on the existence and uniqueness of the solutions of ordinary differential equations (see [17, p. 18]). From the contraction mapping theorem, the conditions (b) and (c) guarantee that (3.4) can be written equiva-

[4]A function $u(\cdot)$: $[0, T] \to \boldsymbol{R}^r$ is piecewise continuous if it is continuous everywhere except at a finite number of points and at any discontinuity point, the function has finite left- and right-hand limits.

lently as

$$\dot{\hat{x}} = \hat{f}(\hat{x}, u) \tag{3.5a}$$

$$\hat{z} = \hat{g}(\hat{x}, u) \tag{3.5b}$$

$$\hat{x}(0) = x(0) \tag{3.5c}$$

where $\hat{f}$ and $\hat{g}$ are Lipschitz continuous with respect to $\hat{x}$ and continuous with respect to u. Hence, by the Picard-Lindelof theorem, (3.5) has a unique solution $(\hat{x}, \hat{z})$ for any given initial condition and any given piecewise-continuous input. Theorem 3.1 simply shows that the canonical WR algorithm is in fact a constructive proof of the existence and uniqueness of the solution.

We conclude this section with the remark that we have presented the WR method in this and the previous sections in a mathematical framework which is quite general. However, since we made no assumption about the structure of the dynamical equations (2.1), we had to make certain theoretical restrictions (such as consistency requirement and the existence of a canonical representation) in order to be able to derive some useful result, i.e., the convergence conditions of the WR method. This makes it difficult to interpret them in terms of physical properties. Moreover, there are many different ways to formulate the circuit differential equations, each differs from the other by the choice of the circuit variables, e.g., nodal analysis formulation, Modified nodal analysis formulation [12], and tableau formulation [14]. It is quite possible that the WR algorithms derived from different formulations of the circuit equations have different convergent properties. In the next section, we shall specialize the method to a particular formulation and a particular class of circuits. There we will see that the theoretical restrictions given in this section are automatically satisfied.

IV. Application of the WR Method for the Time-Domain Solutions of MOS Integrated Circuits

In this section we shall apply the WR Algorithm Model 2.1 to analyze an important class of dynamical systems: MOS digital integrated circuits. In fact, this was the original motivation behind the development of the WR method. A typical large scale digital circuit is usually an interconnection of several basic subcircuits called "gates." Hence decomposition techniques can be applied to the analysis of this class of circuits in the most natural way. We propose two WR algorithms for analyzing MOS digital integrated circuits and show that, under very mild assumptions usually satisfied by practical circuits, the proposed algorithms converge. Although both GS and GJ relaxations can be used in these algorithms, the GS relaxation is preferred since it requires only one copy of the iterated solution, as opposed to two copies required by the GJ relaxation, and its speed of convergence is faster, especially for MOS circuits where unidirectional models are used to model MOS devices (for example, see [3]-[5]), provided that the equations are properly ordered (see [10] for a discussion of this aspect). For the sake of simplicity, both algorithms use the simplest guessing scheme and an assignment-partitioning process in which each partitioned subsystem is a single equation. The generalization of both algorithms to allow more than one equation per subsystem is straightforward and will not be discussed.

Two basic assumptions are made to derive our algorithms.

(i) Each MOS device and its interconnections can be modeled by lumped (linear or nonlinear) voltage-controlled capacitors, conductors, and current sources.

(ii) Every (internal or external) node in the circuit has a (linear or nonlinear) capacitor, called a grounded capacitor, to either ground or dc supply voltage rails.

Note that for LSI MOS circuits, these assumptions are usually satisfied.

For the first algorithm, we use the node voltages as the circuit variables. Thus because of our assumptions, the circuit equations can be written as follows:

$$C(v, u)\,\dot{v} + q(v, u) = 0 \tag{4.1a}$$

$$v(0) = v_0 \tag{4.1b}$$

where $v \in \boldsymbol{R}^n$ is the vector of all unknown node voltages, v_0 is the given initial values of v, $u \in \boldsymbol{R}^r$ is the vector of all inputs and their time derivatives, q: $\boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^n$ is a continuous function each component of which represents the net sum of currents charging the capacitor at each node due to the conductors and the controlled-current sources, C: $\boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^{n \times n}$ is a symmetric diagonally dominant matrix-value function in which $-C_{ij}(v, u)$; $i \neq j$ is the total capacitance between nodes i and j, and $C_{ii}(v, u)$ is the sum of the capacitances of all capacitors connected to node i.

Algorithm 4.1 (WR algorithm for solving (4.1) from $t = 0$ to $t = T$)

Comment: Superscript denotes the iteration count.

Step 1: Set $k = 1$ and $v^0(t) = v(0)$ for all $t \in [0, T]$.

Step 2: For $i = 1, 2, \cdots, n$, solve for $\{v_i^k(t); t \in [0, T]\}$ from

$$\sum_{j=1}^{i} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\,\dot{v}_j^k + \sum_{j=i+1}^{n} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\,\dot{v}_j^{k-1} + q_i(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u) = 0 \tag{4.2}$$

with the initial condition $v_i^k(0) = v_i(0)$.

Step 3: Set $k = k + 1$ and go to Step 2. ■

Remarks.

1) Equation (4.2) is actually a single differential equation in one unknown variable v_i^k. The variables $v_{i+1}^{k-1}, \cdots, v_n^{k-1}$ are known from the previous iteration and $v_1^k, \cdots, v_{i-1}^k$ have already been computed.

2) In practice, the circuit equations (4.1) are usually very sparse, i.e., only a few variables are actually involved in each equation. This fact can be exploited in the implementation of the algorithm on a computer.

3) With regard to the theoretical concepts presented in the previous two sections, we can say that Algorithm 4.1 assigns v_i to the ith equation of (4.1) and the assigned variable be-

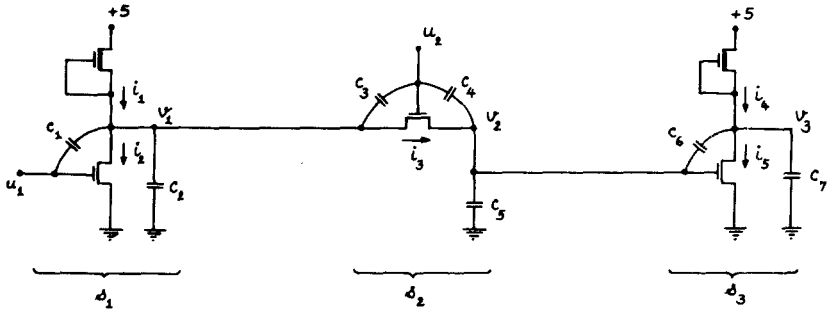


Fig. 1. An MOS dynamic shift register.

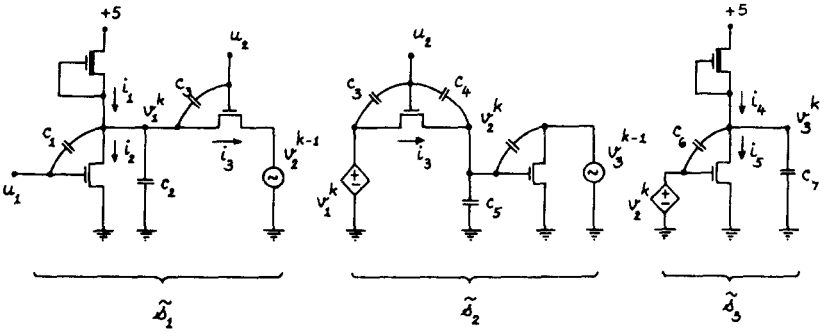


Fig. 2. The decomposition of the circuit in Fig. 1 at the kth iteration of Algorithm 4.1.

comes a state variable of the decomposed system (4.2). Furthermore, it is clear that v_i, $i = 1, 2, \cdots, n$ form a state vector of both (4.1) and (4.2), implying that the algorithm is consistent. ∎

Example 4.1

Consider the circuit shown in Fig. 1. For simplicity we assume that all capacitors are linear. Using the node voltages v_1, v_2, and v_3 as variables, the circuit equations are

$$(c_1 + c_2 + c_3)\dot{v}_1 - i_1(v_1) + i_2(v_1, u_1) + i_3(v_1, u_2, v_2) - c_1\dot{u}_1 - c_3\dot{u}_2 = 0 \quad (4.3a)$$

$$(c_4 + c_5 + c_6)\dot{v}_2 - c_6\dot{v}_3 - i_3(v_1, u_2, v_2) - c_4\dot{u}_2 = 0 \quad (4.3b)$$

$$(c_6 + c_7)\dot{v}_3 - c_6\dot{v}_2 - i_4(v_3) + i_5(v_3, v_2) = 0. \quad (4.3c)$$

Applying Algorithm 4.1, the kth iteration corresponds to solving the following equations:

$$(c_1 + c_2 + c_3)\dot{v}_1^k - i_1(v_1^k) + i_2(v_1^k, u_1) + i_3(v_1^k, u_2, v_2^{k-1}) - c_1\dot{u}_1 - c_3\dot{u}_2 = 0 \quad (4.4a)$$

$$(c_4 + c_5 + c_6)\dot{v}_2^k - c_6\dot{v}_3^{k-1} - i_3(v_1^k, u_2, v_2^k) - c_4\dot{u}_2 = 0 \quad (4.4b)$$

$$(c_6 + c_7)\dot{v}_3^k - c_6\dot{v}_2^k - i_4(v_3^k) + i_5(v_3^k, v_2^k) = 0. \quad (4.4c)$$

Fig. 2 shows the circuit interpretation of (4.4). From this figure we see that the algorithm decomposes the circuit, which is an interconnection of 3 subcircuits s_1, s_2, and s_3 into 3 augmented decomposed subcircuits $\tilde{s}_1$, $\tilde{s}_2$, and $\tilde{s}_3$, respectively. Subcircuit $\tilde{s}_1$ is first analyzed from $t = 0$ to $t = T$. Then $\tilde{s}_2$ is analyzed from $t = 0$ to $t = T$ by using the output v_1 of $\tilde{s}_1$ as its input. The kth iteration is then completed with the analysis of $\tilde{s}_3$ from $t = 0$ to $t = T$. ∎

The second algorithm is intended for MOS circuit containing pass transistors (or transmission gates) such as the circuit in Fig. 1. By choosing the node voltages and the drain currents of pass transistors as the circuit variables, the circuit equations can be formulated as follows:

$$C(v, u)\dot{v} + \tilde{q}(z, v, u) = 0 \quad (4.5a)$$

$$z - g(v, u) = 0 \quad (4.5b)$$

$$v(0) = v_0 \quad (4.5c)$$

where C, v, u, v_0 are as defined in (4.1), $z \in \boldsymbol{R}^l$ is the vector of drain currents of the pass transistors, g: $\boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^l$ is a continuous function (each component of which describes the drain current of each pass transistor in terms of its terminal node voltages), and $\tilde{q}$: $\boldsymbol{R}^l \times \boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^n$ is a continuous function (each component of which represents the net current charging the capacitor at each node due to the pass transistors, other conductive elements and controlled current sources).

Algorithm 4.2 (WR algorithm for solving (4.5) from $t = 0$ to $t = T$)

Comment: Superscript denotes the iteration count.

Step 1: Set $k = 1$, $z^0(t) = 0$, and $v^0(t) = v(0)$ for all $t \in [0, T]$.

Step 2: a) For $i = 1, 2, \cdots, n$, solve for $(v_i^k(t); t \in [0, T])$ from

$$\sum_{j=1}^{i} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\dot{v}_j^k + \sum_{j=i+1}^{n} C_{ij}(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)\dot{v}_j^{k-1} + \tilde{q}_i(z^{k-1}, v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u) = 0 \quad (4.6a)$$

with the initial condition $v_i^k(0) = v_i(0)$.

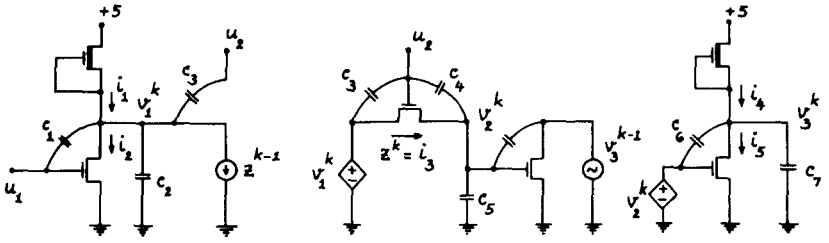


Fig. 3. The decomposition of the circuit in Fig. 1 at the kth iteration of Algorithm 4.2.

b) Compute $z^k(t); t \in [0, T]$ from

$$z^k = g(v^k, u). \tag{4.6b}$$

Step 3: Set $k = k + 1$ and go to Step 2. ■

Remarks.

1) Just like (4.1), (4.5) is also very sparse.

2) Equations (4.6a) and (4.6b) can actually be solved together as will be demonstrated in the following example.

3) By the same arguments as those for Algorithm 4.1, we can conclude that Algorithm 4.2 is also consistent with (4.5). ■

Example 4.2

Once again we consider the circuit of Fig. 1. This time we formulate the circuit equations as follows:

$$(c_1 + c_2 + c_3)\dot{v}_1 - i_1(v_1) + i_2(v_1, u_1) + z - c_1\dot{u}_1 - c_3\dot{u}_2 = 0 \tag{4.7a}$$

$$z - i_3(v_1, u_2, v_2) = 0 \tag{4.7b}$$

$$(c_4 + c_5 + c_6)\dot{v}_2 - c_6\dot{v}_3 - i_3(v_1, u_2, v_2) - c_4\dot{u}_2 = 0 \tag{4.7c}$$

$$(c_6 + c_7)\dot{v}_3 - c_6\dot{v}_2 - i_4(v_3) + i_5(v_3, v_2) = 0. \tag{4.7d}$$

Applying Algorithm 4.2, the kth iteration corresponds to solving the following equations:

$$(c_1 + c_2 + c_3)\dot{v}_1^k - i_1(v_1^k) + i_2(v_1^k, u_1) + z^{k-1} - c_1\dot{u}_1 - c_3\dot{u}_2 = 0 \tag{4.8a}$$

$$z^k - i_3(v_1^k, u_2, v_2^k) = 0 \tag{4.8b}$$

$$(c_4 + c_5 + c_6)\dot{v}_2^k - c_6\dot{v}_3^{k-1} - i_3(v_1^k, u_2, v_2^k) - c_4\dot{u}_2 = 0 \tag{4.8c}$$

$$(c_6 + c_7)\dot{v}_3^k - c_6\dot{v}_2^k - i_4(v_3^k) + i_5(v_3^k, v_2^k) = 0. \tag{4.8d}$$

Fig. 3 shows the circuit interpretation of (4.8). Note the difference between the decompositions induced by Algorithm 4.1 (Fig. 2) and Algorithm 4.2 (Fig. 3). From Fig. 3 we see that (4.8b) and (4.8c) can be solved together since they belong to the same subcircuit. ■

Theorem 4.1. Assume that

a) The charge-voltage characteristic of each capacitor, or the volt-ampere characteristic of each conductor, or the drain current characteristic of each MOS device is Lipschitz continuous with respect to its controlling variables,

b) $C_{\min} > 0$ and $C_{\max} < \infty$ where $C_{\min} \in \boldsymbol{R}$ is the minimum value of all grounded capacitances at any permissible values of voltages, and $C_{\max} \in \boldsymbol{R}$ is the maximum value of all capacitances between any two nodes at any permissible values of voltages,

c) The current through any controlled conductor (e.g., the drain current of an MOS device) is uniformly bounded throughout the relaxation process.

Then, for any MOS circuit with any given set of initial conditions, and any given piecewise-continuous input $u(\cdot)$, either Algorithm 4.1 or 4.2 generates a converging sequence of iterated solutions whose limit satisfies the dynamical equations of the circuit and the given set of initial conditions. ■

Note that the first assumption implies that for any capacitor, conductor, or MOS device, its incremetal (or small-signal) characteristic, i.e., capacitance, conductance or transconductance, at any permissible dc operating point must be uniformly bounded. The second assumption states that the value of any grounded capacitor must be bounded away from zero and the value of any floating capacitor must not be arbitrarily large. The third assumption implies that during the relaxation process, the current through any conductor or MOS device must not grow arbitrarily large. These three assumptions are very mild in practice and hence either Algorithm 4.1 or 4.2 is guaranteed to converge for any MOS integrated circuit of practical interest. The rate of convergence is linear, a typical property of any relaxation method.

It should be pointed out that the convergence of WR algorithms can be established by Theorem 4.1 for integrated circuits implemented in other technologies as long as the circuit equations can be written as in (4.1) and the assumptions of Theorem 4.1 on the branch equations are satisfied. Note also that the strong assumption of Theorem 3.1 regarding the global contractivity of f in (3.1) with respect to $\dot{x}$ is automatically satisfied because $C(\cdot, \cdot)$ is strictly diagonally dominant. Moreover, the contractivity of g in (3.1) with respect to z is irrelevant for Algorithm 4.1 since its canonical form does not involve algebraic equations. For Algorithm 4.2, it is also irrelevant since g does not depend on z. These results depend critically on Assumption (ii).

In both algorithms, the initial guesses are chosen, for convenience, to be constant waveforms. From Theorem 3.1 we know that other choices of initial guesses will not destroy the guaranteed convergence of both algorithms if they are piecewise continuous waveforms. Hence, for MOS digital integrated circuits, a logic simulation could be used to generate the initial guesses for these two algorithms. It is also possible to show that, under the same assumptions of Theorem 4.1, the corresponding GJ relaxation versions of Algorithms 4.1 and 4.2

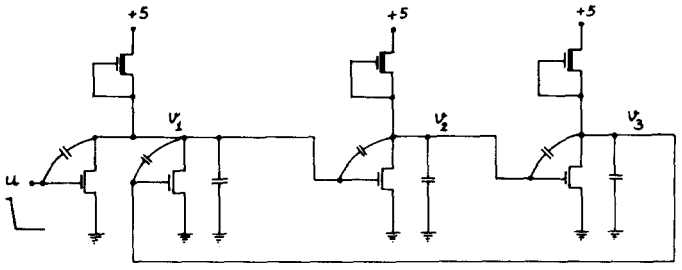


Fig. 4. An MOS ring oscillator.

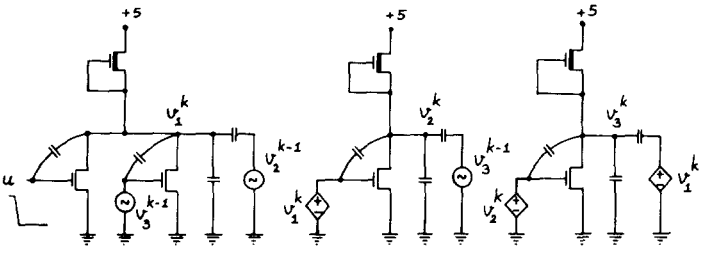


Fig. 5. The decomposition of the circuit in Fig. 4 at the kth iteration of Algorithm 4.1.

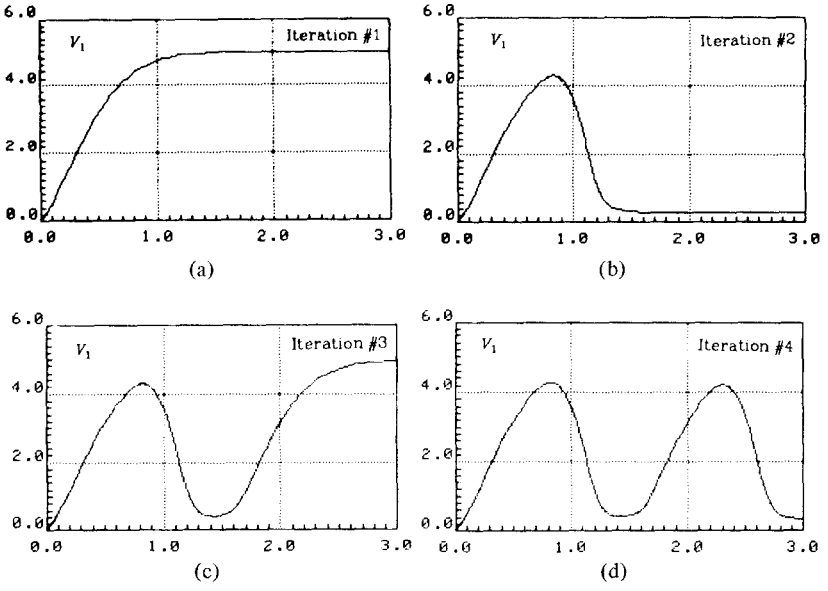


Fig. 6. (a) The waveform of v_1 of the circuit in Fig. 4 after the first WR iteration. (b) The waveform of v_1 of the circuit in Fig. 4 after the second WR iteration. (c) The waveform of v_1 of the circuit in Fig. 4 after the third WR iteration. (d) The waveform of v_1 of the circuit in Fig. 4 after the fourth WR iteration.

are guaranteed to converge. Moreover, a relaxation parameter ω can be introduced into these GJ- or GS-WR algorithms (as described in Section II) without destroying their guaranteed convergence, provided that $\omega \in (0, 2)$.

For example, the ring oscillator shown in Fig. 4 is used to illustrate the convergence of Algorithm 4.1. The circuit interpretation of the relaxation process is shown in Fig. 5. The resulting waveforms at different iterations of the algorithm are shown in Fig. 6(a)-(d). Note that since the oscillator is highly nonunidirectional due to the feedback from v_3 to the input of the NOR gate, the convergence of the iterated solutions is achieved with the number of iterations being proportional to the number of oscillating cycles of interest.

V. Experimental Program RELAX: A Computational Study of the WR Method

To study the computational properties of the WR method, an experimental Fortran program called RELAX has been written for simulating MOS digital integrated circuits. The algorithm implemented in RELAX is based on Algorithm 4.1 with the following major modifications.

1) RELAX allows each partitioned subsystem to have more than one equation so that each subsystem corresponds to a physical digital subcircuit, e.g., NOR, NAND, FLIP-FLOP, etc. Each decomposed subcircuit is solved by using standard simulation techniques, i.e., Backward Euler integration formula and Newton-Raphson method.

2) The first iteration of RELAX is essentially the first iteration of Algorithm 4.2, but after that RELAX switches back to use Algorithm 5.1 for the rest of the relaxation process. That is, in the first iteration of RELAX the drain currents of pass transisters do not contribute any loading effect on the subcircuits to which they are connected. This is done because, in the first iteration when all initial guesses are constant waveforms, a pass transistor can be driven continuously into its conductive region and may adversely effect the speed of convergence if its current is treated as a load of the other subcircuit.

Note that these modifications are just examples of specializing the WR method for a particular class of dynamical systems, exploiting their physical characteristics to improve the speed of convergence. In addition, RELAX incorporates two techniques to speed up the analysis of the decomposed circuit. The first technique is based on the latency of each decomposed subcircuit and is similar to the technique described in [7]. The second technique, which takes effect after the second iteration, is based on the partial convergence of iterated waveforms. That is, when the difference between a waveform at the current iteration and the same waveform at the previous iteration is within the specified convergence error over a subinterval of the analysis time, the waveform will not be recomputed for that subinterval in the next iteration if its associated input waveforms also converge during that subinterval. More details on these two techniques and on RELAX are given in [18].

Several MOS digital circuits have been analyzed by RELAX and the results have been compared with those obtained by two other simulation programs, SPICE [1] and SPLICE [4]. In these tests, SPLICE has been run as a timing simulator. All three programs use the Schichman-Hodges equation [9] to model the drain current of an MOS device and linear capacitors to model the charge storing mechanism of the device and its parasitic capacitances (see Fig. 7). Since the version of SPLICE used in this text did not allow the timing analysis of circuits containing floating capacitors, each test circuit has been analyzed with no floating capacitors for the comparison with SPLICE. Since all three programs use numerical integration methods with adjustable timesteps, the same maximum timestep, which is an input parameter of each program, is specified for the numerical integration routine of each program. For RELAX, the specified convergence error is 0.05 V, i.e., the relaxation process stops when the maximum difference of all voltage waveforms between the current iteration and the previous iteration is less than 0.05 V. The same convergence error is also used by RELAX to bypass unnecessary analysis when a partial convergence is detected.

The schematic diagram of MOS circuits being tested and their input waveforms are shown in Figs. (8a)-12(a). Comparisons of output waveforms obtained by RELAX and SPLICE are shown in Figs. 8(b)-12(b). For RELAX outputs, each rectangular mark denotes the computed value after every two internal timesteps to illustrate the effect of the latency technique implemented in the program. A comparison of the analysis time in CPU seconds spent by each program is given in Tables I and II. All three programs run on a VAX 11/780 using Berkeley VM UNIX[5] operating system. The tabulated figures do not include the time spent in the read-in, set-up, and read-out phases of each program. For RELAX, the tabulated figure is the sum of the analysis time spent at each iteration and hence the total number of iterations is also included. As seen from the tables, the analysis time of RELAX is at least one order of magnitude less than SPICE and of the same order of the timing simulation part of SPLICE. These are accounted for by the following factors.

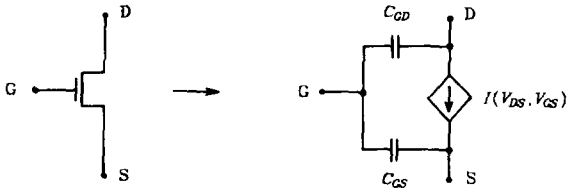


Fig. 7. MOS model used in the test circuits.

TABLE I
A COMPARISON OF THE ANALYSIS TIME BETWEEN SPICE AND RELAX

Circuit[1] of	Fig. 8	Fig. 9	Fig. 10	Fig. 11	Fig. 12
# of unknown nodes	4	8	16	27	45
# of MOS devices	6	21	42	131	263
CPU-SPICE (sec)	21.30	121.57	211.53	618.00	1334.80
CPU-RELAX (sec)	1.08	4.38	5.85	18.42	22.30
# of RELAX iterations	5	5	7	5	4
CPU-SPICE/CPU-RELAX	19.70	27.73	36.16	44.42	59.86

[1]With floating capacitors. The ratio of a floating capacitance to a grounded capacitance is approximately 1 to 12.

TABLE II
A COMPARISON OF THE ANALYSIS TIME BETWEEN SPLICE AND RELAX

Circuit[2] of	Fig. 8	Fig. 9	Fig. 10	Fig. 11	Fig. 12
# of unknown nodes	4	8	16	27	45
# of MOS devices	6	21	42	131	263
CPU-SPLICE (sec)	1.18	5.00	8.62	20.95	37.47
CPU-RELAX (sec)	0.53	4.68	7.62	27.00	44.73
# of RELAX iterations	2	3	7	2	2

[2]With no floating capacitors. Although the circuits and their input waveforms used in Tables I and II are the same, the initial conditions are different.

[5]UNIX is a trademark of Bell Laboratories.

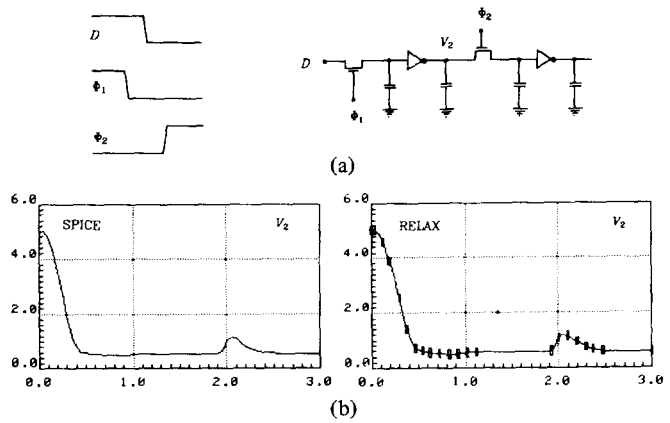


Fig. 8. (a) Schematic diagram of a dynamic shift register. (b) Output waveforms obtained by RELAX and SPICE.

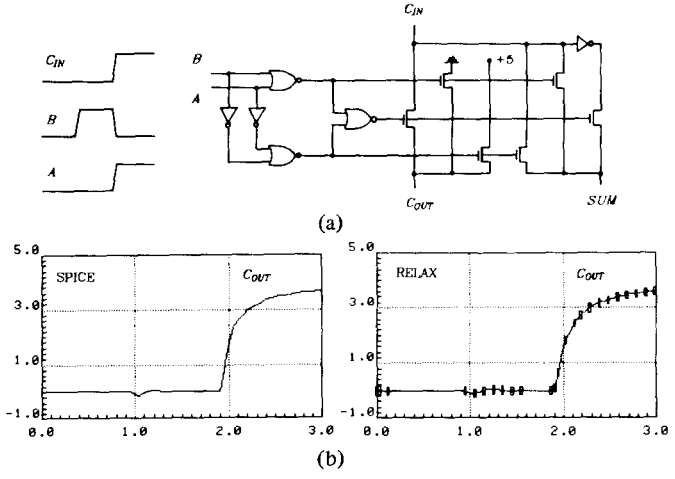


Fig. 9. (a) Schematic diagram of a one-bit full adder with a pass transistor carry-chain. (b) Output waveforms obtained by RELAX and SPICE.

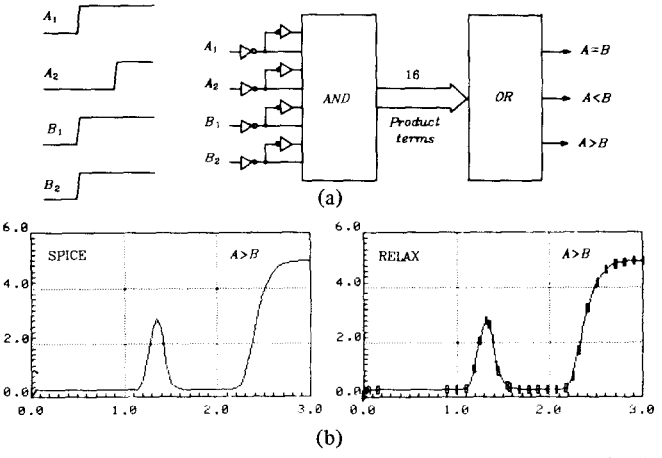


Fig. 11. (a) Schematic diagram of a 2-bit magnitude comparator implemented by a NOR-NOR PLA with no minimization of the product terms. (b) Output waveforms obtained by RELAX and SPICE.

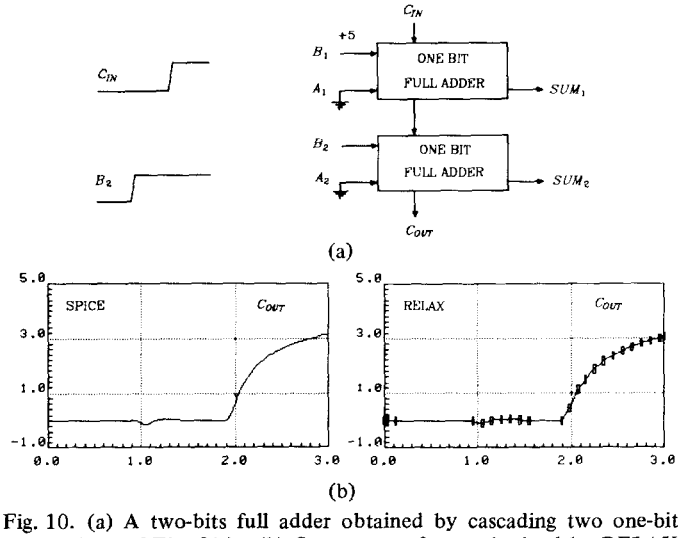


Fig. 10. (a) A two-bits full adder obtained by cascading two one-bit full adder of Fig. 8(a). (b) Output waveforms obtained by RELAX and SPICE.

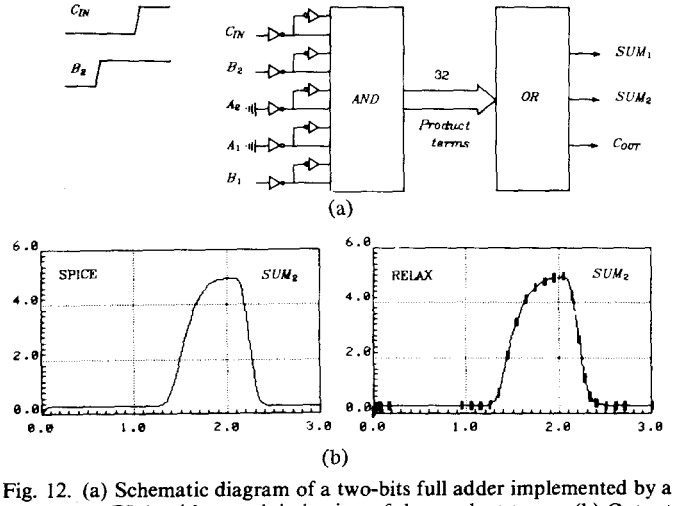


Fig. 12. (a) Schematic diagram of a two-bits full adder implemented by a NOR-NOR PLA with no minimization of the product terms. (b) Output waveforms obtained by RELAX and SPICE.

1) Both RELAX and SPLICE use decomposition techniques, although of different natures. It is known that the complexity of the analysis of a circuit without decomposition is proportional to n^m where n is the size, i.e., the number of unknown variables, of the circuit and m is usually between 1.2 and 2 depending on the sparseness of the circuit, i.e., m is small (large) if the number of interactions between the unknown variables is small (large). With decomposition, the complexity

is usually proportional to n. Hence, decomposition is a major factor in reducing the analysis time of a circuit. The larger the circuit, the more reduction is achieved, as shown in Table I.

1) Both RELAX and SPLICE incorporate latency techniques, although of different types. These techniques take advantage of the latency of a circuit avoiding unnecessary computations. RELAX also has an additional technique that takes advantage of the partial waveform convergence which enables it to reduce the analysis time of the decomposed circuit at later iterations.

3) Due to the decomposition used in RELAX and SPLICE each decomposed subcircuit can be analyzed with its own integration method and optimal timestep sequence. The number of timesteps per each subcircuit is, therefore, usually less than the one of SPICE, a definite computational advantage.

4) Both RELAX and SPLICE are specialized programs for analyzing MOS integrated circuits, as opposed to SPICE which is capable of analyzing circuits using different types of devices, e.g., MOS, bipolar transistors etc. Therefore, the overhead incurred by SPICE in order to be able to deal with different models and different devices is eliminated in RELAX and SPLICE.

As for a comparison between RELAX and SPLICE, it is important to note that while the analysis method in the timing simulation part of SPLICE is an "approximate" method that may suffer from instability and inaccuracy problems (see [4] and [8]), the WR method used in RELAX is reliable and accurate. Note also that, without floating capacitors, the circuits of Figs. 11(a) and 12(a) possess a unidirectional property which is referred to as *one-way* property in [10]. Circuits having unidirectional property can be analyzed exactly by performing only one iteration of GS relaxation. Since the present version of RELAX does not recognize this property, it has to perform two iterations before convergence can be detected.

VI. Conclusion and Discussion

The WR method is a new method for the time-domain analysis of large-scale dynamical systems based on the relaxation of the nonlinear algebraic-differential equations describing the system to be analyzed. Since the method decouples these equations, independent integration of each decomposed subsystem can be performed with different integration formulas and timesteps. In addition, latency of subsystems can be easily detected and exploited to reduce the analysis time. This method is particularly suitable for VLSI circuits. In comparison with the relaxation methods used by timing simulators, e.g., MOTIS and the timing simulation part of SPLICE, the WR method is guaranteed to converge for a wide class of circuits while sharing the main advantages of these techniques.

On the negative side, we need to store the waveforms at the current iteration for computing the waveforms at the next iteration. For large circuits and/or large time intervals, the amount of storage needed can be very large. However, due to the sparsity of the circuit equations, only a few waveforms are required in the analysis of each subcircuit. The waveforms can be stored in the secondary storage and brought back into the primary storage only when they are needed. To avoid delays due to the slow access time of the secondary storage, a buffering scheme [18] can be implemented to transfer the waveforms from primary to secondary storage and back without stopping the execution of the method. A common problem of relaxation methods shared by WR is that if there is a logic feedback path between the decomposed subcircuits, the speed of convergence can be very slow unless a good initial guess is provided. In our experiments, we found that an initial guess based on a logic simulation of the circuit will essentially remove this difficulty.

The computational efficiency of WR can be further improved by the following techniques.

1) The use of an adaptive error control scheme. In this scheme, the errors incurred in computing the solutions of the decomposed system during the initial iterations are allowed to be large and are progressively decreased as the sequence of iterated solutions converges to the final solution. Various approximation techniques can be used to compute the solution of the decomposed system during the initial iterations of the method. These techniques are discussed and a proof of convergence of WR with this adaptive error control scheme is given in [16].

2) The implementation of WR on pipeline or parallel processors. Both types of relaxation, i.e., GJ and GS relaxation, can be implemented in nonstandard computer architectures. However, parallel processors are better suited to GJ-WR since the analysis of all decomposed subsystems for any given iteration can be done concurrently.

Finally, we remark that the application of the WR method is not limited to only MOS digital circuits. The method can be applied equally well to digital circuits implemented by other types of devices such as bipolar transistors.

Appendix

Example A1. The purpose of this example is to illustrate some behaviors of a WR algorithm that uses an inconsistent assignment-partitioning process. The dynamical equations to be analyzed are

$$\dot{y}_1 + y_2 - u = 0, \quad y_1(0) = 0 \tag{A1.1a}$$

$$y_1 - y_2 = 0. \tag{A1.1b}$$

Let y_1 and y_2 be assigned to (A1.1b) and (A1.1a), respectively, and let (A1.1) be partitioned into two subsystems consisting of {(A1.1a)} and {(A1.1b)}. Applying the WR Algorithm Model 2.1, the kth iteration of the resulting GS-WR algorithm is given by

$$y_2^k = -\dot{y}_1^{k-1} + u \tag{A1.2a}$$

$$y_1^k = y_2^k. \tag{A1.2b}$$

Notice that, while the original system (A1.1) has one state variable, the decomposed system (A1.2) at the kth iteration has no state variable since it consists of only algebraic equations. Hence the above assignment-partitioning process is not consistent with (A.1). It is easy to show that the solutions of (A1.2) are

$$y_1^k(t) = y_2^k(t) = \sum_{j=0}^{k-1} (-1)^j \frac{d^j}{dt^j} u(t) + (-1)^k \frac{d^k}{dt^k} y_1^0(t).$$

Thus both $y_1^k(t)$ and $y_2^k(t)$ diverge when the initial guess is $y_1^0(t) = e^{-at}$, $a > 1$. Now suppose that the initial guess is of the form $y_1^0(t) = \Sigma_{j=0}^{l} a_j t^j$ where l is a finite positive integer and a_j, $j = 1, 2, \cdots, l$ are constants. Then we can deduce the following results.

a) When the input $u(t) = \alpha$; $t \geqslant 0$, the solutions of (A1.1) are $y_1(t) = y_2(t) = \alpha(1 - e^{-t})$ but both $y_1^k(t)$ and $y_2^k(t)$ converge to $\hat{y}_1(t) = \hat{y}_2(t) = \alpha$ which are solutions of (A1.1) with a different initial condition, i.e., $\hat{y}_1(0) = \alpha$.

b) When $u(t) = e^{-at}$, $a > 1$, both $y_1^k(t)$ and $y_2^k(t)$ diverge while the solutions of (A1.1) are $y_1(t) = y_2(t) = (1/1 - a)(e^{-at} - e^{-t})$.

c) If $u(\cdot)$ is piecewise continuous with at least one discontinuity point, then both $y_1^k(\cdot)$ and $y_2^k(\cdot)$ will be unbounded at those discontinuity points but the solutions of (A1.1) will be continuous and bounded within the given time interval.

This example shows that for an inconsistent WR algorithm, the iteration may converge to solutions of the original dynamical system with a different set of initial conditions. Furthermore, the convergence to the correct solutions arises only under special conditions on the input waveforms, initial guesses and initial conditions. Hence inconsistent WR algorithms tend to have poor behaviors and should not be used. ∎

Lemma A2. Let $\|\cdot\|_a$, $\|\cdot\|_b$, be norms in $\boldsymbol{R}^n$, $\boldsymbol{R}^{n+l}$, respectively. Then there exists a constant μ such that

$$\|x\|_a \leqslant \mu \left\| \begin{matrix} x \\ z \end{matrix} \right\|_b, \quad \text{for all } x \in \boldsymbol{R}^n \text{ and } z \in \boldsymbol{R}^l.$$

Proof. We use the fact that all norms in a finite-dimensional space are equivalent (see [15, p. 39]). That is, if $\|\cdot\|_a$ and $\|\cdot\|_{\tilde{a}}$ are norms in $\boldsymbol{R}^n$, there exist constants μ_1 and μ_2 such that for any $x \in \boldsymbol{R}^n$

$$\|x\|_a \leqslant \mu_1 \|x\|_{\tilde{a}} \quad \text{and} \quad \|x\|_{\tilde{a}} \leqslant \mu_2 \|x\|_a. \tag{A2.1}$$

Define

$$\|x\|_{\tilde{a}} \triangleq \left\| \begin{matrix} x \\ 0 \end{matrix} \right\|_b \quad \text{and} \quad \left\| \begin{matrix} x \\ z \end{matrix} \right\|_{\tilde{b}} \triangleq \left\| \begin{matrix} x \\ 0 \end{matrix} \right\|_b + \left\| \begin{matrix} 0 \\ z \end{matrix} \right\|_b. \tag{A2.2}$$

Then from (A2.1), there exist $\mu_1, \tilde{\mu}_2$ such that

$$\|x\|_a \leqslant \mu_1 \|x\|_{\tilde{a}} \quad \text{and} \quad \left\| \begin{matrix} x \\ z \end{matrix} \right\|_{\tilde{b}} \leqslant \tilde{\mu}_2 \left\| \begin{matrix} x \\ z \end{matrix} \right\|_b. \tag{A2.3}$$

Equations (A2.2) and (A2.3) imply that

$$\|x\|_a \leqslant \mu_1 \|x\|_{\tilde{a}} \leqslant \mu_1 \left\| \begin{matrix} x \\ z \end{matrix} \right\|_{\tilde{b}} \leqslant \mu_1 \tilde{\mu}_2 \left\| \begin{matrix} x \\ z \end{matrix} \right\|_b.$$

Therefore, the proof is completed. ∎

Proof of Theorem 3.1. Define

$$\boldsymbol{D} \triangleq \{t \in [0, T] \,|\, t \text{ is a discontinuity point of either } u(\cdot), \dot{x}^0(\cdot), \text{ or } z^0(\cdot)\}$$

$$\boldsymbol{X} \times \boldsymbol{Z} \triangleq \{(x(\cdot), z(\cdot)\colon [0, T] \to \boldsymbol{R}^n \times \boldsymbol{R}^l \,|\, \dot{x}(\cdot) \text{ and } z(\cdot) \text{ are piecewise continuous with possible discontinuity points in } \boldsymbol{D}\}$$

$\boldsymbol{F}\colon \boldsymbol{X} \times \boldsymbol{Z} \to \boldsymbol{X} \times \boldsymbol{Z}$ such that $\begin{pmatrix} \tilde{x}(\cdot) \\ \tilde{z}(\cdot) \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x(\cdot) \\ z(\cdot) \end{pmatrix}$ satisfies

$$\dot{\tilde{x}} = f(\tilde{x}, x, \dot{x}, z, u), \qquad \tilde{x}(0) = x(0) \tag{A3.1a}$$

$$\tilde{z} = g(\tilde{x}, x, \dot{x}, z, u). \tag{A3.1b}$$

Since f is Lipschitz continuous with respect to $\tilde{x}$, by the Picard-Lindelof theorem on the existence and uniqueness of the solutions of ordinary differential equations (see [17, p. 18]), the above equations have a unique solution $(\tilde{x}(\cdot), \tilde{z}(\cdot))$ which belongs to $\boldsymbol{X} \times \boldsymbol{Z}$. Therefore, $\boldsymbol{F}$ is well defined. From these definitions the canonical WR algorithm described by (3.3) can be written as

$$\begin{pmatrix} x^k(\cdot) \\ z^k(\cdot) \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x^{k-1}(\cdot) \\ z^{k-1}(\cdot) \end{pmatrix}. \tag{A3.2}$$

Let $\hat{\gamma} = \max(\gamma, 0.1)$ and α be a positive constant whose value will be chosen later. Define norms in $\boldsymbol{R}^n \times \boldsymbol{R}^l \times \boldsymbol{R}^n$ and $\boldsymbol{X} \times \boldsymbol{Z}$ as follows:

$$\left\| \begin{matrix} \dot{x}(t) \\ z(t) \\ x(t) \end{matrix} \right\| \triangleq \left\| \begin{matrix} \dot{x}(t) \\ z(t) \end{matrix} \right\| + \frac{\lambda_2}{\hat{\gamma}} \|x(t)\| \tag{A3.3}$$

$$\left\| \begin{matrix} x(\cdot) \\ z(\cdot) \end{matrix} \right\| \triangleq \max_{t \in [0, T]} e^{-\alpha t} \left\| \begin{matrix} \dot{x}(t) \\ z(t) \\ x(t) \end{matrix} \right\| \tag{A3.4}$$

where γ, λ_2 and the norms in $\boldsymbol{R}^n \times \boldsymbol{R}^l$ and $\boldsymbol{R}^n$ are as given in Theorem 3.1. Since

$$\left\| \begin{matrix} x(\cdot) \\ z(\cdot) \end{matrix} \right\| \geqslant e^{-\alpha T} \max_{t \in [0, T]} \left\| \begin{matrix} \dot{x}(t) \\ z(t) \\ x(t) \end{matrix} \right\|.$$

Therefore, it can be shown that the space $(\boldsymbol{X} \times \boldsymbol{Z}, \|\cdot\|)$ is closed (hence it is a Banach space). Next we shall show that $\boldsymbol{F}$ is contractive in $(\boldsymbol{X} \times \boldsymbol{Z}, \|\cdot\|)$. Let

$$\begin{pmatrix} \tilde{x}^1(\cdot) \\ \tilde{z}^1(\cdot) \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x^1(\cdot) \\ z^1(\cdot) \end{pmatrix}$$

and

$$\begin{pmatrix} \tilde{x}^2(\cdot) \\ \tilde{z}^2(\cdot) \end{pmatrix} = \boldsymbol{F} \begin{pmatrix} x^2(\cdot) \\ z^2(\cdot) \end{pmatrix}$$

then from (A3.1) and the condition (b) of Theorem 3.1, we have that

$$\begin{aligned} &\left\| \begin{matrix} \dot{\tilde{x}}^1(t) - \dot{\tilde{x}}^2(t) \\ \tilde{z}^1(t) - \tilde{z}^2(t) \end{matrix} \right\| \\ &= \left\| \begin{matrix} f(\tilde{x}^1, x^1, \dot{x}^1, z^1, u) - f(\tilde{x}^2, x^2, \dot{x}^2, z^2, u) \\ g(\tilde{x}^1, x^1, \dot{x}^1, z^1, u) - g(\tilde{x}^2, x^2, \dot{x}^2, z^2, u) \end{matrix} \right\| \\ &\leqslant \lambda_1 \|\tilde{x}^1(t) - \tilde{x}^2(t)\| + \lambda_2 \|x^1(t) - x^2(t)\| \\ &\quad + \gamma \left\| \begin{matrix} \dot{x}^1(t) - \dot{x}^2(t) \\ z^1(t) - z^2(t) \end{matrix} \right\|. \end{aligned}$$

After some algebraic manipulations using (A3.3), $\hat{\gamma}$ and the above inequality, we have that

$$\left\|\begin{matrix}\dot{\tilde{x}}^1(t)-\dot{\tilde{x}}^2(t)\\ \tilde{z}^1(t)-\tilde{z}^2(t)\\ \tilde{x}^1(t)-\tilde{x}^2(t)\end{matrix}\right\| \leqslant \hat{\gamma}\left\|\begin{matrix}\dot{x}^1(t)-\dot{x}^2(t)\\ z^1(t)-z^2(t)\\ x^1(t)-x^2(t)\end{matrix}\right\| + \left(\lambda_1+\frac{\lambda_2}{\hat{\gamma}}\right)\|\tilde{x}^1(t)-\tilde{x}^2(t)\|. \quad \text{(A3.5)}$$

From Lemma A2, there exists a constant μ such that

$$\|\dot{\tilde{x}}^1(t)-\dot{\tilde{x}}^2(t)\| \leqslant \mu\left\|\begin{matrix}\dot{\tilde{x}}^1(t)-\dot{\tilde{x}}^2(t)\\ \tilde{z}^1(t)-\tilde{z}^2(t)\\ \tilde{x}^1(t)-\tilde{x}^2(t)\end{matrix}\right\|. \quad \text{(A3.6)}$$

From (A3.5) and (A3.6), letting $\mu_1=\mu\hat{\gamma}$ and $\mu_2=\mu(\lambda_1+\lambda_2/\hat{\gamma})$, we have

$$\|\dot{\tilde{x}}^1(t)-\dot{\tilde{x}}^2(t)\| \leqslant \mu_1\left\|\begin{matrix}\dot{x}^1(t)-\dot{x}^2(t)\\ z^1(t)-z^2(t)\\ x^1(t)-x^2(t)\end{matrix}\right\| + \mu_2\|\tilde{x}^1(t)-\tilde{x}^2(t)\|. \quad \text{(A3.7)}$$

Applying the fundamental results in differential inequalities to (A3.7) (see [17, corollary 6.2, pp. 30-32]), using the fact that $\tilde{x}^1(0)-\tilde{x}^2(0)=0$, we have

$$\|\tilde{x}^1(t)-\tilde{x}^2(t)\| \leqslant \mu_1 e^{\mu_2 t}\int_0^t e^{-\mu_2\tau}\left\|\begin{matrix}\dot{x}^1(\tau)-\dot{x}^2(\tau)\\ z^1(\tau)-z^2(\tau)\\ x^1(\tau)-x^2(\tau)\end{matrix}\right\| d\tau$$

$$= \mu_1 e^{\mu_2 t}\int_0^t e^{(\alpha-\mu_2)\tau}e^{-\alpha\tau}\left\|\begin{matrix}\dot{x}^1(\tau)-\dot{x}^2(\tau)\\ z^1(\tau)-z^2(\tau)\\ x^1(\tau)-x^2(\tau)\end{matrix}\right\| d\tau. \quad \text{(A3.8)}$$

From (A3.4) we have

$$e^{-\alpha\tau}\left\|\begin{matrix}\dot{x}^1(\tau)-\dot{x}^2(\tau)\\ z^1(\tau)-z^2(\tau)\\ x^1(\tau)-x^2(\tau)\end{matrix}\right\| \leqslant \left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|, \quad \text{for all } \tau\in[0,T].$$

Substituting this inequality in (A3.8), we have

$$\|\tilde{x}^1(t)-\tilde{x}^2(t)\| \leqslant \mu_1 e^{\mu_2 t}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|\int_0^t e^{(\alpha-\mu_2)\tau}d\tau$$

$$= \frac{\mu_1 e^{\mu_2 t}}{\alpha-\mu_2}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|(e^{(\alpha-\mu_2)t}-1)$$

$$\leqslant \frac{\mu_1 e^{\alpha t}}{\alpha-\mu_2}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|$$

assuming $\alpha-\mu_2>0$.

Substituting this inequality in (A3.5) and multiplying by $e^{-\alpha t}$, we have

$$e^{-\alpha t}\left\|\begin{matrix}\dot{\tilde{x}}^1(t)-\dot{\tilde{x}}^2(t)\\ \tilde{z}^1(t)-\tilde{z}^2(t)\\ \tilde{x}^1(t)-\tilde{x}^2(t)\end{matrix}\right\| \leqslant \hat{\gamma}e^{-\alpha t}\left\|\begin{matrix}\dot{x}^1(t)-\dot{x}^2(t)\\ z^1(t)-z^2(t)\\ x^1(t)-x^2(t)\end{matrix}\right\| + \frac{\mu_1\left(\lambda_1+\frac{\lambda_2}{\hat{\gamma}}\right)}{\alpha-\mu_2}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|.$$

Hence, using (A3.4), the above inequality implies that

$$\left\|\begin{matrix}\tilde{x}^1(\cdot)-\tilde{x}^2(\cdot)\\ \tilde{z}^1(\cdot)-\tilde{z}^2(\cdot)\end{matrix}\right\| \leqslant \hat{\gamma}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\| + \frac{\mu_1\left(\lambda_1+\frac{\lambda_2}{\hat{\gamma}}\right)}{\alpha-\mu_2}\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|.$$

That is

$$\left\|\boldsymbol{F}\begin{pmatrix}x^1(\cdot)\\ z^1(\cdot)\end{pmatrix}-\boldsymbol{F}\begin{pmatrix}x^2(\cdot)\\ z^2(\cdot)\end{pmatrix}\right\| \leqslant \left[\hat{\gamma}+\frac{\mu_1\left(\lambda_1+\frac{\lambda_2}{\hat{\gamma}}\right)}{\alpha-\mu_2}\right]\cdot\left\|\begin{matrix}x^1(\cdot)-x^2(\cdot)\\ z^1(\cdot)-z^2(\cdot)\end{matrix}\right\|.$$

Since $0.1\leqslant\hat{\gamma}<1$, it is possible to choose α large enough so that

$$\hat{\gamma}+\frac{\mu_1\left(\lambda_1+\frac{\lambda_2}{\hat{\gamma}}\right)}{\alpha-\mu_2}<1.$$

Therefore, $\boldsymbol{F}$ is contractive. Hence, by the contraction mapping theorem (see [15, p. 120]), it has a unique fixed point in $\boldsymbol{X}\times\boldsymbol{Z}$ satisfying

$$\begin{pmatrix}\hat{x}(\cdot)\\ \hat{z}(\cdot)\end{pmatrix}=\boldsymbol{F}\begin{pmatrix}\hat{x}(\cdot)\\ \hat{z}(\cdot)\end{pmatrix}$$

i.e.,

$$\dot{\hat{x}}=f(\hat{x},\hat{x},\dot{\hat{x}},\hat{z},u), \qquad \hat{x}(0)=x(0)$$
$$\hat{z}=g(\hat{x},\hat{x},\dot{\hat{x}},\hat{z},u)$$

and for any given initial guess $(x^0(\cdot),z^0(\cdot))\in\boldsymbol{X}\times\boldsymbol{Z}$ the sequence $\{(x^k(\cdot),z^k(\cdot))\}_{k=1}^{\infty}$ generated by (A3.2) converges uniformly to $(\hat{x}(\cdot),\hat{z}(\cdot))\in\boldsymbol{X}\times\boldsymbol{Z}$. Since we can choose $(x^0(\cdot),z^0(\cdot))\in\boldsymbol{X}\times\boldsymbol{Z}$ such that $x^0(t)=x(0)$ and $z^0(t)=0$ for all $t\in[0,T]$, therefore, we conclude that the discontinuity points of $(\hat{x}(\cdot),\hat{z}(\cdot))$ belong to the set of discontinuity points of $u(\cdot)$ only, i.e., they do not depend on $\dot{x}^0(\cdot)$ and $z^0(\cdot)$. Hence the proof is complete. ■

Proof of Theorem 4.1. For the sake of simplicity, we shall prove this theorem for the case in which all capacitors are linear. The proof for the general case of nonlinear capacitors is given in [19]. Let

$$C=L+U$$

where $C \in \boldsymbol{R}^{n\times n}$ is the capacitance matrix, $L \in \boldsymbol{R}^{n\times n}$ is a lower triangular matrix, and $U \in \boldsymbol{R}^{n\times n}$ is a strictly upper triangular matrix, i.e., the diagonal elements of U are zero.

Due to Assumption (ii) that there is a nonzero grounded capacitor at every node, C is strictly diagonally dominant. Therefore, it can be shown (e.g., see [13]) that

$$\|L^{-1}U\|_\infty < 1. \tag{A4.1}$$

For Algorithm 4.1, we define f: $\boldsymbol{R}^n \times \boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^n$ such that

$$f_i(v^k, v^{k-1}, u) = q_i(v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)$$

for $i = 1, 2, \cdots, n$. From the first assumption of Theorem 4.1, we conclude that f is Lipschitz continuous with respect to v^k and v^{k-1} and is continuous with respect to u. Then the kth iteration of Algorithm 4.1 can be described as solving for v^k from

$$\dot{v}^k + (L^{-1}U)\,\dot{v}^{k-1} + L^{-1}f(v^k, v^{k-1}, u) = 0. \tag{A4.2}$$

Equation (A4.1) and the Lipschitz property of f imply that the canonical form of Algorithm 4.1 given by (A4.2) satisfies the conditions of Theorem 3.1. Therefore, Algorithm 4.1 is convergent.

For Algorithm 4.2, we define $\tilde{f}$: $\boldsymbol{R}^l \times \boldsymbol{R}^n \times \boldsymbol{R}^n \times \boldsymbol{R}^r \to \boldsymbol{R}^n$ such that

$$\tilde{f}_i(z^{k-1}, v^k, v^{k-1}, u) = \tilde{q}_i(z^{k-1}, v_1^k, \cdots, v_i^k, v_{i+1}^{k-1}, \cdots, v_n^{k-1}, u)$$

for $i = 1, 2, \cdots, n$. From the first assumption of Theorem 4.1, we conclude that $\tilde{f}$ is Lipschitz continuous with respect to z^{k-1}, v^k, and v^{k-1} and is continuous with respect to u. Then the kth iteration of Algorithm 4.2 can be described as solving for v^k and z^k from

$$\dot{v}^k + (L^{-1}U)\,\dot{v}^{k-1} + L^{-1}\tilde{f}(z^{k-1}, v^k, v^{k-1}, u) = 0 \tag{A4.3}$$

$$z^k = g(v^k, u). \tag{A4.4}$$

Substituting (A4.4) into (A4.3), we obtain

$$\dot{v}^k + (L^{-1}U)\,\dot{v}^{k-1} + L^{-1}\tilde{f}(g(v^{k-1}, u), v^k, v^{k-1}, u) = 0. \tag{A4.5}$$

Equation (A4.1) and the Lipschitz properties of both $\tilde{f}$ and g imply that the canonical form of Algorithm 4.2 given by (A4.5) satisfies the conditions of Theorem 3.1. Therefore Algorithm 4.2 is convergent. ■

Acknowledgment

The authors wish to thank V. Visvanathan, R. K. Brayton, G. H. Hachtel, A. R. Newton, N. B. G. Rabbat, R. J. Kaye, and M. J. Chen for many helpful discussions. Also the authors wish to thank W. Nye and B. S. Landman for suggestions in developing RELAX. The comments and suggestions of the reviewers to improve the presentation of this paper are gratefully acknowledged. The authors also acknowledge the interesting discussion on relaxation methods with W. Kahan and B. Parlett.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Res. Lab. Rep. ERL-M520, Univ. California, Berkeley, May 1975.

[2] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for ASTAP–A network analysis program," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 628–624, Nov. 1973.

[3] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS–An MOS timing simulator," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 901–910, Dec. 1975.

[4] A. R. Newton, "The simulation of large scale integrated circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 741–749, Sept. 1979.

[5] A. R. Newton, "The simulation of large scale integrated circuits," Univ. California, Berkeley, Electronics Res. Lab., Memo. ERL-M78/52, July 1978.

[6] G. Arnout and H. De Man, "The use of threshold functions and Boolean-controlled network elements for macromodelling of LSI circuits," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 326–332, June 1978.

[7] N. B. G. Rabbat, A. L. Sangiovanni-Vincentelli, and H. Y. Hsieh, "A multilevel Newton algorithm with macromodelling and latency for the analysis of large-scale nonlinear circuits in the time domain," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 733–741, Sept. 1979.

[8] G. DeMicheli and A. L. Sangiovanni-Vincentelli, "Numerical properties of algorithms for the timing analysis of MOS VLSI circuits," in *Proc. ECCTD'81*, The Hague, Aug. 1981.

[9] A. Vladimirescu and S. Liu, "The simulation of MOS integrated circuits using SPICE2," Univ. California, Berkeley, Electronic Res. Lab. Memo. UCB/ERL-M80/7, Oct. 1980.

[10] A. E. Ruehli, A. L. Sangiovanni-Vincentelli, and N. B. G. Rabbat, "Time analysis of large scale circuits containing one-way macromodels," in *IEEE Proc. Int. Symp. Circuits and Systems*, 1980.

[11] W. Kahan, private notes, 1975.

[12] C. W. Ho, A. E. Ruehli and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 504–509, June 1975.

[13] R. Varga, *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1962.

[14] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 101–113, Jan. 1971.

[15] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic, 1970.

[16] E. Lelarasmee, "The waveform relaxation method for the time domain analysis of large scale nonlinear dynamical systems," Ph.D. dissertation, Univ. California, Berkeley, 1982.

[17] J. K. Hale, *Ordinary Differential Equations*. New York: McGraw-Hill, 1969.

[18] E. Lelarasmee and A. L. Sangiovanni-Vincentelli, "RELAX: A new circuit simulator for large scale MOS integrated circuits," Univ. California, Berkeley, Electronic Res. Lab., Memo. UCB/ERL-M82/6, Feb. 1982.

[19] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time domain analysis of large scale integrated circuit," Univ. California, Berkeley, Electronic Res. Lab., Memo. UCB/ERL-M81/75, June 1981.

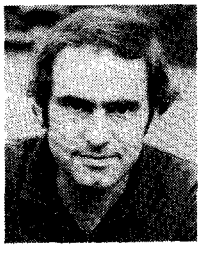
*

Ekachai Lelarasmee (S'78) was born in Bangkok, Thailand, on July 10, 1953. He received the bachelor degree in electrical engineering from Chulalongkorn University in 1974 and the M.S.E.E. degree from the University of California, Berkeley, in 1978.

Since 1974 he has been with the Department of Electrical Engineering, Chulalongkorn University. In 1976, he received a scholarship from the Anandamahidol Foundation to continue his graduate study at University of California,

Berkeley where he is currently a Doctoral student of the Department of Electrical Engineering and Computer Science and a Research Assistant of the Electronics Research Laboratory. He carried out research in computer-aided design at the IBM T. J. Watson Research Center, Yorktown Heights, NY, from April to December 1980 and at Harris Semiconductor, Melbourne, FL., from September to December 1981. In 1982, his research on the Waveform Relaxation method received the D. J. Sakrison Memorial award for outstanding research conducted within the EECS Department in 1981. His current interests are in the areas of techniques for computer-aided analysis and optimization.

*

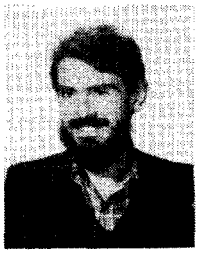


Albert E. Ruehli (M'65-SM'74) was born in Zurich, Switzerland, on June 22, 1937. He received the telecommunication engineering degree from Abend Technicum, Zurich, Switzerland in 1963, and the Ph.D. degree in electrical engineering from the University of Vermont, Burlington, in 1972.

From 1958 to 1963, he worked at the IBM Research Laboratory in Zurich on thin magnetic film memories. In 1963, he transferred to the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, where he worked in the area of semiconductor circuits and devices. In 1966, he moved to IBM Burlington, where he was associated with the Mathematical and Engineering Analysis Group at the Components Division Laboratory. Since 1971, he has been a Research Staff Member in the Circuit and Computer Design Automation Group at the Thomas J. Watson Research Center, developing new techniques for the electrical analysis of computer-hardware technologies. He has published several papers on methods of computer-aided inductance and capacitance analysis as well as equivalent-circuit description of hardware.

In 1975, Dr. Ruehli received the IBM Outstanding Contribution Award for work in the interactive circuit design area.

*



Alberto Sangiovanni-Vincentelli (M'74-SM'81) was born in Milano, Italy, in June 1947. He received the Dr. Eng. degree (Summa cum Laude) from the Politecnico di Milano, Italy, in 1971.

From 1971 to 1977, he was with the Instituto di Elettrotecnica ed Elettronica del Politecnico di Milano, Italy, where he held the position of Research Associate, Assistant and Associate Professor. In 1975 he was Research Associate at the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. He joined the department in 1977 after spending a year as Visiting Assistant Professor. Presently he is Associate Professor and Vice Chairman of the same department. In 1980 he was a Visiting Scientist at the Mathematical Sciences Department of IBM T. J. Watson Research Center. In the Fall 1981, he was with Harris Semiconductor Analog Product Division. He is a consultant in the area of computer-aided design to several industries. His research interests are in computer-aided design of electronic circuits and systems, with particular emphasis on VLSI circuits and optimization, combinatorial optimization and nonlinear systems.

Dr. Sangiovanni-Vincentelli was Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, and is Associate Editor of the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN of Integrated Circuits and a member of the Large Scale Systems Committee of the IEEE Circuits and Systems Society. He was the Guest Editor of a special issue of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS on CAD for VLSI. In 1981 he received the Distinguished Teaching Award of the University of California.