VLSI '87, C.H. Séquin (editor)
Elsevier Science Publishers B.V. (North-Holland)
© IFIP, 1988

Simulating MOS VLSI Circuits Using SuperCrystal

Romy L. Bauer
Antony P-C Ng
Arvind Raghunathan
Mark W. Saake

Computer Science Division
573 Evans Hall
University of California
Berkeley, CA 94720.

Clark D. Thompson

Department of Computer Science
University of Minnesota, Duluth
Duluth, MN 55812

ABSTRACT

This paper describes SuperCrystal, a computer program for quick yet reasonably accurate simulation of MOS VLSI circuits.

SuperCrystal divides the time axis into time steps. In each time step, a transistor circuit maps to a linear RC network for the purpose of waveform estimation. Single exponential approximations to node waveforms in the RC network are then calculated using an algorithm due to Raghunathan and Thompson.

Unlike several other timing analyzers based on the linear RC model, SuperCrystal approximates transistor resistances as functions of the voltage waveforms at the gate, source, and drain nodes, after distinguishing between the triode and pinchoff modes of operation. An iterative algorithm is presented for the calculation of a transistor's average resistance over a given time period.

Preliminary tests using SuperCrystal are highly encouraging. Several MOS circuits are analyzed in this paper using Crystal, SuperCrystal and SPICE, and the resulting outputs and CPU times are compared.

1. Introduction

Giant leaps in technology over the last several years have made possible the routine production and use of MOS VLSI chips with hundreds of thousands of transistors. SPICE [7], a popular circuit simulator, provides detailed and highly accurate voltage waveforms. Unfortunately, SPICE is much too slow to analyze VLSI chips directly. It is useful only on isolated subcircuits of tens or hundreds of transistors.

This work was supported in part by National Science Foundation through its Computer Engineering Program, under grant DMC-8406408. Author Ng is also supported by a Regent's Fellowship; author Saake is also supported by the Golden Nugget Scholarship Fund.

Relaxation-based simulators, such as SPLICE [16], ELOGIC [5] and WASIM [8], and waveform-relaxation based simulators, such as RELAX2.1 [22], offer potentially faster runtimes while retaining the accuracy of SPICE.

At the other end of the speed-accuracy spectrum are timing analyzers such as Crystal [12], MOSSIM [2], SDS [6], and TV [4]. These analyzers are fast enough to process entire VLSI chips, but are sometimes woefully inaccurate.

SuperCrystal is a new circuit simulator suitable for full-custom VLSI. Our intention was to build a simulator that is nearly as fast as Crystal, but much more accurate. SuperCrystal is implemented in the C programming language, and runs under Berkeley UNIX.

Conventional circuit simulators attempt to find the exact voltage waveforms at nodes in a MOS circuit. Since these circuits are in general non-linear, waveform estimation is time consuming [9]. In an attempt to get around this problem, SuperCrystal restricts circuit elements to be MOS transistors, linear resistors, and linear grounded capacitors. Voltage waveforms of nodes in the circuit are approximated as first-order exponentials.

SuperCrystal is most closely akin to the waveform estimators which model a digital MOS circuit by a linear RC network. This approach, described initially by Bryant [2], has been used in a number of timing analyzers [4,6,12,15,19]. Raghunathan and Thompson [13,14] extend Bryant's model by incorporating leakage resistors in RC trees. SuperCrystal currently uses the Raghunathan Thompson algorithm for waveform estimation.

This paper is organized as follows: In Section 2, we review classical circuit simulation techniques. Section 3 gives an outline and organization of SuperCrystal. The Raghunathan Thompson algorithm is detailed in section 4. Section 5 shows how the transistor model is incorporated into the Raghunathan Thompson algorithm. Section 6 describes how SuperCrystal schedules the simulation of different parts of the circuit. Discretization of the simulation into time steps is discussed in section 7. Section 8 details the comparative results of running Crystal, SuperCrystal, and SPICE on various circuit examples. Our conclusions and directions for future work are presented in section 9.

2. A Brief Overview of Circuit Simulation

For the purpose of waveform estimation, a MOS circuit maps to an LCR network. Such an LCR network is modeled by a system of non-linear, first-order differential equations which is in general impossible to solve exactly. This motivates the discretization of the time axis into *time points*. An approximate solution to the system of equations is obtained for each time point. Information from previous time points is used to predict the solution for the current time point.

2.1. Standard Circuit Simulation

Standard circuit simulators like SPICE [7] apply stiffly stable integration formulas (e.g. Backward Euler) at each time point to the non-linear system of nodal equations to yield a set of non-linear algebraic difference equations. These equations are solved iteratively using a damped Newton-Raphson algorithm. Each iteration produces a sparse linear system, which is solved by sparse LU decomposition or Gaussian Elimination. Experimental evidence indicate runtimes of $O(n^{1.1-1.5})$ to solve the sparse linear system.

Newton-Raphson approximation techniques are preferred in practice because the rate of convergence is quadratic and convergence is guaranteed if the initial guess is sufficiently close to the solution.

2.2. Relaxation-Based Methods

Relaxation-based methods do not require the direct solution of a large non-linear system of equations. They also permit the simulator to only solve for the nodes whose waveforms are actually changing. The two most common techniques used to solve the system of nodal equations are the Gauss-Jacobi and the Gauss-Seidel methods [11, 20].

Both the Gauss-Jacobi and the Gauss-Seidel methods are iterative methods. If the circuit is restricted to have a grounded capacitor at each node, then convergence of both methods is guaranteed, and the rate of convergence is at least linear. This compares unfavorably with the provable quadratic convergence rate of the Newton-Raphson algorithm.

One advantage of relaxation-based methods is that they involve solving a set of decoupled equations, while Newton-Raphson methods involve solving a set of simultaneous equations. Therefore the computational cost of each iteration of relaxation-based methods is $O(n)$. This compares well with the sparse LU decomposition runtimes if the number of iterations is small.

Examples of relaxation-based circuit simulators are SPLICE [16], and WASIM [8].

3. The SuperCrystal Approach

SuperCrystal restricts circuit elements to be MOS transistors, linear resistors, and linear grounded capacitors. SuperCrystal models such a circuit as a *control graph* where each control graph node is a *transistor group*.

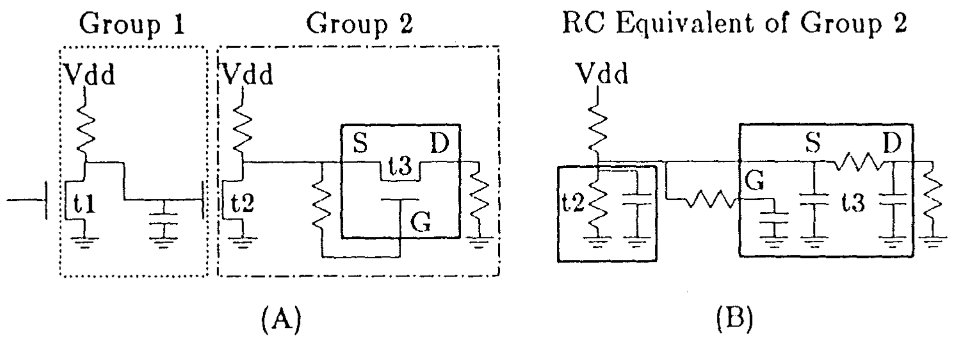

Figure 1: Transistor Groups

Definition 1: A *transistor group* is a collection of circuit elements that are electrically connected by wires or by transistor channels. Notice that under this definition, the gate of a transistor is not in the same transistor group as the source and drain unless explicitly connected, as in transistor *t3* in figure 1A.

For the purposes of waveform estimation, a transistor group maps to an RC network. Figure 1B shows the such a mapping. In this mapping, a MOS transistor is modeled by a non-linear resistor. However, the Raghunathan Thompson algorithm expects linear resistors. Our solution is to partition the time axis into *time steps* and determine a linear approximation to the non-linear resistor within each time step.

Definition 2: The *Control Graph* of a circuit is a digraph where the nodes are transistor groups and where a directed edge (u,v) exists between distinct nodes u and v iff the gate node of some transistor t is in transistor group u, and the source and drain nodes of t are in transistor group v.

The decomposition of the circuit into transistor groups is motivated by the fact that the voltages of the nodes in a transistor group are heavily interdependent, and have to be solved as a simultaneous system of equations. On the other hand, transistor groups interact by modulating the effective resistance of transistors. Note that this is "unidirectional"; if the gate of a transistor belongs in a different transistor group from its source and drain, the voltage at the gate of a transistor affects its source-drain resistance, but not vice versa. So it suffices to ensure that the gate waveforms of transistors in a transistor group are valid before simulating it.

The naive approach of simulating the entire circuit can be improved by exploiting *latency*. We say a transistor group v is *latent* with respect to some input transient applied to a node in another transistor group u, iff there **doesn't** exist a directed path from node u to v in the control graph. Intuitively, this means that a transient applied to a node in transitor group u cannot affect transistor group v. SuperCrystal only simulates transistor groups that are not latent with respect to the applied transient.

4. The RC Model

This section introduces the model of Raghunathan and Thompson for their waveform estimation algorithm.

Definition 3: An *RC network* is a tree on n nodes. With each edge i is associated a nonnegative resistance r_i. With each node k is associated a positive resistance R_k, a positive capacitance C_k, a nonnegative charge Q_k, and a voltage source $v_k \in \{V_{DD}, GND, \phi\}$, where ϕ indicates no connection.

In the above definition, circuit elements associated with a node exist between the node and GND in the network. Note that this definition does not allow for floating capacitors (capacitors associated with the edges of the graph) in the network.

The incorporation of leakage resistors in the RC network has a twofold effect:

1 Node capacitors need not have to charge to V_{DD}, or discharge to GND, but could take on intermediate voltage values.

2 RC networks can now be driven by more than one source.

When the edges and nodes are all labeled with numbers, these parameters can be grouped together as vectors. An RC network is then denoted by $N(n, r, R, C, Q, V)$, where n is the number of nodes, r is a vector of edge resistances, R is a vector of leakage resistances, C is a vector of node capacitances, Q is a vector of capacitance charges, and V is a vector of node voltage sources. When an RC network is driven by exactly one source, it is called a standard RC network.

4.1. Modeling Voltage Waveforms in an RC Network

The Raghunathan Thompson algorithm defines the approximate time constant of the voltage waveform at node k in an RC network to be

$$\tau_k = \frac{\int_0^\infty [v_k(\infty) - v_k(t)]\,dt}{v_k(\infty) - v_k(0)}$$

We now have the following exponential waveform with time constant τ_k,

$$\bar{v}_k(t) = v_k(\infty) + [v_k(0) - v_k(\infty)]\, e^{-t/\tau_k}$$

where $v_k(t)$ represents the actual voltage waveform at node k.

Theorem 1. The exponential waveform $\bar{v}_k(t)$ of node k in an RC network satisfies the property

$$\int_0^\infty [v_k(t) - \bar{v}_k(t)]\, dt = 0$$

In other words, the average error over time is 0.

In an RC network with no input or only step inputs, the Laplace transform [3] of the voltage waveform of a node can be written as

$$V_k(s) = \frac{1}{s}\left[v_k(0) + \frac{c_{1,k}}{1+t_{1,k}s} + \cdots + \frac{c_{n,k}}{1+t_{n,k}s}\right]$$

Stated more simply, the voltage waveform can be expressed as a sum of exponentials. This leads to one further property of τ_k, namely that it is a weighted average of the individual time constants.

Theorem 2. τ_k can be expressed as

$$\tau_k = \frac{c_{1,k}t_{1,k} + \cdots + c_{n,k}t_{n,k}}{c_{1,k} + \cdots + c_{n,k}}$$

Although τ_k has many interesting properties, it is not a linear function in the sense that the time constant of the sum of two voltage waveforms is not the sum of the individual time constants. This leads us to define the following parameter.

Definition 4: The parameter D_k is defined as the product of $(v_k(\infty) - v_k(0))$ and τ_k.

$$D_k = [v_k(\infty) - v_k(0)]\cdot\tau_k$$

It can easily be shown that D_k is a linear function.

When the context is not obvious, we will refer to D_k in a network $N(n, r, R, C, Q, V)$ as $D_k(n, r, R, C, Q, V)$.

We need another important result from network theory, the **Superposition Theorem** [3], which can be stated as follows for the purpose of this paper.

Theorem 3. $D_k(n, r, R, C, Q, V)$ is obtained as the sum of the D_k in each network obtained from $N(n, r, R, C, Q, V)$ by setting all but one source to GND.

$$\begin{aligned} D_k(n, r, R, C, Q, V) = \; & D_k(n, r, R, C, Q, [v_1]) \\ & + D_k(n, r, R, C, Q, [v_2]) \\ & + \cdots \\ & + D_k(n, r, R, C, Q, [v_n]) \end{aligned} \tag{5}$$

where $[v_i]$ represents the vector V with all driven sources other than v_i set to GND.

Since the Raghunathan Thompson algorithm provides $v_k(0)$, $v_k(\infty)$, and D_k for RC networks driven by a single source, we can use the superposition theorem for estimating waveforms in RC networks driven by several sources.

4.2. Waveform Estimation in Standard RC Networks

Prior to any further discussion, a more constructive definition of standard RC networks is given. A standard RC network is recursively defined to be one of the following:

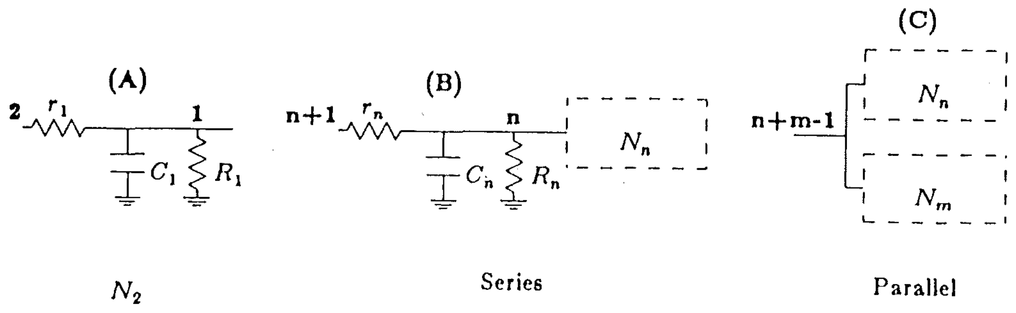

Figure 2: Primitive Element, Series Connection, and Parallel Connection

1 A resistor in series with a nonideal capacitor. The free end of the resistor is the input, labeled 2, and its other end is the output, labeled 1. The other end of the capacitor is grounded. This is shown in figure 2(A). This network will also be referred to as the primitive element or N_2 in the rest of the paper.

2 A series connection of the primitive element and an RC network with n nodes, N_n, to give N_{n+1}. The input of N_{n+1} is the input of N_2, with the input of N_n connected to the output of N_2. The nodes in N_{n+1} are renumbered as follows: The node numbers in N_n remain unchanged. Node 2 of N_2 is relabeled $n+1$. This is shown in figure 2(B).

3 A parallel composition of an n-node network N_n with an m-node network N_m, forming a network N_{n+m-1}. The input of N_{n+m-1} is the input of N_n, as shown in figure 2(C). The nodes in N_{n+m-1} are renumbered as follows: The node numbers in one of them, say N_n, remain the same except for the input node, while the node numbers in N_m get incremented by $n-1$. The input node of N_{n+m-1} gets the label $n+m-1$.

In all three cases, the input node is connected to V_{DD}.

Any N_n has a set of twelve parameters associated with it. While it is beyond the scope of this paper to define them all in detail, some of the more important parameters are defined below.

Parameter	**Dimension**	**Remark**
$\rho^{(n)}$	resistance	Effective Resistance between input and GND.
$v_k^{(n)}(\infty)$	voltage	Final voltage of node k.
$D_k^{(n)}$	time × voltage	Time-Voltage product at node k.
$Q^{(n)}$	charge	Total initial charge in the network.
$\overline{C}^{(n)}$	charge	Total final charge in the network.

From definition 4, the time constant of the voltage waveform at node k is given by

$$\tau_k^{(n)} = \frac{D_k^{(n)}}{v_k^{(n)}(\infty) - v_k^{(n)}(0)}$$

The Raghunathan Thompson algorithm provides simple equations involving only additions and multiplications for the three separate cases: (A) calculating the parameters for N_2, (B) deriving the parameters for N_{n+1} from the parameters for N_2 and N_n, and (C) deriving the parameters for

N_{n+m-1} from the parameters for N_n and N_m. Some of these equations are listed below:

Parameter	Primitive $k=2$	Series $k=n+1$	Parallel $k=n+m-1$
$\rho^{(k)}$	r_1+R_1	$r_n+\frac{R_n\,\rho^{(n)}}{R_n+\rho^{(n)}}$	$\frac{\rho^{(n)}\cdot\rho^{(m)}}{\rho^{(n)}+\rho^{(m)}}$
$v_i^{(k)}(\infty)$	$V_{DD}\left[\frac{R_1}{r_1+R_1}\right]$	$V_{DD}\left[1-\frac{r_i}{\rho^{(i+1)}}\right]$ $i=n$ $\left[\frac{v_n^{(n+1)}(\infty)}{V_{DD}}\right]v_i^{(n)}(\infty)$ otherwise	$v_i^{(n)}(\infty)$ $1\le i<n$ $v_{i-n+1}^{(m)}(\infty)$ $n\le i<n+m-1$
$Q^{(k)}$	$v_1(0)\cdot C_1$	$v_n(0)\cdot C_n+Q^{(n)}$	$Q^{(n)}+Q^{(m)}$
$\overline{C}^{(k)}$	$v_1^{(2)}(\infty)\cdot C_1$	$v_n^{(n+1)}(\infty)\left[C_n+\frac{\overline{C}^{(n)}}{V_{DD}}\right]$	$\overline{C}^{(n)}+\overline{C}^{(m)}$

Table 1: Some Equations from the Raghunathan Thompson Algorithm

The set of equations given in [13] provides us with the following simple linear time algorithm for estimating the waveform at a node in the RC network:

> Starting off from the leaf nodes of the tree, build the tree using the serial and parallel connections described earlier, until the entire tree is built. At each stage of tree construction, update the relevant parameters using the equations of [13].

The node waveform can now be approximated as

$$\bar{v}_i(t) = v_i(\infty) + [v_i(0) - v_i(\infty)]\,e^{-t/\tau_i}$$

5. Transistor Model

SuperCrystal supports any transistor model that satisfies the following requirements:

1 The transistor model must define a finite set of states. A transistor is required to be in exactly one of these states at a given time. The *final* state is the state that a transistor assumes at $t=\infty$. A transistor has *triggered* when it enters its final state permanently. The transistor model must be able to determine when a transistor triggers.

2 The transistor model must provide the effective resistance as a function of the voltages at its gate, source, and drain nodes and other technology dependent process parameters.

Although there are several models for the MOS transistor in the literature, SuperCrystal currently supports the second-order Shichman-Hodges model [18,21]. The Shichman-Hodges model defines three states: OFF, TRIODE, and PINCHOFF.

Let $\mathbf{r}(t)$ denote the effective source-drain resistance of transistor k. Using the Shichman-Hodges model, we have:

$$\mathbf{r}(t) = \begin{cases} \infty & v_{gs}(t) < v_t(t) & & \text{OFF} \\ \dfrac{1}{K(v_{gs}(t)-v_t(t))} & v_{gs}(t) \ge v_t(t) & v_{gd}(t) \ge v_t(t) & \text{TRIODE} \\ \dfrac{2\,v_{ds}(t)}{K(v_{gs}(t)-v_t(t))^2} & v_{gs}(t) \ge v_t(t) & v_{gd}(t) < v_t(t) & \text{PINCHOFF} \end{cases} \qquad (1)$$

where K is a constant dependent on the geometry of the transistor and other technology dependent process parameters, and $v_t(t)$ is the transistor threshold voltage, given by:

$$v_t(t) = \begin{cases} v_{to} + \gamma[(v_{sb}(t) + 2_{\phi f})^{1/2} + (2_{\phi f})^{1/2}] & \text{N-Channel} \\ v_{to} - \gamma[(v_{sb}(t) + 2_{\phi f})^{1/2} + (2_{\phi f})^{1/2}] & \text{P-Channel} \end{cases}$$

where v_{to}, γ, and $2_{\phi f}$ are process parameters.

5.1. The Iterative RC Network Algorithm

There are two difficulties to be overcome in modeling a transistor as a linear resistor. Firstly, the equation for $\mathbf{r}(t)$ is dependent on the voltages at the source and drain nodes, $v_s(t)$ and $v_d(t)$, which are in turn dependent on $\mathbf{r}(t)$. This interdependence motivates the iterative RC network algorithm, described below. Secondly, $\mathbf{r}(t)$ is non-linear. We overcome the non-linearity of equation (1) by simulating a transistor group over some small time interval $[\alpha,\beta]$. The average effective resistance, $\bar{\mathbf{r}}$ is approximated by an integrating average:

$$\bar{\mathbf{r}} = \frac{\int_{\beta}^{\alpha} \mathbf{r}(t)\, dt}{\beta - \alpha}$$

Notation: $X^{(i)}$ denotes the quantity X during the ith iteration of the iterative RC network algorithm.

The iterative RC network algorithm makes an initial guess, $\bar{\mathbf{r}}_k{}^{(0)}$, for each transistor k using the transistor model and the initial voltages at the nodes of the RC network. The node voltages are then approximated with the Raghunathan Thompson algorithm. The transistor model is then run again on the updated node voltages. This iterative process continues until the effective resistances of the transistors have "stabilized".

Formally, at the ith iteration, we have $\bar{\mathbf{r}}_k^{(i)}$. We then run the RC network algorithm to get $v_{s,k}{}^{(i+1)}(t)$, and $v_{d,k}{}^{(i+1)}(t)$, the source and drain voltages for each transistor k. These voltage waveforms then give a value for $\mathbf{r}_k^{(i+1)}(t)$ which we integrate by the above equation to get a value for $\bar{\mathbf{r}}_k^{(i+1)}$.

Definition 5: δ_{trans} is defined as:

$$\delta_{trans}{}^{(i)} = \frac{\sum_{k \in T} \left[\frac{\bar{\mathbf{r}}_k^{(i+1)} - \bar{\mathbf{r}}_k^{(i)}}{\bar{\mathbf{r}}_k^{(i)}} \right]^2}{|T|}$$

where T denotes the set of transistors in the transistor group under consideration. $\delta_{trans}{}^{(i)}$ is the mean of the squares of the normalized difference between iterations of the effective resistance. Intuitively, $\delta_{trans}{}^{(i)}$ is a measure of how much the transistor resistances have changed. The algorithm converges when $\delta_{trans}{}^{(i)}$ falls below some threshold value.

Further details can be found in [10].

6. Transistor Group Firing Sequence

We have seen how a transistor group g is simulated within a time interval. Ideally, the voltage waveforms at the gates of transistors in g should have been computed prior to the simulation of g - in other words, the parents of g in the control graph should, as far as possible, be simulated before g.

Consider the simulation of a circuit for which a transient is applied to some node n in a transistor group g. The following tasks are performed:

1 Cycles in the control graph are detected by running a depth-first search algorithm [1] starting from g. Back edges (edges that complete cycles) are marked.

2 The control graph, minus the marked edges, is then a DAG. A breadth-first search [1] is done on the DAG to decide the order in which the transistor groups are processed by the iterative RC network algorithm.

  We note that the DAG defines a partial order on the firing sequence. The breadth-first search algorithm finds a total order compatible with the partial order such that a transistor group is processed only after all ancestor groups in the DAG are processed.

3 The breadth-first search will only process groups that are affected by the input exponential – i.e. groups which can be reached by a directed path from the root group. This way, unaffected groups are not re-computed.

7. Time Steps

As mentioned previously, the resistance of a transistor, $\mathbf{r}(t)$, is a time varying quantity, and cannot be used directly in the corresponding RC network for waveform estimation. Instead, we partition the time axis into small enough intervals so that $\mathbf{r}(t)$ can be assumed to be constant over that time interval.

Notation: $[\mathrm{T}_{j-1}, \mathrm{T}_j]$ denotes the jth time interval. $X^{[j]}$ denotes the quantity X in the jth time interval.

The iterative RC network algorithm provides us with $v_k^{[j]}(t)$ for the jth time interval. Since this is valid for the jth time interval only, we set the voltage at the beginning of the $(j+1)$th time interval to be the voltage reached at the end of the jth time interval:

$$v_k^{[j+1]}(0) = v_k^{[j]}(\mathrm{T}_j - \mathrm{T}_{j-1})$$

We then rerun the iterative RC network algorithm.

This iterative process gives us waveforms $v_k^{[j]}(t)$ for all $j = 1,2,\ldots$, which we put together piecewise to obtain $v_k(t)$.

7.1. Triggering of Transistors

Although it is theoretically feasible to find $v_k^{[j]}(t)$ for $j = 1, 2, \ldots, \infty$, in practice, we need to stop at some finite value for j. This section describes the criterion SuperCrystal uses to determine this value.

Since any transistor changing state can potentially change the topology of the RC network, any measure of convergence can only be applied after all transistors have *triggered*. Once all transistors have triggered, the measure of convergence is the mean of the squares of the difference between iterations of the $v(\infty)$ terms of each node:

Definition 6: $\delta_\mathrm{T}^{[j]}$ is defined as:

$$\delta_\mathrm{T}^{[j]} = \frac{\sum_{k \in N} (v_k^{[j]}(\infty) - v_k^{[j-1]}(\infty))^2}{|N|}$$

where N is the set of the nodes of the RC network under consideration.

Convergence is reported when $\delta_T^{[j]}$ falls below some threshold. Let convergence be reported at the fth time interval. We then define $v_k(t)$ to be:

$$v_k(t) = \begin{cases} v_k^{[j]}(t - T_{j-1}) & T_{j-1} \le t < T_j \le T_{f-1} \\ v_k^{[f]}(t - T_{f-1}) & t > T_{f-1} \end{cases}$$

7.2. The Firing Sequence and Time Steps

A naive approach to simulating a circuit would be to run the iterative RC network algorithm on each transistor group in a given firing sequence for each time step. However, empirical evidence [9] has shown that not all transistor groups need to be simulated for each time step. SuperCrystal uses the above time step convergence criterion to determine dynamically if a transistor group needs to be simulated for the next time step. If a transistor group needs to be simulated, then all descendents of that group in the control graph will also be simulated. If there exists descendents of the current group that preceed it in the firing sequence, then simulation is backed up to the earliest such descendent. Otherwise, simulation continues with the next group in the firing sequence.

8. Experimental Results

Contrived Examples				
Circuit	Description	# Nodes	# Fets	# Transistor Groups
osc	3-inverter ring oscillator	5	6	3
nand4	4-input NAND driving pass-gate	12	9	6
Real-Life Examples				
Circuit	Description	# Nodes	# Fets	# Transistor Groups
rsff	1-bit clocked static RS flip-flop	13	16	7
bit	1-bit static register	23	18	13
eq0	16-bit equal-0 comparator	38	40	21
decode	4 to 16 decoder	74	136	24
decrem	16-bit parallel-prefix decrementer	499	932	317

Table 2: Circuit Sizes

Table 2 gives the seven circuits that were simulated using SuperCrystal and SPICE. In addition, the real-life circuits were analyzed by Crystal. For the simulations, SuperCrystal used a subset of the SPICE parameters for the transistor model. In all the experiments, the parameters for the transistor models for both SuperCrystal and SPICE were identical. The Crystal parameters were derived from these SPICE parameters [17]. In addition, SuperCrystal and SPICE simulated the circuits with the same time steps and time intervals.

8.1. Contrived Examples

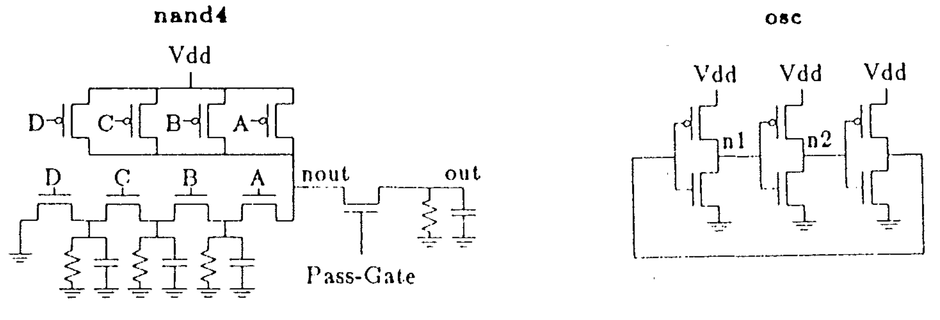

Figure 3: Circuit Diagrams for *nand4* and *osc*.

Figure 3 gives the circuit diagrams for two circuits that demonstrate SuperCrystal's ability to handle "difficult" simulations. For *nand4*, the n-channel transistors on the D, C, and B inputs and the pass transistor have varying voltages at the source terminals. In addition, loading the nodes with parasitic capacitances and leakage resistances "slows" the waveforms.

Steady-state analysis was performed with Input A held at 0V, inputs B, C, and D held at 5V, and the Pass-Gate held at 3V. A 0V→5V step was applied to input A for transient analysis.

The ring oscillator *osc* demonstrates SuperCrystal's ability to handle astable sequential circuits. In addition, there are no load capacitances on the nodes, thus the method by which SuperCrystal estimates the source and drain capacitances on the transistors determines the delay through the inverter and thus the frequency of oscillation. Since *osc* has no steady-state, the initial transient solution was specified with $N1 = 5V$.

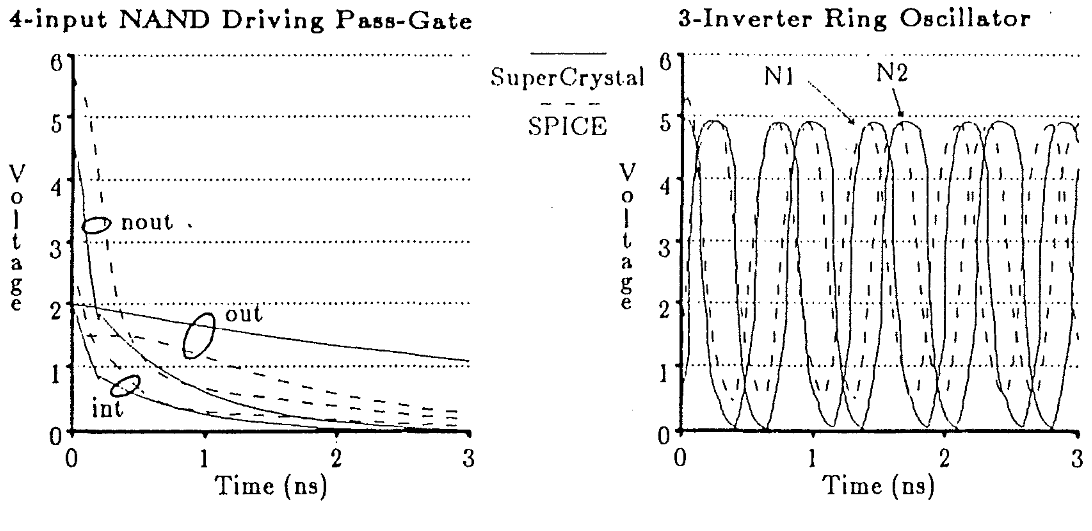

Figure 4: Simulation Waveforms for *nand4* and *osc*.

Figure 4 shows the results from SPICE and SuperCrystal for the *nand4* and *osc* circuits. SuperCrystal models the waveforms in *nand4* quite accurately, despite the fact that the end points of the waveforms are not V_{DD} or GND. In *osc*, SuperCrystal is acceptably close to SPICE even though we do not allow floating capacitors between the gate-source and gate-drain terminals of each transistor in our transistor model.

8.2. Real-life Examples

Simulations were run using Crystal, SuperCrystal, and SPICE on real-life circuit examples. *eq0*, *decode*, and *decrem* are combinational circuits, and *rsff* and *bit* are sequential circuits.

Two simulation runs were performed on *eq0*. In the first run, all inputs were held at 0V for the steady-state analysis. A rising 0V→5V step was applied to least-significant bit for transient analysis. In the second run, steady-state analysis was done with all inputs except for the least significant bit, which was held at 5V. A falling 5V→0V step was applied to the least-significant bit during transient analysis.

For *decode* and *decrem*, the steady-state analysis was done with all inputs at 0V. A rising 0V→5V step was then applied to the least significant bit of the input during transient analysis.

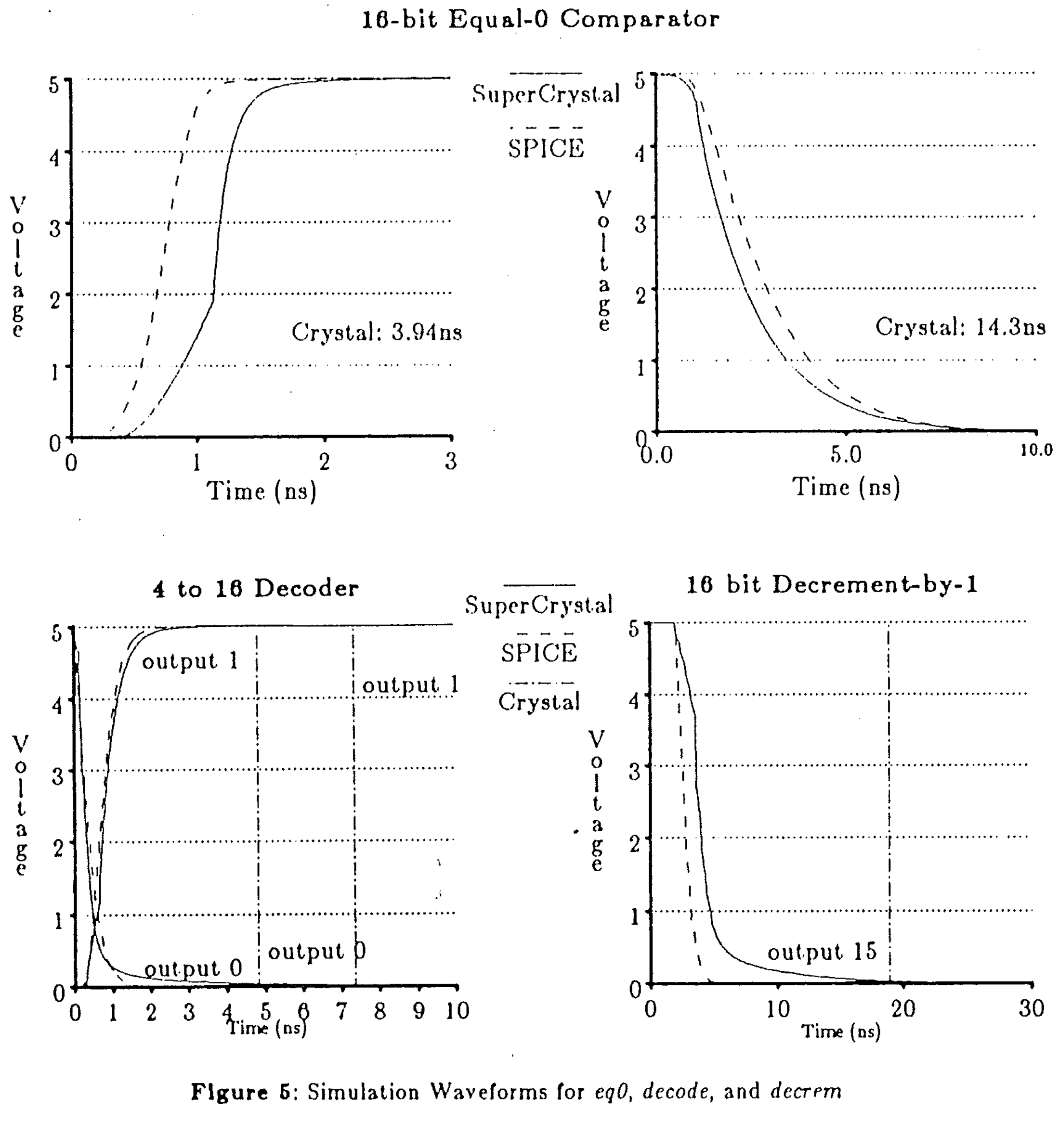

Figure 5: Simulation Waveforms for *eq0*, *decode*, and *decrem*

Figure 5 shows the Crystal delay values, and SuperCrystal and SPICE simulation waveforms for the three combinational circuits. In each case, the SuperCrystal waveform closely tracks the SPICE waveform.

The disparity in the delay for *eq0* arises from the way the circuit is implemented. The inputs to the circuit drive 4-input CMOS NOR gates, with the least significant bit driving the topmost p-transistor in the series chain. It therefore takes longer to pull the output of the NOR gate to V_{DD} through 4 p-transistors, than it takes to pull in to *GND* through 1 n-transistor.

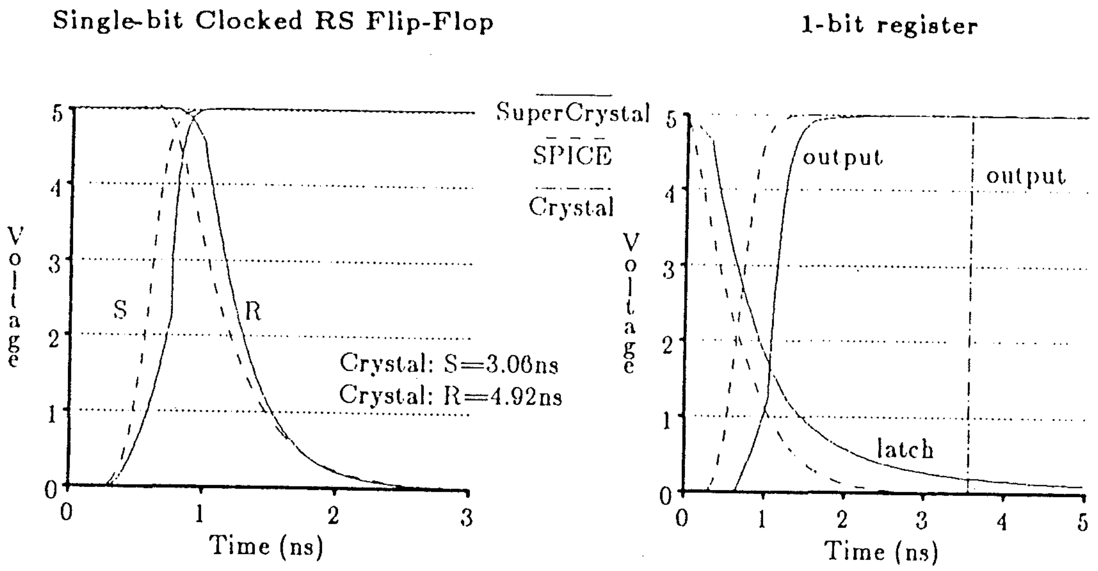

Figure 6: Simulation Waveforms for *rsff* and *bit*.

For *rsff* the steady-state was computed with the flip-flop in its "reset" state and both inputs inactive. A 5V step was applied to the Set input for transient analysis, causing the flip-flop to change state. For *bit* the steady-state was computed with the register storing 0V and input held at 5V, with the register-load disabled. A 0V→5V rising step was applied to *ld* and 5V→0V falling step was simultaneously applied to $\overline{ld}$, thus loading the register with the input value, i.e. 5V.

These two examples were chosen to demostrate SuperCrystal's ability to handle different types of sequential circuits. *rsff* is a pair of back-to-back NAND gates, so that the actual loading and storage functions are realized digitally. *bit* realizes the storage mechanism by a pair of inverters of different strength driving the same node. This node waveform is plotted as *latch*.

8.3. RunTimes

Circuit	Crystal	SPICE	SuperCrystal			SPICE/SuperCrystal Ratio
			Steady-State	Transient	Total	
osc	-	74.0	-	13.2	13.2	5.6
nand4	-	79.7	5.1	2.0	7.1	11.3
bit	0.1	168.3	0.3	20.3	20.6	8.2
rsff	0.1	122.4	0.3	10.0	10.3	11.9
eq0 – up	0.1	296.5	0.6	17.7	18.3	16.2
decode	1.0	1478.4	1.5	31.3	32.8	45.1
eq0 – down	0.1	342.2	0.6	3.3	3.9	87.7
decrem	1.0	19940.3	12.2	33.0	45.2	441.2

Table 3: SuperCrystal and SPICE Runtimes.

Table 3 shows the runtimes of Crystal, SuperCrystal, and SPICE. Note that the SuperCrystal runtimes scale more favorably than SPICE.

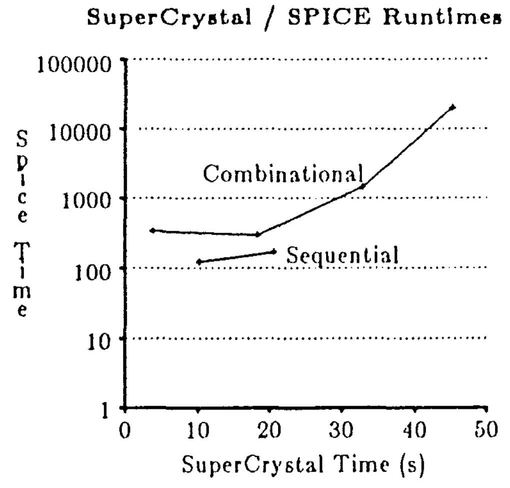

Figure 7: Comparison Plot of SuperCrystal and SPICE Runtimes.

Figure 7 is a comparison plot of SuperCrystal and SPICE runtimes. Note that the SPICE scale is logarithmic. These results show that SuperCrystal is more than just a constant factor faster than SPICE. The gains in runtime appear to increase non-trivially as circuits get larger. In addition, whereas the SPICE runtimes appear in be a function of the size and not the topology of the circuit, the SuperCrystal runtimes appear to depend on both size and topology. In particular, the distinction between combinational and sequential circuits is more pronounced in the SuperCrystal case as is evident from the figure.

9. Conclusions and Future Work

This paper described SuperCrystal, a computer program for the simulation of full-custom VLSI chips. The highlights of SuperCrystal are briefly listed below:

1 The time axis is divided into time steps.

2 In each time step, a transistor circuit maps to a linear RC network for the purpose of waveform estimation.

3 Node voltage waveforms are single time constant exponentials.

4 Node capacitors need not charge all the way to V_{DD}, or discharge all the way to GND, but could take on any real number between V_{DD} and GND.

5 Node waveforms are speedily calculated in the equivalent RC network.

6 Transistor channel resistances are not independent of gate waveforms; instead, SuperCrystal distinguishes between the TRIODE and PINCHOFF modes of operation, and arrives at an effective resistance over a given time step that is a function of the gate, source and drain waveforms.

7 SuperCrystal allows the introduction of other transistor models. In particular, models used by the various levels of SPICE can be incorporated. Table-lookup models can also be incorporated.

A version of SuperCrystal has been coded and is running on Berkeley 4.3 UNIX. The program has been distributed to various on-campus test-sites. We expect a version available for external users shortly.

Further work is required in the following areas. Empirical evidence indicates that the iterative RC network algorithm converges quickly for the MOS circuits analyzed. It still remains to be shown that it will converge in all cases.

Further studies are required on the effect of varying the length of timesteps. The relationship between the convergence criteria, δ_T and δ_{trans}, and the length of the timestep needs careful investigation.

We are currently developing a new algorithm to obtain exponential approximations to the waveforms for all n nodes in a transistor group in $O(n)$ time.

Finally, extensive code optimization needs to be done to achieve faster running times.

10. Acknowledgements

The authors would like to thank Prof. John Ousterhout for graciously allowing us to look at the Crystal source code. Thanks are also due to Prof. Alberto Sangiovanni-Vincentelli for many useful comments, and Prof. Eugene Lawler for his helpful support throughout the course of this project.

References

1. A. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

2. R.E. Bryant, "A Switch-Level Simulation Model for Integrated Logic Circuits," *MIT/LCS/TR-259*, Doctoral Thesis, MIT, Mar. 1981.

3. C.A. Desoer and E.S. Kuh, *Basic Circuit Theory*, McGraw Hill, New York, 1969.

4. N.P. Jouppi, "TV: An nMOS Timing Analyzer," *Proc. 3rd CalTech Conf. VLSI*, pp. 71-86, Mar. 1983.

5. Young Hwan Kim, "ELOGIC: A Relaxation-Based Switch-Level Simulation Technique," *UCB/ERL M86/2*, UC Berkeley, 3 Dec 1985.

6. T. Lin and C.A. Mead, "Signal Delay in General RC Networks," *IEEE Trans. CAD*, vol. CAD-3, pp. 331-349, Oct. 1984.

7. L.W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," *ERL Memo ERL-M520*, UC Berkeley, May 1975.

8. Sani R. Nassif and Stephen W. Director, "WASIM: A Waveform Based Simulator for VLSICs," *Proc. ICCAD*, pp. 29-31, 1985.

9. A.R. Newton and A.L. Sangiovanni-Vincentelli, "Relaxation-Based Electrical Simulation," *IEEE Transactions on Computer-Aided Design*, vol. CAD-3, pp. 308-330, Oct. 1984.

10. Antony P-C Ng, A. Raghunathan, and Clark D. Thompson, "Incorporating Input Slopes in the Linear RC Model for MOS VLSI Signal Delays," *Technical Report (in preparation).*

11. J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables,* New York Academic Press, 1970.

12. J.K. Ousterhout, "Crystal: A Timing Analyzer for nMOS VLSI Circuits," *Report No. UCB/CSD 83/115*, Computer Sciences Division, UC Berkeley, Jan. 1983.

13. A. Raghunathan and C.D. Thompson, "Signal Delay in RC Trees with Charge Sharing or Leakage," *Report No. UCB/CSD 85/243*, Computer Science Division, UC Berkeley, June 1985.

14. A. Raghunathan and C.D. Thompson, "Signal Delay in RC Trees with Charge Sharing or Leakage," *Proc. Nineteenth Asilomar Conference on Circuits, Systems and Computers*, Nov. 6-8, 1985.

15. J. Rubinstein, P. Penfield, and M. Horowitz, "Signal Delays in RC Tree Networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202-211, July 1983.

16. Resve A. Saleh, "Iterated Timing Analysis and SPLICE1," *UCB/ERL M84/2*, University of California, Berkeley, Jan 1984.

17. W.S. Scott, R.N. Mayo, G. Hamachi, and J.K. Ousterhout, "1986 VLSI Tools: Still More Works by the Original Artists," *Report No. UCB/CSD 86/272*, Computer Science Division, UC Berkeley, December 1985.

18. H. Shichman and D. A. Hodges, "Modeling and simulation of insulated gate field-effect transistor switching circuits," *IEEE J. Solid-State Circuits*, vol. SC-3, pp. 285-289, Sept., 1968.

19. E. Tamura, K. Ogawa, and T. Nakano, "Path Delay Analysis for Hierarchical Building Block Layout System," *Proc. 20th Design Automation Conf.*, pp. 403-410, 1983.

20. J. Varga, *Matrix Iterative Analysis*, Prentice Hall, 1969.

21. A. Vladimirescu and S Liu, "The Simulation of MOS Integrated Circuits Using SPICE2," *UCB/ERL M80/7,* University of California, Berkeley, Oct 1980.

22. J. White and A.L. Sangiovanni-Vincentelli, "Relax2.1: A Waveform Relaxation Based Circuit Simulation Program," *Proc. 1984 Custom Integrated Circuits Conference*, pp. 232-236, May 1984.