Simulating Switch-Level Networks with Uncertain Transistor Strengths and Node Sizes

Cheryl L. Harkness
Daniel P. Lopresti

Technical Report No. CS-89-07

February 1989

Simulating Switch-Level Networks with Uncertain Transistor Strengths and Node Sizes †

Cheryl L. Harkness
Daniel P. Lopresti

Department of Computer Science
Brown University
Providence, RI 02912

February 16, 1989

Abstract

Numerous techniques for performing switch-level simulation have been developed in recent years. Without exception, these simulators require that the relative conductances, or strengths, of all transistors be precisely specified. We explore the problem of simulating networks with uncertain transistor strength. To solve this problem, we examine the underlying mathematical structure of an existing switch-level simulation algorithm, MOSSIM II, and extend this structure to generate an algorithm for simulating networks with uncertain transistor strengths and node sizes. Such an algorithm may be useful in an integrated design-simulation environment, where designers test circuits while creating them.

1. Introduction

Numerous techniques for performing rapid switch-level simulation have been developed in recent years [1, 2, 3, 4, 5, 6]. Without exception, these methods require detailed knowledge of circuit implementation. In particular, the relative conductances, or strengths, of all transistors must be specified in advance. As a result, the circuit shown in Figure 1 cannot be simulated, even though its output is undefined only when both transistors are conducting.

While strength information can be derived from low-level circuit descriptions (e.g. device geometry), it is rarely present in the higher-level specifications used in today's advanced design tools. Uncertainty is inherent in top-down design, where all of the implementation details may not be available when testing the circuit at an intermediate design stage. Simulating a transistor network to verify its behavior before calculating the exact transistor size ratios is one such example.

In this paper, we describe an extension to an existing switch-level simulator, MOSSIM II [2], capable of handling uncertain transistor strengths. The MOSSIM II algorithm assumes a certain underlying mathematical structure. We represent transistor strengths by ranges of values and extend this structure to accomodate these strength intervals. Our method is simple, yet conservative in the presence of uncertainty. The resulting algorithm has a time complexity comparable to that of the

†This research was supported in part by the Defense Advanced Research Projects Agency under ONR contract number N00014-83-K-0146. Daniel Lopresti is also supported by the National Science Foundation under contract MIP 8710745.

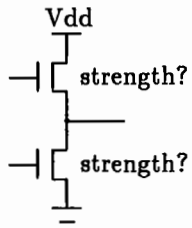

Figure 1: A simple transistor network (strengths not specified)

gate	n-type	p-type	d-type
0	open	closed	closed
1	closed	open	closed
X	indeterminate	indeterminate	closed

Table 1: Transistor State Table

original. In addition, we show that our algorithm can be applied to solve the analogous problem of unknown node size. Finally, we present a formal proof of correctness for our method.

2. The Transistor Model

2.1. Traditional Approach

In switch-level simulation, a transistor is modeled as an ideal switch, in which the gate of the transistor controls the flow of current between the other two terminals. The state of a transistor (i.e. whether the switch is *open* (nonconducting), *closed* (conducting), or *indeterminate*) is uniquely determined by the transistor type and the signal value at its gate. Table 1 gives this function for the three transistor types most commonly used in VLSI technologies.

A network of transistors is typically modeled as a weighted, undirected graph. The vertices of this graph represent the source and drain terminals of the transistors in the network and the edges indicate the connectivity between these terminals. The vertices of the graph are divided into two mutually exclusive classes: inputs (which are direct connections to the voltage sources Vdd and Gnd) and storage nodes (everything else). Associated with each vertex are two attributes: signal value and node size. The first attribute is the current signal value of the node (0, 1, or X). The size of a node is a measure of its capacitance. Rather than using the actual capacitances themselves, switch-level simulators usually employ a set of discrete nodes sizes. If the vertex is an input node, it is assigned size ω, indicating infinite capacitance. All other nodes are ranked by capacitance and assigned values from the set $\{\kappa_1,\ldots,\kappa_{max}\}$.

A graph edge drawn between two vertices indicates that the transistor linking that pair of nodes is *potentially* closed. There are two types of edges: *1-edges* for transistors in a *closed* state and *X-edges* for transistors in an *indeterminate* state. Associated with each edge is a strength attribute, which is a measure of the conductance of the transistor. Conductance is determined by the geometry

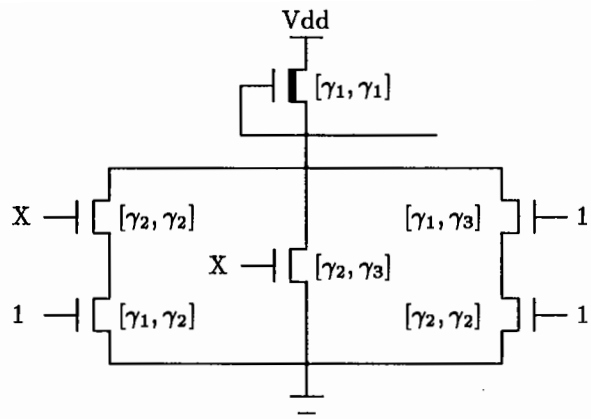

Figure 2: Transistor network with unknown transistor strengths

of a transistor and, in particular, by the ratio of its length to its width. Instead of using the actual ratios in the simulation, switch-level algorithms employ a set of discrete strengths, like those used to specify node sizes. To determine these strengths, the transistors are ranked according to ratio size and transistor type and are assigned values, indicating relative conductance, from a set of discrete possibilities, $\{\gamma_1,...,\gamma_{max}\}$.

2.2. Unknown Transistor Strength

Since a transistor with uncertain strength may assume any strength chosen from a subset of the strengths $\{\gamma_1,...,\gamma_{max}\}$, we represent a transistor's strength by a range of these values or an interval. If T is the ordered set of transistor strengths where $\gamma_1 < \gamma_2 < ... < \gamma_{max}$, then an interval $[a, b]$ over the set T is defined as the subset $\{x \in T \mid a \leq x \leq b\}$. A *degenerate* interval is one in which $a = b$. Degenerate intervals allow us to model cases where the relative strength of a transistor is precisely known; thus, our extension subsumes current switch-level simulation algorithms.

2.3. An Example

Figure 2 depicts a transistor network containing unknown transistor strengths. All transistor strengths are assigned as intervals over the set T. From this network, the connectivity graph in Figure 3 is generated. Each graph vertex is labeled with a unique identifier followed by its signal value and size attributes in parentheses. All edges are drawn and labeled with their associated strength. 1-edges appear as solid lines and X-edges as dashed lines.

3. The Simulation Algorithm

To analyze a network containing transistors with unknown strength, we must effectively determine the steady-state value of the network under all possible strength assignments to each transistor of unknown strength. If the steady-state value of a node is the same regardless of the strengths assigned to these transistors, then that signal value is taken to be the steady-state value of the node.

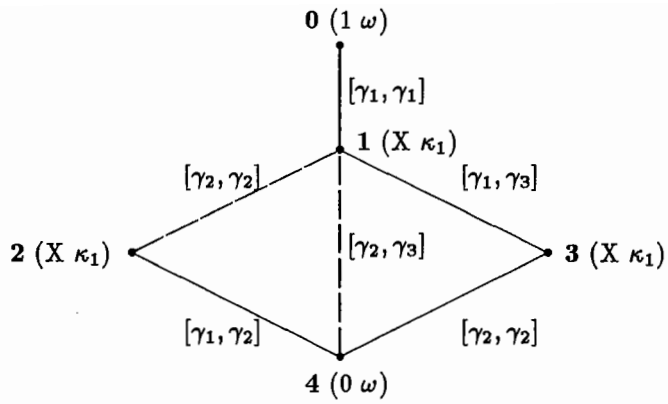

Figure 3: Connectivity graph of network with unknown transistor strength

If, however, different strength assignments yield different signal values for the same node, then the steady-state value of the node is taken to be X.

Table 2 presents this analysis for the network in Figure 3. All combinations of possible strengths are assigned to the unknown-strength transistors in the network and the resulting steady-state is computed using the original MOSSIM II algorithm. Since the network must be analyzed for each possible strength combination, the basic simulation algorithm is executed $\prod_{i=1}^{k} g_i$ times, where k is the number of transistors with unknown strength and g_i is the number of different strengths that can be assigned to transistor i. This method has a worst-case time complexity of $O(g^k)$, where k is number of transistors with unknown strength and g is the maximum number of different transistor strengths. Clearly, a more efficient technique is needed. To remedy this problem, we create an interval version of MOSSIM II which more efficiently computes the steady-state value of the network. Our extended version has the same time complexity as the original presented in [2].

3.1. The MOSSIM II Simulation Algorithm

In this section, we review the MOSSIM II switch-level simulation algorithm as presented in [2]. The resultant steady-state value of a node in the network is determined by the strongest signal path through the network to reach the node. The strength of a path is defined as the minimum of the source node size and the strength of the weakest transistor in the path. Bryant identifies three types of paths distinguished by their strengths: input paths (with strength ω), driven paths (with strengths in the range γ_1 to γ_{max}), and stored-charge paths (with strengths in the range κ_1 to κ_{max}). He adds an identity element, λ, to these path strengths and orders them as follows:

$$\omega > \gamma_{max} > \ldots > \gamma_1 > \kappa_{max} > \ldots > \kappa_1 > \lambda \tag{1}$$

He also introduces three operations on this ordered set of values: $+$, $\cdot$, and $\sim$. The first two operators, $+$ and $\cdot$, are the standard maximum and minimum functions. The third operator, $\sim$, is a blocking function defined as follows:

$$a \sim b \equiv \begin{cases} a & \text{if } a \geq b \\ \lambda & \text{otherwise} \end{cases} \tag{2}$$

Using these operations, the steady-state value of a transistor network is computed. The MOSSIM II algorithm has four distinct stages: describing the network, calculating blocking path strength, finding

Strength			Steady-state				
$t_{2,4}$	$t_{1,4}$	$t_{1,3}$	0	1	2	3	4
γ_1	γ_2	γ_1	1	X	X	0	0
γ_1	γ_2	γ_2	1	0	0	0	0
γ_1	γ_2	γ_3	1	0	0	0	0
γ_1	γ_3	γ_1	1	X	X	0	0
γ_1	γ_3	γ_2	1	0	0	0	0
γ_1	γ_3	γ_3	1	0	0	0	0
γ_2	γ_2	γ_1	1	X	0	0	0
γ_2	γ_2	γ_2	1	0	0	0	0
γ_2	γ_2	γ_3	1	0	0	0	0
γ_2	γ_3	γ_1	1	X	0	0	0
γ_2	γ_3	γ_2	1	0	0	0	0
γ_2	γ_3	γ_3	1	0	0	0	0
Final values			1	X	X	0	0

Table 2: Combinational analysis of network containing unknown strengths

unblocked pull-up and pull-down paths, and finally computing the steady-state.

Creating the matrices which describe the transistor network is the first stage of the simulation algorithm. The first two matrices, s and y, contain the size and initial signal value of each vertex in the matrix; thus, s_i contains the size of node i and y_i its initial value. The connectivity between these nodes is given by two matrices, $G1$ and GX; one to hold the *1-edges* and the other to hold the *X-edges*. If there is a closed transistor joining nodes i and j in the transistor network, then there is a 1-edge between these nodes in the corresponding connectivity graph. An entry for this edge is made in matrix $G1$ by assigning the transistor strength of this closed transistor to $G1(i,j)$. Since connectivity is a commutative property, $G1(i,j) = G1(j,i)$. After all of the 1-edges are recorded, the remaining locations in $G1$ are filled with λ. The matrix GX is defined similarly, recording *X-edges* instead of *1-edges*.

The second step of the simulation algorithm is to compute the blocking path strength for each of the nodes in the network. When computing path strength, signal paths containing X-edges pose problems. Since a transistor with indeterminate state may be either closed or open, the corresponding edge may or may not actually exist in the graph. If considered, strong signal paths containing X-edges may prevent weaker signal paths which do not contain X-edges from influencing the signal value at a node. To prevent this undesirable condition, X-edges are ignored when computing blocking path strength. Signal paths which do not contain any X-edges are called *definite* paths; thus, blocking path strength is defined as the strength of the strongest definite path to reach a vertex. According to the MOSSIM II algorithm, this blocking path strength is derived using the following recurrence:

$$\begin{aligned} q(i,0) &= s_i \\ q(i,r) &= s_i + \sum_{j=1}^{n} G1(i,j) \cdot q(j,r-1) \end{aligned} \tag{3}$$

The entry $q(i,r)$ gives the strength of the strongest definite path with length less than or equal to r to reach vertex i. Note that r is bounded by the number of vertices n in the graph; thus, $q_i = q(i,n)$. This new matrix q is the blocking strength matrix.

The third step is to compute the strongest unblocked pull-up and pull-down paths to each vertex. To perform this calculation, the algorithm begins at nodes with signal values of 1 (or X) and traces the paths from these nodes to all other vertices in the connectivity graph, computing the strongest

pull-up path to each vertex. In contrast to the previous step, both definite and indefinite paths are included in this computation. Once the strongest pull-up path to a vertex is computed, it is compared with the blocking path strength for that vertex using the $\sim$ operator given in equation 2. If the strength of this pull-up path is less than the blocking path strength, then a stronger definite path exists; therefore, this pull-up path is disregarded. Only paths of strength greater than or equal to the blocking strength are recorded. By starting at nodes with signal values of 0 (or X), the strongest unblocked pull-down paths are similarly computed.

Matrices u and d contain the strength of the strongest unblocked pull-up and pull-down paths to each vertex. To assist in generating these matrices, two new functions **up** and **down** are introduced. These functions identify the sources of possible pull-up and pull-down paths and are defined as follows:

$$\mathbf{up}(s_i, y_i) \equiv \begin{cases} s_i & \text{if } y_i = 1 \text{ or } X \\ \lambda & \text{if } y_i = 0 \end{cases} \tag{4}$$

$$\mathbf{down}(s_i, y_i) \equiv \begin{cases} s_i & \text{if } y_i = 0 \text{ or } X \\ \lambda & \text{if } y_i = 1 \end{cases} \tag{5}$$

Given these two functions, the matrices u and d are generated by the following recurrences:

$$\begin{aligned} u(i,0) &= \mathbf{up}(s_i, y_i) \sim q_i \\ u(i,r) &= \left[\mathbf{up}(s_i, y_i) + \sum_{j=1}^{n} (G1(i,j) + GX(i,j)) \cdot u(j, r-1)\right] \sim q_i \qquad (6) \\ d(i,0) &= \mathbf{down}(s_i, y_i) \sim q_i \\ d(i,r) &= \left[\mathbf{down}(s_i, y_i) + \sum_{j=1}^{n} (G1(i,j) + GX(i,j)) \cdot d(j, r-1)\right] \sim q_i \qquad (7) \end{aligned}$$

The entry $u(i,r)$ gives the strength of the strongest unblocked pull-up path with length less than or equal to r and destination node i. As with q, the recursion terminates in at most n steps; thus, u_i is assigned the value $u(i,n)$. Matrix d is computed analogously.

In the final step, the results of the previous computation are used to determine the steady-state response of the network. If there is an unblocked pull-up path to a vertex and no unblocked pull-down paths, then the new signal value at the vertex is 1. Similarly, if there is an unblocked pull-down path to the vertex and no unblocked pull-up paths, then the new signal value is 0. If both unblocked pull-up and pull-down paths exist, then the signal value is taken to be X. This function is defined as follows:

$$Y_i \equiv \begin{cases} 1 & \text{if } d_i = \lambda \\ 0 & \text{if } u_i = \lambda \\ X & \text{otherwise} \end{cases} \tag{8}$$

3.2. Extensions to Handle Unknown Transistor Strength

To create an algorithm for simulating transistor networks containing uncertain transistor strengths, we modify the path analysis algorithm given above to accommodate intervals. In particular, we define three interval operators which are the logical extensions of the discrete operators $+$, $\cdot$, and $\sim$. The maximum operator for intervals $\oplus$ and the minimum operator $\odot$ are derived from their discrete counterparts as follows:

$$[a,b] \oplus [c,d] \equiv [a+c, b+d] \tag{9}$$

$$[a,b] \odot [c,d] \equiv [a \cdot c, b \cdot d]. \tag{10}$$

	s_i	y_i
0	$[\omega, \omega]$	1
1	$[\kappa_1, \kappa_1]$	X
2	$[\kappa_1, \kappa_1]$	X
3	$[\kappa_1, \kappa_1]$	X
4	$[\omega, \omega]$	0

Table 3: Describing the network – the vertices

G1					
	0	1	2	3	4
0	$[\lambda, \lambda]$	$[\gamma_1, \gamma_1]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$
1	$[\gamma_1, \gamma_1]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\gamma_1, \gamma_3]$	$[\lambda, \lambda]$
2	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\gamma_1, \gamma_2]$
3	$[\lambda, \lambda]$	$[\gamma_1, \gamma_3]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\gamma_2, \gamma_2]$
4	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\gamma_1, \gamma_2]$	$[\gamma_2, \gamma_2]$	$[\lambda, \lambda]$

GX					
	0	1	2	3	4
0	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$
1	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\gamma_2, \gamma_2]$	$[\lambda, \lambda]$	$[\gamma_2, \gamma_3]$
2	$[\lambda, \lambda]$	$[\gamma_2, \gamma_2]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$
3	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$
4	$[\lambda, \lambda]$	$[\gamma_2, \gamma_3]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$	$[\lambda, \lambda]$

Table 4: Describing the network – the edges

In addition to these two operators, we introduce a blocking operator for intervals $\simeq$ defined as follows:

$$[a, b] \simeq [c, d] \equiv \begin{cases} [a + c, b] & \text{if } b \geq c \\ \Lambda & \text{otherwise} \end{cases} \tag{11}$$

where Λ is the degenerate interval $[\lambda, \lambda]$. Each of these definitions is obtained by applying the associated discrete function to all combinations of elements from each interval and taking the union of the results $(F([a, b], [c, d]) \equiv \bigcup f(x, y)$ where $x \in [a, b]$ and $y \in [c, d]$. This is called calculating the *united extension* of function f).

Having computed these interval operations, we substitute them into the path analysis equations given in the previous section to generate a switch-level simulation algorithm which handles unknown transistor strength. A proof of correctness for this method is given in the appendix of this paper.

3.3. Examples

Tables 3-5 present the analysis of the transistor network given in Figure 3. The first two tables describe the initial state of the transistor network. Table 5 displays the resultant blocking path strength, unblocked pull-up and pull-down paths, and the final steady-state for each node in the network. Note that the steady state values computed are identical to those derived in Table 2, as expected.

To further illustrate how the interval operations are employed to determine these results, we examine the computations for one of the nodes. Table 6 enumerates all paths through the network which terminate at node 3. The table gives both the signal value and strength of each path. To calculate the blocking path strength of node 3, we find the strength range of the strongest definite path. In this computation, we consider four paths: the path from node 0 through the pull-up transistor to node 3 (with strength $[\gamma_1, \gamma_1]$), the path originating at node 1 (with strength $[\kappa_1, \kappa_1]$), the path consisting of node 3 itself (with strength $[\kappa_1, \kappa_1]$), and the pull-down path originating at node 4 (with strength $[\gamma_2, \gamma_2]$). The blocking path strength is found by computing the maximum of these strengths; thus, $q_3 = [\gamma_1, \gamma_1] \oplus [\kappa_1, \kappa_1] \oplus [\kappa_1, \kappa_1] \oplus [\gamma_2, \gamma_2] = [\gamma_2, \gamma_2]$. In the next stage of the algorithm, we calculate unblocked pull-up and pull-down path strengths. We accomplish this by

	q_i	u_i	d_i	Y_i
0	$[\omega,\omega]$	$[\omega,\omega]$		1
1	$[\gamma_1,\gamma_2]$	$[\gamma_1,\gamma_1]$	$[\gamma_2,\gamma_3]$	X
2	$[\gamma_1,\gamma_2]$	$[\gamma_1,\gamma_1]$	$[\gamma_2,\gamma_2]$	X
3	$[\gamma_2,\gamma_2]$		$[\gamma_2,\gamma_3]$	0
4	$[\omega,\omega]$		$[\omega,\omega]$	0

Table 5: Interval algorithm results

Path	X-edges?	Value	Strength
0,1,3	No	1	$[\omega,\omega]\odot[\gamma_1,\gamma_1]\odot[\gamma_1,\gamma_3]=[\gamma_1,\gamma_1]$
1,3	No	X	$[\kappa_1,\kappa_1]\odot[\gamma_1,\gamma_3]=[\kappa_1,\kappa_1]$
2,1,3	Yes	X	$[\kappa_1,\kappa_1]\odot[\gamma_2,\gamma_2]\odot[\gamma_1,\gamma_3]=[\kappa_1,\kappa_1]$
3	No	X	$[\kappa_1,\kappa_1]$
4,3	No	0	$[\omega,\omega]\odot[\gamma_2,\gamma_2]=[\gamma_2,\gamma_2]$
4,1,3	Yes	0	$[\omega,\omega]\odot[\gamma_2,\gamma_3]\odot[\gamma_1,\gamma_3]=[\gamma_1,\gamma_3]$
4,2,1,3	Yes	0	$[\omega,\omega]\odot[\gamma_1,\gamma_2]\odot[\gamma_2,\gamma_2]\odot[\gamma_1,\gamma_3]=[\gamma_1,\gamma_2]$

Table 6: Network paths to Node 3

finding the strength ranges of the strongest pull-up and pull-down paths and then comparing them to the blocking path strength to eliminate blocked paths. The strongest pull-down path strength is the maximum strength of all paths originating at nodes with value 0 or X; therefore, the maximum pull-down path strength is $[\kappa_1,\kappa_1]\oplus[\kappa_1,\kappa_1]\oplus[\kappa_1,\kappa_1]\oplus[\gamma_2,\gamma_2]\oplus[\gamma_1,\gamma_3]\oplus[\gamma_1,\gamma_2]=[\gamma_2,\gamma_3]$. Likewise, the strongest pull-up path strength is $[\gamma_1,\gamma_1]$. When compared to the blocking path strength for node 3, we discover that there are no unblocked pull-up paths (since $[\gamma_1,\gamma_1]\simeq[\gamma_2,\gamma_2]=\Lambda$), but that unblocked pull-down paths do exist (since $[\gamma_2,\gamma_3]\simeq[\gamma_2,\gamma_2]=[\gamma_2,\gamma_3]$). Therefore, we conclude that the steady-state value of node 3 is 0.

The CMOS selector shown in Figure 4 is another example. Since no explicit strengths are given, we assign strength $[\gamma_1,\gamma_{max}]$ to all transistors and simulate the network. The simulation results given in Table 7 show that if all the transistors in the circuit have equal strength, then the network does not exhibit the desired switching behavior when the selected input value changes from 1 to 0. Instead of changing from 0 to 1, the output becomes undefined. Because the pull-up path is too strong, it competes with the input signal, thus producing an undefined steady-state. The conflict is resolved by assigning a weak transistor strength to the uppermost p-transistor and giving all other transistors in the network greater relative strength. Once this change is made, the circuit exhibits the desired behavior as shown in Table 7.

3.4. Performance

The interval path analysis algorithm has been implemented in the C programming language and runs on several machines (a VAX-11/780 and an Encore Multimax multiprocessor). We ran both programs on identical circuits and recorded their response times and found that our extension to MOSSIM II analyzes networks with speeds comparable to the original. Our experiments show that the interval version takes about twice as long as MOSSIM II to compute the steady-state value of a transistor network. This is the expected result, since the interval algorithm performs two calculations for each minimum, maximum, or blocking operator encountered. The worst-case time complexity of both implementations is $O(s+t)$, where t represents the number of transistors and s represents the

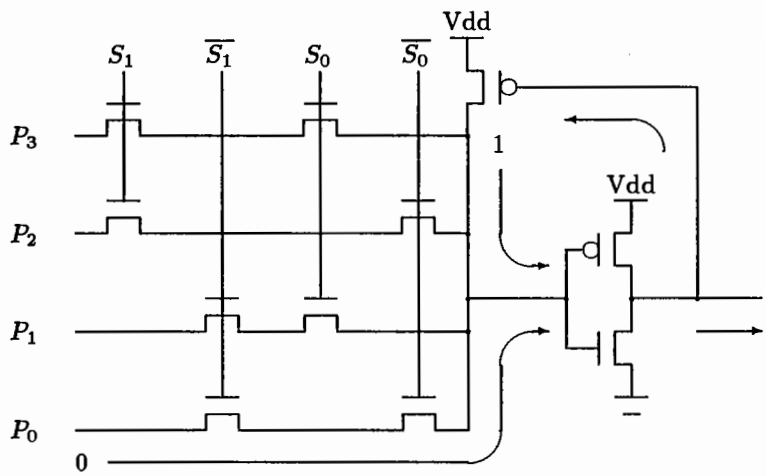

Figure 4: CMOS selector with feedback

P_i transition	Initial pull-up state	Equal strength Output value	Weak pull-up Output value
$0 \rightarrow 0$	open	1	1
$1 \rightarrow 0$	closed	X	1
$X \rightarrow 0$	indeterminate	X	1
$0 \rightarrow 1$	open	0	0
$1 \rightarrow 1$	closed	0	0
$X \rightarrow 1$	indeterminate	0	0
$0 \rightarrow X$	open	X	X
$1 \rightarrow X$	closed	X	X
$X \rightarrow X$	indeterminate	X	X

Table 7: Simulation results for CMOS selector circuit

number of distinct node sizes and transistor strengths.

3.5. Extensions to Handle Uncertain Node Size

It should be obvious that our interval version of the switch-level simulation algorithm can also be applied to uncertain node sizes, since node size and transistor strength have the same underlying algebraic structure. To handle uncertain node sizes, we represent them as intervals over the ordered set $\{\kappa_1, \kappa_2, ..., \kappa_{max}, \omega\}$. We then employ the interval operations given above to manipulate these size ranges. No further changes to the algorithm are required.

4. Conclusions

In this paper, we considered the problem of simulating transistor networks containing uncertain transistor strengths and node sizes. We represented uncertain transistor strengths by strength ranges and developed an interval algebra to manipulate these ranges. We further generalized an existing switch-level simulator, MOSSIM II, to create a simulation algorithm for networks with interval transistor strengths. Our algorithm has an operating speed of about twice that of the original and a comparable worst-case time complexity. In addition, this extension may be applied to solve the analogous problem of unknown node sizes. Finally, we presented a formal proof of correctness for our method.

Appendix A. Proof of the Method

In this section, we formally prove that our interval method correctly computes the resultant steady-state value of a transistor network with interval transistor strengths and node sizes. In his presentation of MOSSIM II, Bryant lists the fundamental properties of his path analysis algebra. He further proves that his algorithm works, because these properties hold. In the next section, we summarize these properties. We then present our interval algebra and show that it exhibits the same behavior. Finally, we conclude that if the original algebra correctly analyzes a circuit with discrete transistor strengths and node sizes, then our extended algebra will correctly analyze the same circuit with interval transistor strengths and node sizes.

A.1. Bryant's algebra

Let S be an ordered set of discrete transistor strengths and node sizes

$$S \equiv \{\lambda, \kappa_1, ..., \kappa_{max}, \gamma_1, ..., \gamma_{max}, \omega\}, \tag{12}$$

where

$$\lambda < \kappa_1 < ... < \kappa_{max} < \gamma_1 < ... < \gamma_{max} < \omega. \tag{13}$$

Maximum $+$ and minimum $\cdot$ are two operations over the set S. The algebra $\langle S, +, \cdot, \lambda, \omega \rangle$ forms a closed semi-ring. A third operator, the blocking path operator $\sim$, is also defined over the set S.

$$a \sim b \equiv \begin{cases} a & \text{if } a \geq b \\ \lambda & \text{otherwise} \end{cases} \tag{14}$$

These three operations have the following properties:

1. Properties of $+$
   (a) $+$ is commutative: $a + b = b + a$
   (b) $+$ is associative: $(a + b) + c = a + (b + c)$
   (c) $+$ has identity element λ: $a + \lambda = \lambda + a = a$
   (d) $+$ is closed over S: $a, b \in S \Rightarrow a + b \in S$
   (e) $+$ is idempotent: $a + a = a$
   (f) $+$ is monotonic: $a \geq b \Rightarrow (a + c) \geq (b + c)$ *and* $(c + a) \geq (c + b)$
2. Properties of $\cdot$
   (a) $\cdot$ is associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
   (b) $\cdot$ has identity element ω: $a \cdot \omega = \omega \cdot a = a$
   (c) $\cdot$ is closed over S: $a, b \in S \Rightarrow a \cdot b \in S$
   (d) λ is the annihilator for $\cdot$: $a \cdot \lambda = \lambda \cdot a = \lambda$
   (e) $\cdot$ is monotonic: $a \geq b \Rightarrow (a \cdot c) \geq (b \cdot c)$ *and* $(c \cdot a) \geq (c \cdot b)$
3. $\cdot$ distributes over $+$: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
4. Properties of $\sim$
   (a) $\sim$ is idempotent: $a \sim a = a$
   (b) $\sim$ distributes over $+$: $(a + b) \sim c = (a \sim c) + (b \sim c)$
   (c) $\sim$ distributes over $\cdot$: $(a \cdot b) \sim c = (a \sim c) \cdot (b \sim c)$
   (d) $\sim$ is monotonic in its first argument: $a \geq b \Rightarrow (a \sim c) \geq (b \sim c)$

A.2. The Interval Algebra

An interval $[a, b]$ over the set S is defined as the set $\{x \in S | a \leq x \leq b\}$. Let S' be the set of all intervals over S; thus, $S' \equiv \{[a, b] | a, b \in S \text{ and } a \leq b\}$. Further, we introduce two operations on the set S': the maximum operator for intervals $\oplus$ and the minimum operator for intervals $\odot$. These are defined as follows:

$$[a, b] \oplus [c, d] \equiv [a + c, b + d] \tag{15}$$

$$[a, b] \odot [c, d] \equiv [a \cdot c, b \cdot d]. \tag{16}$$

The identity element for $\oplus$ is $\Lambda = [\lambda, \lambda]$ and the unit element for $\odot$ is $\Omega = [\omega, \omega]$. The algebra $\langle S', \oplus, \odot, \Lambda, \Omega \rangle$ forms a closed semi-ring. In addition to these two operators, we introduce a blocking operator for intervals $\simeq$ defined as follows:

$$[a, b] \simeq [c, d] \equiv \begin{cases} [a + c, b] & \text{if } b \geq c \\ \Lambda & \text{otherwise} \end{cases} \tag{17}$$

Now we must show that these interval operators obey the same properties as their discrete counterparts.

A.3. The Proof

1. Properties of $\oplus$

   (a) $\oplus$ is commutative:
   Proof. Given $[a, b], [c, d] \in S'$, we must show that $[a, b] \oplus [c, d] = [c, d] \oplus [a, b]$.
   $[a, b] \oplus [c, d] = [a + c, b + d]$ (Definition of $\oplus$).
   $[a, b] \oplus [c, d] = [c + a, d + b]$ (Commutativity of $+$).
   $[a, b] \oplus [c, d] = [c, d] \oplus [a, b]$ (Definition of $\oplus$).

   (b) $\oplus$ is associative:
   Proof. Given $[a, b], [c, d], [e, f] \in S'$, we must show that $([a, b] \oplus [c, d]) \oplus [e, f] = [a, b] \oplus ([c, d] \oplus [e, f])$.
   $([a, b] \oplus [c, d]) \oplus [e, f] = [a + c, b + d] \oplus [e, f]$ (Definition of $\oplus$).
   $([a, b] \oplus [c, d]) \oplus [e, f] = [(a + c) + e, (b + d) + f]$ (Definition of $\oplus$).
   $([a, b] \oplus [c, d]) \oplus [e, f] = [a + (c + e), b + (d + f)]$ (Associativity of $+$).
   $([a, b] \oplus [c, d]) \oplus [e, f] = [a, b] \oplus ([c + e, d + f])$ (Definition of $\oplus$).
   $([a, b] \oplus [c, d]) \oplus [e, f] = [a, b] \oplus ([c, d] \oplus [e, f])$ (Definition of $\oplus$).

   (c) $\oplus$ has identity element Λ:
   Proof. Given $[a, b] \in S'$, we must show that $[a, b] \oplus \Lambda = \Lambda \oplus [a, b] = [a, b]$.
   $[a, b] \oplus \Lambda = [a, b] + [\lambda, \lambda]$ (Definition of Λ).
   $[a, b] \oplus \Lambda = [a + \lambda, b + \lambda]$ (Definition of $\oplus$).
   $[a, b] \oplus \Lambda = [a, b]$ (λ is the identity for $+$).
   A similar argument can be made for $\Lambda \oplus [a, b] = [a, b]$.

   (d) $\oplus$ is closed over S':
   Proof. Given $[a, b], [c, d] \in S'$, we must show that $[a, b] \oplus [c, d] \in S'$.
   $[a, b] \oplus [c, d] = [a + c, b + d]$ (Definition of $\oplus$).
   $a, b \in S$ and $a \leq b$ (Since $[a, b] \in S'$).
   $c, d \in S$ and $c \leq d$ (Since $[c, d] \in S'$).
   $a + c \in S$ and $b + d \in S$ ($+$ is closed over S).
   $a + c \leq b + d$ (Since $a \leq b$ and $c \leq d$).
   $[a + c, b + d] \in S'$ (Definition of S').
   Therefore, $[a, b] \oplus [c, d] \in S'$.

(e) $\oplus$ is idempotent:

Proof. Given $[a, b] \in S'$, we must show that $[a, b] \oplus [a, b] = [a, b]$.
$[a, b] \oplus [a, b] = [a + a, b + b]$ (Definition of $\oplus$).
$[a, b] \oplus [a, b] = [a, b]$ ($+$ is idempotent).

(f) $\oplus$ is monotonic:

Proof. Given $[a, b], [c, d] \in S'$ and $[a, b] \geq [c, d]$, we must show that $([a, b] \oplus [e, f]) \geq ([c, d] \oplus [e, f])$ *and* $([e, f] \oplus [a, b]) \geq ([e, f] \oplus [c, d])$.
Firstly, we define $\geq$ for intervals to mean that $[a, b] \geq [c, d] \Leftrightarrow a \geq c$ *and* $b \geq d$.
$[a, b] \oplus [e, f] = [a + e, b + f]$ (Definition of $\oplus$).
Similarly, $[c, d] \oplus [e, f] = [c + e, d + f]$.
$a \geq c$ and $b \geq d$ (Since $[a, b] \geq [c, d]$).
$a + e \geq c + e$ (Since $a \geq c$).
Similarly, $b + f \geq d + f$.
$[a + e, b + f] \geq [c + e, d + f]$ (Definition of $\geq$ for intervals).
Therefore, $([a, b] \oplus [e, f]) \geq ([c, d] \oplus [e, f])$.
Similar arguments can be made to prove that $([e, f] \oplus [a, b]) \geq ([e, f] \oplus [c, d])$.

2. Properties of $\odot$

(a) $\odot$ is associative:

Proof. Given $[a, b], [c, d] \in S'$, we must show that $([a, b] \odot [c, d]) \odot [e, f] = [a, b] \odot ([c, d] \odot [e, f])$.
$([a, b] \odot [c, d]) \odot [e, f] = [a \cdot c, b \cdot d] \odot [e, f]$ (Definition of $\odot$).
$([a, b] \odot [c, d]) \odot [e, f] = [(a \cdot c) \cdot e, (b \cdot d) \cdot f]$ (Definition of $\odot$).
$([a, b] \odot [c, d]) \odot [e, f] = [a \cdot (c \cdot e), b \cdot (d \cdot f)]$ (Associativity of $\cdot$).
$([a, b] \odot [c, d]) \odot [e, f] = [a, b] \odot ([c \cdot e, d \cdot f])$ (Definition of $\odot$).
$([a, b] \odot [c, d]) \odot [e, f] = [a, b] \odot ([c, d] \odot [e, f])$ (Definition of $\odot$).

(b) $\odot$ has unit element Ω:

Proof. Given $[a, b] \in S'$, we must show that $[a, b] \odot \Omega = \Omega \odot [a, b] = [a, b]$.
$[a, b] \odot \Omega = [a, b] \cdot [\omega, \omega]$ (Definition of Ω).
$[a, b] \odot \Omega = [a \cdot \omega, b \cdot \omega]$ (Definition of $\odot$).
$[a, b] \odot \Omega = [a, b]$ (ω is the unit element for $\cdot$).
A similar argument can be made for $\Omega \odot [a, b] = [a, b]$.

(c) $\odot$ is closed over S':

Proof. Given $[a, b], [c, d] \in S'$, we must show that $[a, b] \odot [c, d] \in S'$.
$[a, b] \odot [c, d] = [a \cdot c, b \cdot d]$ (Definition of $\odot$).
$a, b \in S$ and $a \leq b$ (Since $[a, b] \in S'$).
$c, d \in S$ and $c \leq d$ (Since $[c, d] \in S'$).
$a \cdot c \in S$ and $b \cdot d \in S$ ($\cdot$ is closed over S).
$a \cdot c \leq b \cdot d$ (Since $a \leq b$ and $c \leq d$).
$[a \cdot c, b \cdot d] \in S'$ (Definition of S').
Therefore, $[a, b] \odot [c, d] \in S'$.

(d) Λ is the annihilator for $\odot$:

Proof. Given $[a, b], [c, d] \in S'$, we must show that $[a, b] \odot \Lambda = \Lambda \odot [a, b] = \Lambda$.
$[a, b] \odot \Lambda = [a, b] \odot [\lambda, \lambda]$ (Definition of Λ).
$[a, b] \odot \Lambda = [a \cdot \lambda, b \cdot \lambda]$ (Definition of $\odot$).
$[a, b] \odot \Lambda = [\lambda, \lambda]$ (λ is the annihilator for $\cdot$).
$[a, b] \odot \Lambda = \Lambda$
A similar argument can be made for $\Lambda \odot [a, b] = \Lambda$.

(e) $\odot$ is monotonic:

Proof. Given $[a, b], [c, d] \in S'$ and $[a, b] \geq [c, d]$, we must show that $([a, b] \odot [e, f]) \geq$

$([c,d] \odot [e,f])$ *and* $([e,f] \odot [a,b]) \geq ([e,f] \odot [c,d])$.
Firstly, we define $\geq$ for intervals to mean that $[a,b] \geq [c,d] \Leftrightarrow a \geq c$ *and* $b \geq d$.
$[a,b] \odot [e,f] = [a \cdot e, b \cdot f]$ (Definition of $\odot$).
Similarly, $[c,d] \odot [e,f] = [c \cdot e, d \cdot f]$.
$a \geq c$ and $b \geq d$ (Since $[a,b] \geq [c,d]$).
$a \cdot e \geq c \cdot e$ (Since $a \geq c$).
Similarly, $b \cdot f \geq d \cdot f$.
$[a \cdot e, b \cdot f] \geq [c \cdot e, d \cdot f]$ (Definition of $\geq$ for intervals).
Therefore, $([a,b] \odot [e,f]) \geq ([c,d] \odot [e,f])$.
Similar arguments can be made to prove that $([e,f] \odot [a,b]) \geq ([e,f] \odot [c,d])$.

3. $\odot$ distributes over $\oplus$:
Proof. Given $[a,b], [c,d], [e,f] \in S'$, we must show that $[a,b] \odot ([c,d] \oplus [e,f]) = ([a,b] \odot [c,d]) \oplus ([a,b] \odot [e,f])$.
$[a,b] \odot ([c,d] \oplus [e,f]) = [a,b] \odot [c+e, d+f]$ (Definition of $\oplus$).
$[a,b] \odot ([c,d] \oplus [e,f]) = [a \cdot (c+e), b \cdot (d+f)]$ (Definition of $\odot$.)
$[a,b] \odot ([c,d] \oplus [e,f]) = [(a \cdot c) + (a \cdot e), (b \cdot d) + (b \cdot f)]$ ($\cdot$ distributes over $+$).
$[a,b] \odot ([c,d] \oplus [e,f]) = [a \cdot c, b \cdot d] \oplus [a \cdot e, b \cdot f]$ (Definition of $\oplus$).
$[a,b] \odot ([c,d] \oplus [e,f]) = ([a,b] \odot [c,d]) \oplus ([a,b] \odot [e,f])$. (Definition of $\odot$).

4. Properties of $\simeq$

(a) $\simeq$ is idempotent:
Proof. Given $[a,b] \in S'$, we must show that $[a,b] \simeq [a,b] = [a,b]$
$[a,b] \simeq [a,b] = [a+a, b]$ (Definition of $\simeq$).
$[a,b] \simeq [a,b] = [a,b]$ ($+$ is idempotent).

(b) $\simeq$ distributes over $\oplus$:
Proof. Given $[a,b], [c,d], [e,f] \in S'$, we must show that $([a,b] \oplus [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f])$.
Case 1. $b \geq e$ and $d \geq e$
$([a,b] \oplus [c,d]) \simeq [e,f] = [a+c, b+d] \simeq [e,f]$ (Definition of $\oplus$).
$b+d \geq e$ (Since $b \geq e$ and $d \geq e$).
$([a,b] \oplus [c,d]) \simeq [e,f] = [(a+c)+e, b+d]$ (Definition of $\simeq$).
$[a,b] \simeq [e,f] = [a+e, b]$ (Definition of $\simeq$).
$[c,d] \simeq [e,f] = [c+e, d]$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [(a+e)+(c+e), b+d]$ (Definition of $\oplus$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [(a+c)+e, b+d]$ ($+$ is commutative, associative and idempotent).
Therefore, $([a,b] \oplus [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f])$.
Case 2. $b \geq e > d$
$([a,b] \oplus [c,d]) \simeq [e,f] = [a+c, b+d] \simeq [e,f]$ (Definition of $\oplus$).
$b+d \geq e$ (Since $b \geq e > d$).
$([a,b] \oplus [c,d]) \simeq [e,f] = [(a+c)+e, b+d]$ (Definition of $\simeq$).
$([a,b] \oplus [c,d]) \simeq [e,f] = [(a+c)+e, b]$ (Since $b > d$).
$([a,b] \oplus [c,d]) \simeq [e,f] = [a+e, b]$ (Since $+$ is associative and $e > d \geq c$).
$[a,b] \simeq [e,f] = [a+e, b]$ (Definition of $\simeq$).
$[c,d] \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [(a+e)+\lambda, b+\lambda]$ (Definition of $\oplus$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [a+e, b]$ (λ is the identity for $+$).
Therefore, $([a,b] \oplus [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f])$.
Case 3. $d \geq e > b$
Repeat previous case, exchanging $[a,b]$ and $[c,d]$.

Case 4. $e > b$ and $e > d$
$([a,b] \oplus [c,d]) \simeq [e,f] = [a+c, b+d] \simeq [e,f]$ (Definition of $\oplus$).
$e > b + d$ (Since $e > b$ and $e > d$).
$([a,b] \oplus [c,d]) \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$[a,b] \simeq [e,f] = \Lambda$ (Definition of $\simeq$)
$[c,d] \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [\lambda + \lambda, \lambda + \lambda]$ (Definition of $\oplus$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = [\lambda, \lambda]$ (λ is the identity element for $+$).
$([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f]) = \Lambda$.
Therefore, $([a,b] \oplus [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \oplus ([c,d] \simeq [e,f])$.

(c) $\simeq$ distributes over $\odot$:
Proof. Given $[a,b], [c,d], [e,f] \in S'$, we must show that $([a,b] \odot [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f])$.
Case 1. $b \geq e$ and $d \geq e$
$([a,b] \odot [c,d]) \simeq [e,f] = [a \cdot c, b \cdot d] \simeq [e,f]$ (Definition of $\odot$).
$b \cdot d \geq e$ (Since $b \geq e$ and $d \geq e$).
$([a,b] \odot [c,d]) \simeq [e,f] = [(a \cdot c) + e, b \cdot d]$ (Definition of $\simeq$).
$([a,b] \odot [c,d]) \simeq [e,f] = [(a+e) \cdot (c+e), b \cdot d]$ ($+$ distributes over $\cdot$).
$[a,b] \simeq [e,f] = [a+e, b]$ (Definition of $\simeq$).
$[c,d] \simeq [e,f] = [c+e, d]$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = [(a+e) \cdot (c+e), b \cdot d]$ (Definition of $\odot$).
Therefore, $([a,b] \odot [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f])$.
Case 2. $b \geq e > d$
$([a,b] \odot [c,d]) \simeq [e,f] = [a \cdot c, b \cdot d] \simeq [e,f]$ (Definition of $\odot$).
$e > b \cdot d$ (Since $b \geq e > d$).
$([a,b] \odot [c,d]) \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$[a,b] \simeq [e,f] = [a+e, b]$ (Definition of $\simeq$).
$[c,d] \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = [(a+e) \cdot \lambda, b \cdot \lambda]$ (Definition of $\odot$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = [\lambda, \lambda]$ (λ is the annihilator for $\cdot$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = \Lambda$.
Therefore, $([a,b] \odot [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f])$.
Case 3. $d \geq e > b$
Repeat previous case, exchanging $[a,b]$ and $[c,d]$.
Case 4. $e > b$ and $e > d$
$([a,b] \odot [c,d]) \simeq [e,f] = [a \cdot c, b \cdot d] \simeq [e,f]$ (Definition of $\odot$).
$e > b \cdot d$ (Since $e > b$ and $e > d$).
$([a,b] \odot [c,d]) \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$[a,b] \simeq [e,f] = \Lambda$ (Definition of $\simeq$)
$[c,d] \simeq [e,f] = \Lambda$ (Definition of $\simeq$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = [\lambda \cdot \lambda, \lambda \cdot \lambda]$ (Definition of $\odot$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = [\lambda, \lambda]$ (λ is the annihilator for $\cdot$).
$([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f]) = \Lambda$.
Therefore, $([a,b] \odot [c,d]) \simeq [e,f] = ([a,b] \simeq [e,f]) \odot ([c,d] \simeq [e,f])$.

(d) $\simeq$ is monotonic in its first argument:
Proof. Given $[a,b], [c,d], [e,f] \in S'$ and $[a,b] \geq [c,d]$, we must show that $([a,b] \simeq [e,f]) \geq ([c,d] \simeq [e,f])$.
We again use the following definition of $\geq$ for intervals: $[a,b] \geq [c,d] \Leftrightarrow a \geq c$ *and* $b \geq d$.
Therefore, $a \geq c$ and $b \geq d$.
Case 1. $b \geq d \geq e$
$[a,b] \simeq [e,f] = [a+e, b]$ (Definition of $\simeq$).

$[c, d] \simeq [e, f] = [c + e, d]$ (Definition of $\simeq$).
$a + e \geq c + e$ (Since $a \geq c$).
$[a + e, b] \geq [c + e, d]$ (Definition of $\geq$ for intervals).
Therefore, $([a, b] \simeq [e, f]) \geq ([c, d] \simeq [e, f])$.
Case 2. $b \geq e > d$
$[a, b] \simeq [e, f] = [a + e, b]$ (Definition of $\simeq$).
$[c, d] \simeq [e, f] = \Lambda$ (Definition of $\simeq$).
$[a + e, b] \geq [\lambda, \lambda]$ (Definition of $\geq$ for intervals).
Therefore, $([a, b] \simeq [e, f]) \geq ([c, d] \simeq [e, f])$.
Case 3. $e > b \geq d$
$[a, b] \simeq [e, f] = \Lambda$ (Definition of $\simeq$)
$[c, d] \simeq [e, f] = \Lambda$ (Definition of $\simeq$).
$\Lambda \geq \Lambda$ (Definition of $\geq$ for intervals).
Therefore, $([a, b] \simeq [e, f]) \geq ([c, d] \simeq [e, f])$. Q.E.D.

References

[1] Bryant, Randal E. "An Algorithm for MOS Logic Simulation." *Lambda,* Vol. 1, No. 3, 1980, pp. 46-53.

[2] Bryant, Randal E. "A Switch-Level Model and Simulator for MOS Digital Systems." *IEEE Transactions on Computers,* February 1984, pp. 160-177.

[3] Byrd, Richard H., Gary D. Hachtel, Micheal R. Lightner and Michel H. Heydemann. "Switch-level Simulation: Models, Theory, and Algorithms." *Computer-Aided Design of VLSI Circuits and Systems.* Vol. I. *Advances in Computer-Aided Engineering Design.* Ed. A. Sangiovanni-Vincentelli. Greenwich, CT: JAI Press, 1985. pp. 93-148.

[4] Gecsei, Jan and Eduard Cerny. "Self-adjusting Networks for VLSI Simulation." *IEEE Transactions on Computers.* September 1987. pp. 1114-1120.

[5] Ramachandran, Vijaya. "An Improved Switch-Level Simulator for MOS Circuits." *20th IEEE Design Automation Conference.* 1983, Paper 21.3, pp. 293-299.

[6] Sundblad, Rolf and Christer Svensson. "Fully Dynamic Simulation of CMOS Circuits." *IEEE Transactions on Computer-Aided Design.* March 1987. pp. 282-289.