September 1991

UILU-ENG-91-2248
DAC-32

Analog and Digital Circuits

AN ALGEBRAIC APPROACH TO SWITCH-LEVEL SIMULATION

Ibrahim N. Hajj

Coordinated Science Laboratory
College of Engineering
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-91-2248 (DAC-32)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Semiconductor Research Corp.
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) SRC PO Box 12053 Research Triangle Park, NC 27709
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Semiconductor Research Corp. 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER SRC 88-DP-109
8c. ADDRESS (City, State, and ZIP Code) SRC PO 12053 Research Triangle Park, NC 27709	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) An Algebraic Approach to Switch-Level Simulation	
12. PERSONAL AUTHOR(S) Hajj, Ibrahim N.	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 91 Sep 15. PAGE COUNT 32
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Switch-level simulation; MOS VLSI circuits; switch-level matrix algebra; path algebra

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper presents a new methodology for computing the steady-state solution of switch-level networks. The method is based on a matrix algebra similar in many respects to circuit nodal analysis in that the formulation steps are similar to the nodal equation formulation steps and the network matrix has the same dimension and structure as the nodal admittance matrix, except that logic operations (min and max operations) are employed in formulating and solving the network equations. Solution algorithms are then developed using the new algebra. The approach has been implemented as a part of a mixed-mode simulator which uses partitioning and sparse matrix solution techniques in analyzing the circuit. In its switch-level mode, the simulator can perform logic and concurrent fault simulation using realistic fault models, and has been found to be competitive with existing switch-level simulators.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

An Algebraic Approach to Switch-Level Simulation†

Ibrahim N. Hajj

Coordinated Science Laboratory and
Department of Electrical and Computer Engineering
University of Illinois
1101 West Springfield Avenue
Urbana, Illinois 61801

Abstract

This paper presents a new methodology for computing the steady-state solution of switch-level networks. The method is based on a matrix algebra similar in many respects to circuit nodal analysis in that the formulation steps are similar to the nodal equation formulation steps and the network matrix has the same dimension and structure as the nodal admittance matrix, except that logic operations (min and max operations) are employed in formulating and solving the network equations. Solution algorithms are then developed using the new algebra. The approach has been implemented as a part of a mixed-mode simulator which uses partitioning and sparse matrix solution techniques in analyzing the circuit. In its switch-level mode, the simulator can perform logic and concurrent fault simulation using realistic fault models, and has been found to be competitive with existing switch-level simulators.

†This work was supported by the Semiconductor Research Corporation.

I. Introduction

Switch-level simulation is now a well established method for verifying the steady-state logic behavior of digital MOS circuits. The technique has attracted a lot of attention, and since its introduction 10 years ago [1], a large number of papers have been published on the subject [2-4], to mention just a few. The technique has also been used for fault simulation, test generation, and symbolic analysis (because of limitation of space and since this is not a survey paper, we will keep the list of references to a minimum). Almost all, if not all, switch-level solution techniques proposed so far have relied on path tracing, path strengths and propagation of signals along paths. Even when matrix techniques and elimination algorithms were developed, the concepts were based on using matrix algebra to determine path strengths in labeled graphs [5], [6].

In this paper, we present a matrix algebra for switch-level simulation similar in many respects to circuit nodal analysis, in that the matrix formulation has the same dimension and structure as the nodal admittance matrix of the underlying circuit model, except that logical operations are used in solving the equations, rather than real arithmetic. In contrast, the method proposed in [6] requires setting up three systems of Boolean equations, and the number of equations depends on the number of signal strengths in the switch-level network model.

In some respects, the proposed method can be considered as a matrix implementation of the switch-level theory presented in [4]. Even though the authors in [4] explain the relation between switch-level theory and circuit theory and the connection between the path-based algorithms and Gaussian elimination, no specific matrix-based algorithm is included. One motivation of this work is to related switch-level solution algorithms, from the implementation

point of view, to circuit-level nodal analysis, and to incorporate the algorithms into a mixed-mode simulator by sharing as many common subroutines as possible with other simulation nodes, such as circuit and timing simulation algorithms. Another motivation is to have the flexibility of performing fault simulation using realistic fault models of various strengths, including bridging faults. Bridging faults can be more efficiently injected into a matrix-based formulation than a path-based formulation, especially if symbolic representation is used.

In the next two sections the basic operations of the new algebra are explained. The formulation of the switch-level network model in matrix form and its solution are considered in Section 4. Implementation issues and examples are given in Section 5.

II. Basic Operations

We use the "order-of-magnitude" switch-level circuit model proposed in [4], which is a generalization of the switch-level models of Bryant [2] and of Hayes [3]. Each node is considered to have any one of three values: 0 (low), 1 (high), X (unknown, or intermediate between and *including* 0 and 1). Each transistor is allowed to have a lower (guaranteed) strength and an upper (potential) strength, determined by the range of its conductance values. The relative strength values of the branches are represented by two integers (a_1,a_2), where a_1 represents the lower strength and a_2 the upper one. For example, an edge consisting of a single-NMOS transistor has strength (0,0) if its gate voltage is 0, (a_2,a_2) if its gate voltage is 1, and $(0,a_2)$ if its gate voltage is X. The initial condition at a storage node is modeled by adding a branch connecting it to a source whose value is equal to the node's initial value. The node branch is considered to be always fully on with an upper strength chosen to reflect the capacitance at the node relative to the capacitances at all other storage nodes in the circuit.

The maximum strength of the node branches is chosen to be weaker than the lowest upper strength index assigned to any transistor branch in the circuit; but this is not necessary, a large capacitor could be given a strength number higher than a transistor with low conductance if the capacitor is expected to hold most of its charge during a clock cycle.

Let the strengths of the branches, both node and transistor branches, in the switch-level network model be ordered in ascending order as follows:

$$0 < 1 < 2 < \cdots < s \tag{1}$$

where s is the number of discrete upper strengths assigned to the branches in the switch-level network model. In general, the strength of a branch is assigned two integers (a_1,a_2), where $a_1 \leq a_2$, which represent the interval of its conductance values.

We now define a set P to consist of all possible intervals (a_1,a_2), $a_1 \leq a_2$, defined on the set of integers in (1). A component of P will also be denoted by a lower case letter, e.g., x. We also define the join (addition, OR) and the multiplication (AND) operations on the elements of P, respectively, as follows, where $x \equiv (a_1,a_2)$, $y \equiv (a_3,a_4)$:

$$x + y = (a_1,a_2) + (a_3,a_4) = (\max(a_1,a_3), \max(a_2,a_4)) \tag{2}$$

$$x \cdot y = (a_1,a_2) \cdot (a_3,a_4) = (\min(a_1,a_3), \min(a_2,a_4)) \tag{3}$$

when there is no confusion, the product $x \cdot y$ will also be written as xy.

Claim 1. The set P given in (1) together with the + and · operations defined in (2) and (3), and the zero element being $\phi \equiv (0,0)$, and the unit element being $e \equiv (s,s)$, define a path algebra [7].

Proof:

(A) The + operation is idempotent, commutative, and associative:

(A.1) $x + x = (a_1,a_2) + (a_1,a_2) = (\max(a_1,a_1), \max(a_2,a_2))$
$= (a_1,a_2) = x \quad \forall\, x \in P$

(A.2) $x + y = (a_1,a_2) + (a_3,a_4) = (\max(a_1,a_3), \max(a_2,a_4))$
$= (\max(a_3,a_1), \max(a_4,a_2))$
$= (a_3,a_4) + (a_1,a_2), \quad \forall \in P$
$= y + x \quad \forall\, x, y \in P$

(A.3) $(x+y) + z = ((a_1,a_2) + (a_3,a_4)) + (a_5,a_6) = (\max(a_1,a_3), \max(a_2,a_4)) + (a_5,a_6)$
$= (\max(a_1,a_3,a_5), \max(a_2,a_4,a_6))$
$= (a_1,a_2) + (\max(a_3,a_5), \max(a_4,a_6))$
$= (a_1,a_2) + ((a_3,a_4) + (a_5,a_6)) \quad \forall\, x, y, z \in P$
$= x + (y+z).$

(B) The · operation is associative, and distributive over + :

(B.1) $(x\,y)z = ((a_1,a_2) \cdot (a_3,a_4)) \cdot (a_5,a_6) = (\min(a_1,a_3), \min(a_2,a_4)) \cdot (a_5,a_6)$
$= (\min(a_1,a_3,a_5), \min(a_2,a_4,a_6))$
$= (a_1,a_2) \cdot ((a_3,a_4) \cdot (a_5,a_6))$
$= x(yz) \quad \forall\, x, y, z \in P$

(B.2) $x(y+z) = (a_1,a_2) \cdot ((a_3,a_4) + (a_5,a_6))$
$= (a_1,a_2) \cdot (\max(a_3,a_5), \max(a_4,a_5))$ (4)
$= (\min(a_1,\max(a_3,a_5), \min(a_2,\max(a_3,a_6))$

and

$xy + xz = ((a_1,a_2) \cdot (a_3,a_4)) + ((a_1,a_2) \cdot (a_5,a_6))$
$= (\min(a_1,a_3), \min(a_2,a_4)) + (\min(a_1,a_5), \min(a_2,a_6))$ (5)
$= (\max(\min(a_1,a_3), \min(a_1,a_5)), \max(\min(a_2,a_4), \min(a_2,a_6))).$

But, it can be shown that if a, b, c are three real numbers, then

$$\max(\min(a,b), \min(a,c)) = \min(a,\max(b,c)). \tag{6}$$

It follows that (4) equals (5), and therefore,

$$x(y+z) = xy + xz \qquad \forall\, x, y, z \in P$$

Similarly, it can be shown that

(B.3) $$(y+z)x = yx + zx \qquad \forall\, x, y, z \in P$$

(C) The zero element $\phi = (0,0)$ and the unit element $e = (s,s)$ have the following properties:

(C.1) $\phi + x = (0,0) + (a_1,a_2) = (\max(0,a_1), \max(0,a_2)) = (a_1,a_2) = x$

(C.2) $\phi \cdot x = (0,0) \cdot (a_1,a_2) = (a_1,a_2) \cdot (0,0) = (\min(0,a_1), \min(0,a_2)) = (0,0) = \phi$

(C.3) $e + x = (s,s) + (a_1,a_2) = (\max(s,a_1), \max(s,a_2)) = (s,s) = e$

(C.4) $e \cdot x = (s,s) \cdot (a_1,a_2) = (\min(s,a_1), \min(s,a_2)) = (a_1,a_2) = x$

$$\forall\, x \in P$$

This completes the proof.

Remark. The joint operation $(a_1,a_2) + (a_3,a_4)$ can be interpreted as the parallel combination of two branches to form a new equivalent branch of conductance strength equal to $(\max(a_1,a_3), \max(a_2,a_4))$, while the product $(a_1,a_2) \cdot (a_3,a_4)$ gives the series combination of two branches. The need for a minimum value of the strength of a branch other than 0 as suggested in [4] can be explained by considering the parallel combination of two branches, one with strength $(0,a_2)$ and the other (a_3,a_3), where $0 < a_3 < a_2$, as shown in Fig. (1a). It can be easily seen that the strength of the equivalent branch will be (a_3,a_2). The series combination of $(0,a_2)$ and (a_3,a_3) is $(0,a_3)$, as shown in Fig. (1b).

It follows from Claim 1 that the elements of P obey the elementary properties of path algebras. These properties are given in [7] and are repeated here for completeness. For example, we can define an ordering $\leq$ of P by the following rule:

$$(a_1,a_2) \leq (a_3,a_4), \text{ if and only if } a_2 \leq a_3. \tag{7}$$

In particular,

$$\phi \leq x \quad \forall\, x \in P. \tag{8}$$

In addition, every element $x \in P$ is sub-unitary; i.e., $x \leq e, \forall\, x \in P$. The powers of an element $x \in P$ are defined by:

$$x^0 = e,\ x^k = x^{k-1}\, x \quad (k = 1,2,...)$$

with $x^2 = x, \forall\, x \in P$. Therefore, every element $x \in P$ is indempotent and stable of index 0, and its *closure* is

$$x^* = e, \quad \forall\, x \in P. \tag{9}$$

Let $M_n(P)$ be the set of all nxn matrices whose entries belong to P. Then for any matrices, $\mathbf{Y} = [y_{ij}]$ and $\mathbf{Z} = [z_{ij}]$ in P,

$$\mathbf{Y} + \mathbf{Z} = [y_{ij} + z_{ij}] \tag{10}$$

$$\mathbf{Y} \cdot \mathbf{Z} = [\sum_{k=1}^{n} y_{ik} \cdot z_{kj}] \tag{11}$$

In addition, $M_n(P)$ is itself a path algebra, and the properties of the addition and multiplication operations given in Claim 1 apply also to the matrix case. In this case, the *zero matrix* is an nxn matrix $\mathbf{\Phi}$ whose entries are all ϕ, and the unit matrix $\mathbf{E}$ is an nxn matrix with unit elements *e* on the diagonal and ϕ entries elsewhere, and

$$\Phi + Y = Y, \Phi \cdot Y = \Phi = Y \cdot \Phi$$

$$E \cdot Y = Y = Y \cdot E \quad \forall Y \varepsilon M_n(P).$$

III. Node Strength and State and Solution of Simple Switch-Level Networks

In the above section, we described the operations that could be used to perform series-parallel combinations of branches in a switch-level network model. Before we explain the solution of general switch-level network equations, we consider a simple network which consists of a single branch of strength (a_1,a_2) connecting an input node I to a storage node v, as shown in Fig. 2. The input node could represent the initial condition at the storage node.

Assertion 1. The value of the signal at node v is given by $(a_1,a_2)_o I$, where I could be 0, 1 or X.

Assertion 1 has often been used to describe the value of a signal at a node, where (a_1,a_2) is termed the "strength" of the signal at v, and I its state. It can also be considered as "Norton" equivalent circuit at v, where $(a_1,a_2)_o I$ represents the equivalent current value. Note that the symbol "$_o$" is introduced because (a_1,a_2) and I are defined over different fields. In contrast, in circuit analysis, the conductance (or the admittance) and the voltage values are both real (or complex), and they can be multiplied to obtain an equivalent current as another real (or complex) member.

Consider now a storage node v connected to two input nodes I_1 and I_2 by two branches (a_1,a_2) and (a_3,a_4), respectively, as shown in Fig. 3.

Assertion 2. The signal value at node v shown in Fig. 3 is given by

$$\begin{aligned}(a_1,a_2) \circ I_1 + (a_3,a_4) \circ I_2 &= ((a_1,a_2) + (a_3,a_4)) \circ \hat{I} \\ &= (\max(a_1,a_3), \max(a_2,a_4)) \circ \hat{I}\end{aligned} \tag{12}$$

where $\hat{I} = \{ I_1$ if $I_1 = I_2$ or if $a_1 > a_4$; I_2 if $a_3 > a_2$; X otherwise$\}$.

The justification of the addition operation in Assertion 2 is based on the assumption that the signal value at a storage node depends on the states of those input nodes connected to it by edges of maximum strength. This is also the basis of the lattice operation used in [2] and [3]. In fact, if $a_1 = a_2$ and $a_3 = a_4$, the addition operation given in (12) becomes exactly the # operator of Hayes [3]. Equation (12) could also be interpreted as the parallel combination of two Norton equivalent circuits.

Let us consider now a storage node v_1 connected to an input node I_1 by a series connection of two branches of strengths (a_1,a_2), (a_3,a_4), as shown in Fig. 4.

Assertion 3. The signal value at v_1, shown in Fig. 4, is given by

$$\begin{aligned}(a_3,a_4) \cdot ((a_1,a_2) \circ I &= ((a_3,a_4) \cdot (a_1,a_2)) \circ I \\ &= (\min(a_1,a_3), \min(a_2,a_4)) \circ I.\end{aligned} \tag{13}$$

Equation (13) can also be interpreted as the elimination of the intermediate node v_2.

We will now consider the properties of the addition and the multiplication operations defined in (12) and (13).

Properties

(D.1) The + operation defined in (12) is indempotent and commutative over $\circ$.

Proof: Obvious.

(D.2) Given $(a_1,a_2)\circ I_1 + (a_3,a_4)\circ I_2 + (a_5,a_6)\circ I_3$; if $I_1 \neq I_2 \neq I_3$, then the + operation defined in (12) is associative over $\circ$. In general, however, the + operation defined in (12) is **not** always associative over $\circ$. In particular, if $I_1 = I_2 \neq I_3$, and if $a_3 \leq a_6 < a_1 \leq a_4$, then the + operation is not associative over $\circ$.

Proof: Suppose that the condition is true; i.e., $I_1 \neq I_2 \neq I_3$, then

$$((a_1,a_2) \circ I_1 + (a_3,a_4) \circ I_2) + (a_5,a_6) \circ I_3 = (\max(a_1,a_3), \max(a_2,a_4)) \circ \hat{I}_1 + (a_5,a_6) \circ I_3$$
$$= (\max(a_1,a_3,a_5), \max(a_2,a_4,a_6)) \circ \hat{\hat{I}}_1$$

where $\hat{I}_1 = \{ I_1$ if $a_1 > a_4$; I_2 if $a_3 > a_2$; X otherwise $\}$, and $\{\hat{\hat{I}}_1 = I_1$ if $a_1 > a_4$ and $a_1 > a_6$; I_2 if $a_3 > a_2$ and $a_3 > a_6$; I_3 if $a_5 > a_2$ and $a_5 > a_4$; X otherwise$\}$.

Similarly,

$$(a_1,a_2) \circ I_1 + ((a_3,a_4) \circ I_2 + (a_5,a_6) \circ I_3) = (a_1,a_2) \circ I_1 + (\max(a_3,a_5), \max(a_4,a_6)) \circ \hat{I}_2$$
$$= (\max(a_1,a_3,a_5), \max(a_2,a_4,a_6)) \circ \hat{\hat{I}}_2$$

where $\hat{I}_2 = \{ I_2$ if $a_3 > a_6$; I_3 if $a_5 > a_4$; X otherwise $\}$, and $\hat{\hat{I}}_2 = \{ I_1$ if $a_1 > a_4$ and $a_1 > a_6$; I_2 if $a_3 > a_2$ and $a_3 > a_6$; I_3 if $a_5 > a_2$ and $a_5 > a_4$; X otherwise $\}$. Therefore, $\hat{\hat{I}}_2 = \hat{\hat{I}}_1$. Now suppose $I_1 = I_2 \neq I_3$, then

$$((a_1,a_2) \circ I_1 + (a_3,a_4) \circ I_1) + (a_5,a_6) \circ I_3 = (\max(a_1,a_3), \max(a_2,a_4)) \circ I_1 + (a_5,a_6) \circ I_3$$
$$= (\max(a_1,a_3,a_5), \max(a_2,a_4,a_6)) \circ \hat{I}_1$$

where $\hat{I}_1 = \{ I_1$ if $\max(a_1,a_3) > a_6$; I_3 if $a_5 > \max(a_2,a_4)$; X otherwise $\}$, which is the correct result.

On the other hand,

$$(a_1,a_2) \circ I_1 + ((a_3,a_4) \circ I_1 + (a_5,a_6) \circ I_3) = (a_1,a_2) \circ I_1 + (\max(a_3,a_5), \max(a_4,a_6)) \circ \hat{I}_3$$
$$= (\max(a_1,a_3,a_5), \max(a_2,a_4,a_6)) \circ \hat{\hat{I}}_3$$

where $\hat{I}_3 = \{ I_1$ if $a_3 > a_6$; I_3 if $a_5 > a_4$; X otherwise $\}$, and $\hat{\hat{I}}_3 = \{ I_1$ if $a_3 > a_6$ or if $a_1 > \max(a_4,a_6)$; I_3 if $a_5 > a_4$ and $a_5 > a_2$; X otherwise $\}$. Now if $a_3 \leq a_6 < a_1 \leq a_4$, then $\hat{I}_1 = I_1$, but $\hat{\hat{I}}_3 = X$. Therefore, $\hat{\hat{I}}_3$ is not necessarily equal to $\hat{I}_1$. *Note, that the final strength of the signal is the same in both cases, but the state is different.* This fact leads to the following rule.

Rule 1. When performing the + operation over $\circ$, perform the operation on signals with the same state first.

A network example illustrating Rule 1 is given below in Section V. Note that Rule 1 need not be applied if the signal strengths are definite; i.e., if $a_1 = a_2$ and $a_3 = a_4$. We will now consider the properties of the multiplication operation defined in (13).

(D.3) The $\cdot$ operation defined in (13) is associative, but **not** necessarily distributive over + and $\circ$. In particular, if $a_3 > a_6$, and $a_1 \leq a_5$ or $a_2 \leq a_6$ (or if $a_5 > a_4$, and $a_1 \leq a_3$ or $a_2 \leq a_4$), then the $\cdot$ operation is not distribute over + and $\circ$.

(a) (Associative)

$$\begin{aligned}((a_1,a_2)\cdot(a_3,a_4))\cdot(a_5,a_6)\circ I_1 &= (\min(a_1,a_3),\min(a_2,a_4))\cdot(a_5,a_6)\circ I_1\\ &= (\min(a_1,a_3,a_5),\min(a_2,a_4,a_6))\circ I_1\\ &= (a_1,a_2)\cdot((a_3,a_4)\cdot(a_5,a_6))\circ I_1 .\end{aligned} \tag{14}$$

(b) (Distributive)

$$\begin{aligned}(a_1,a_2)\cdot((a_3,a_4)\circ I_1+(a_5,a_6)\circ I_2) &= (a_1,a_2)\cdot((\max(a_3,a_5),\max(a_4,a_6))\circ \hat{I}_1)\\ &= (\min(a_1,\max(a_3,a_5)),\min(a_2,\max(a_3,a_5))\circ \hat{I}_1\end{aligned} \tag{15}$$

where $\hat{I}_1 = \{\ I_1$ if $I_1 = I_2$, or if $a_3 > a_6$; I_2 if $a_5 > a_4$; X otherwise $\}$. On the other hand,

$$\begin{aligned}(a_1,a_2)\cdot(a_3,a_4)\circ I_1+(a_1,a_2)\cdot(a_5,a_6)\circ I_2 &= (\min(a_1,a_3),\min(a_2,a_4))\circ I_1+(\min(a_1,a_5),\min(a_2,a_6))\circ I_2\\ &= (\max(\min(a_1,a_3),\min(a_1,a_5)),\max(\min(a_2,a_4),\min(a_2,a_6))\circ \hat{\hat{I}}_1\\ &= (\min(a_1,\max(a_3,a_5)),\min(a_2,\max(a_4,a_6)))\circ \hat{\hat{I}}_1\end{aligned} \tag{16}$$

where $\hat{\hat{I}}_1 = \{\ I_1$ if $I_1 = I_2$, or if $\min(a_1,a_3) > \min(a_2,a_6)$; I_2 if $\min(a_1,a_5) > \min(a_2,a_4)$; X otherwise$\}$. Note, that we have applied relation (6) to shown that the signal strengths in (16) and (15) are the same. Now, if $I_1 \neq I_2$, $a_3 > a_6$, and $a_1 \leq a_5$ or $a_2 \leq a_6$, then $\hat{I}_1 = I_1$, while $\hat{\hat{I}}_1 = X$; or if $I_1 \neq I_2$, $a_5 > a_4$, and $a_1 \leq a_3$ or $a_2 \leq a_4$, then $\hat{I}_1 = I_2$, while $\hat{\hat{I}}_1 = X$.

Rule 2. When performing the operation $(a_1,a_2)\ ((a_3,a_4)\circ I_1 + (a_5,a_6)\circ I_2)$, *if* $I_1 \neq I_2$, and $a_3 > a_6$ or $a_5 > a_4$, *perform the + operation inside the brackets first.*

In other words, if one signal "dominates" the other inside the brackets, perform the + operation inside the brackets first. Rule 2 in fact corresponds to the blocking operation devised in [2]. Note also that in Rule 2, (a_1,a_2) outside the bracket need not be checked; this is due to the fact that if $a_3 \leq a_6$ and $a_5 \leq a_4$, the operation becomes distributive, independent of (a_1,a_2).

As will be seen in the next section, in particular (23) and (24), there will be cases when it is necessary to perform the following operation:

$$(a_1,a_2)((a_3,a_4) \circ 1 + (a_5,a_6) \circ 0 + (a_7,a_8) \circ X) + (a_9,a_{10}) \circ 1 + (a_{11},a_{12}) \circ 0 + (a_{13},a_{14}) \cdot X. \tag{17}$$

Rule 2 calls for performing the + operation inside the brackets first on any two signals in which one signal dominates the other, while Rule 1 calls for the addition of signals of the same state by opening the brackets first. These two rules seem to be in conflict. However, it can be shown that the conditions that necessitate the application of Rule 1 (as stated in property D.2) are not in conflict with the conditions that call for the application of Rule 2 (as stated in property D.3). Thus, Rule 2 has precedence over Rule 1; but when Rule 2 is not applicable, the brackets *should* be opened first in case Rule 1 conditions apply. This is formulated in the following Rule.

Rule 3. *When performing* $(a_1,a_2)((a_3,a_4) \circ 1 + (a_5,a_6) \circ 0 + (a_7,a_8) \circ X)$, *perform the + operation first inside the brackets on any signals in which one dominates the others; otherwise, open the brackets first.*

An example illustrating Rule 3 is given below in Section V. We are now in a position to address the problem of solving for the node voltages in a general switch-level network. The basic operations needed are Eqs. (2), (3), (12), and (13), together with Rules 1 and 3.

IV. Network Solution

We assume the circuit under consideration to be partitioned into an interconnection of channel-connected subcircuits. We consider the computation of the steady-state solution of one such subcircuit, given the states of the gate nodes of each transistor in the subcircuit as

well as the initial conditions at the subcircuit storage nodes. A switch-level network model can then be constructed based on the states of the gates of the transistors, transistor dc characteristics, initial conditions at the storage nodes, and the states of the input signals, including power supply V_{DD} and ground. We use a switch-level nodal formulation approach, similar in many respects to the standard circuit nodal formulation approach, where conductance is replaced by branch strength intervals and conductance addition is replaced by the maximum operation defined in (3).

The switch-level nodal equations can be written in the form:

$$\mathbf{v} = \mathbf{A}\mathbf{v} + \mathbf{B} \circ \mathbf{v}_s \tag{18}$$

where $\mathbf{v}$ is n-vector of voltage values at the storage nodes and $\mathbf{v}_s = [1\ 0\ X]^t$, the vector of input states. Equation (18) can be derived from the nodal circuit equations by using an "order-of-magnitude" model and applying a limiting process as is done in [4], or can be constructed from a labeled graph model [7] of the switch-level network. The entries of **A** and **B** are of the form (a_i,a_j), with A being symmetric of size nxn with ϕ entries on the diagonal of size, and **B** is an nx3 matrix. Note that **A** can also be considered as an andjacency matrix of an *absorptive* graph, and thus is itself absorptive and stable, with a stability index not greater than (n-1), and its closure is

$$\mathbf{A}^* = \mathbf{E} + \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{n-1}. \tag{19}$$

A *feasible* solution of (18) is a vector $\mathbf{v}$ whose elements are of the form $(a_i,a_j) \circ I$ that satisfies the equation. The *least* solution of (18) can be found by

$$\mathbf{v} = \mathbf{A}^* (\mathbf{B} \circ \mathbf{v}_s). \tag{20}$$

The proof can be obtained by direct substitution of (20) into (18). Note that the vector $\mathbf{B}\circ\mathbf{v}_s$ can be written as

$$\mathbf{b} \equiv \mathbf{B} \circ \mathbf{v}_s = \begin{bmatrix} b_{11}\circ 1 + b_{10}\circ 0 + b_{1x}\circ X \\ b_{21}\circ 1 + b_{20}\circ 0 + b_{2x}\circ X \\ \vdots \\ b_{n1}\circ 1 + b_{n0}\circ 0 + b_{nx}\circ X \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} \circ 1 + \begin{bmatrix} b_{10} \\ b_{20} \\ \vdots \\ b_{n0} \end{bmatrix} \circ 0 + \begin{bmatrix} b_{1x} \\ b_{2x} \\ \vdots \\ b_{nx} \end{bmatrix} \circ X. \qquad (21)$$

The closure $\mathbf{A}^*$ is in some sense equivalent to the inverse of a nonsingular matrix in ordinary linear algebra. In performing the product $\mathbf{A}^*$ in (20), Rules 1 and 3 should be followed; otherwise, some states in the solution vector v would be X when they should be 0 or 1. Note that state X is valid even when the actual state is 0 or 1 since X includes 0 and 1. We say that a least solution is *minimal* if it has a minimum number of X states. We claim that following Rules 1 and 3 in performing the product in (20) guarantees a minimal least solution.

When the dimension n of **A** is relatively large, **A** is sparse since it has the same structure as the nodal admittance matrix; but $\mathbf{A}^*$ is in general dense, and the Gauss elimination method provides a more efficient way to solve (18). The proof that Gauss elimination finds the least solution of (18) is given in Appendix A. As shown in Appendix A, at the end of the forward Gauss elimination process all the elements on and below the diagonal of **A** become ϕ. Since **A** is symmetric, only the upper off-diagonal elements of **A**, a_{ij}, i = 1,2,...,n-1, j = i+1,...,n, need to be stored and operated on. Thus, during the elimination process, the upper off-diagonal elements of **A** and the **b** vector are updated by the following algorithm.

{*Forward elimination*}
for k ← 1 to n-1 do
begin
    for i ← k+1 to n-1 do
    begin
        for j ← i+1 to n do
        begin
            $a_{ij} \leftarrow a_{ij} + a_{ki}\, a_{kj}$ (22)
        end
        $b_i \leftarrow b_i + a_{ki}\, b_k$ (23a)
    end
    $b_n \leftarrow b_n + a_{kn}\, b_k$ (23b)
end

The solution is then obtained by backward substitution:

{*Backward substitution*}
for i ← n to 2 do
begin
    $v_i \leftarrow b_i$
    for j = 1 to i-1 do
    begin
        $b_j \leftarrow b_j + a_{ji}\, b_i$ (24)
    end
end
$v_1 = b_1$

In performing the operations in (23) and (24), Rules 1 and 3 should be followed in order to obtain a minimal least solution. Since **A** has the same structure as the nodal admittance matrix, sparse matrix techniques can be employed to minimize computation and storage, similar to what is done in solving linear algebraic equations, especially when n is large.

Remarks

1. Note that the diagonal elements of **A** need not be operated on during the elimination process. This means that no operation is performed when a single off-diagonal element exists in the k-th row of the reduced matrix at the k-th step in the elimination process;

i.e., the operation in (22) can be skipped. Thus, the total number of logical operations required to solve (18) by Gauss elimination is less than the number of floating-point operations needed to solve a linear symmetric system of the same structure. More specifically, the number of logic operations required in the matrix factorization step, Eq. (22), is $\frac{n}{3}(n^2-3n+2)$, where each max or min operation is counted as one logic operation. In practice, for series-parallel networks with sparse matrix techniques used, the number of operations is of order n, as pointed out in [6]. The number of logic operations required in the forward and backward substitution steps, Eqs. (23) and (24) combined, is $6n^2 - 4n$ in the worst case because of the need for applying Rules 1 and 3.

2. Note that Rule 1 needs to be applied only when the transistor strengths in a subcircuit contain intervals rather than definite strengths. We can thus conclude that if all the signal states arriving at the gates of the transistors of a subcircuit are all definitely 0 or 1, with no X states, then the + operation in (12) becomes associative over $_o$, and Rule 1 need not be strictly followed. In addition, the + operation inside the brackets in Rule 3 should be performed first before opening the brackets. In this case, the three components in each row of the **b** vector in (21) can be reduced to a single component at the outset, and the logic operations required in carrying out (23) and (24) become $2(n^2 - n)$.

3. If the **b** vector in (18) is changed, while **A** remains unchanged, then the operation in (22) need not be repeated, and only (23) and (24) are repeated for each new **b** vector.

4. Gauss elimination can also be used to compute the closure $\mathbf{A}^*$ by solving the matrix equation

$$\mathbf{A}\mathbf{A}^{*} = \mathbf{E} \tag{25}$$

instead of using (19).

V. Examples

In this section, we explain the solution of typical small examples to illustrate the concepts and the techniques described above.

Example 1: This example illustrates the need of applying Rule 1. Figure 5(a) shows a simple NMOS NOR gate; its switch-level network model is shown in Fig. 5(b). There is only one storage node in this circuit, v_{out}. The numbers inside brackets indicate the maximum strength of each transistor or storage node. The switch-level equation at v_{out} is:

$$v_{out} = (2,2)\circ 1 + (0,3)\circ 0 + (3,3)\circ 0 + (1,1)\circ X.$$

If the first two terms are added first, contrary to Rule 1, the result would be $v_{out} = (3,3)\circ X$. On the other hand, Rule 1 calls for adding terms 2 and 3 first; and the result in this case is $v_{out} = (3,3)\circ 0$, which is the minimal least solution.

Example 2: This example illustrates the application of Rules 1 and 3. Consider the NMOS circuit with pass transistors shown in Fig. 6(a), with its switch-level network model shown in Fig. 6(b). The initial conditions at the storage nodes are $v_1 = 0$, $v_2 = 0$, $v_3 = 1$. Since a signal with an X state exists at the gate of one of the transistors, the entries of the **b** vector should not be collapsed to one entry in each row at the outset so that Rules 1 and 3 can be applied during the elimination process. The network equations are

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi & (3,3) & \phi \\ (3,3) & \phi & (0,3) \\ \phi & (0,3) & \phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} (3,3) \\ \phi \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (1,1) \\ \phi \end{bmatrix} \circ 0$$

where we have shown the entire network matrix for clarity. As mentioned in the previous section, only the upper or lower part of the matrix need to be stored. Since at every step in the elimination process at most one non-zero off-diagonal element exists in the pivot row, the network matrix remains unchanged. The forward elimination process yields

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi \\ \phi \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (3,3) \\ (0,3) \end{bmatrix} \circ 0$$

where Rule 3 has been applied in which the (4,4) entry dominates the other entries in row 1. Backward substitution yields:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (0,2) \\ (0,2) \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (3,3) \\ (0,3) \end{bmatrix} \circ 0 = \begin{bmatrix} (4,4) \circ 0 \\ (3,3) \circ 0 \\ (2,3) \circ X \end{bmatrix}$$

Example 3: Consider the XNOR circuit shown in Fig. 7. The weights of the transistors are indicated in brackets on the figure; and all the capacitances at the storage nodes are given a weight of one. This circuit is channel-connected with internal feedback. Suppose A = B = 1. We will use relaxation techniques to solve for the output. Let the initial states at the node be $v_1^\circ = v_2^\circ = v_3^\circ = X$. The equations become

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi & \phi & (0,3) \\ \phi & \phi & (0,3) \\ (0,3) & (0,3) & \phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} (2,2) \\ (2,2) \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (4,4) \\ \phi \end{bmatrix} \circ 0 + \begin{bmatrix} (1,1) \\ (1,1) \\ (1,1) \end{bmatrix} \circ X.$$

Applying forward elimination and backward substitution, one gets

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (0,2) \\ (0,2) \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (4,4) \\ (0,3) \end{bmatrix} \circ 0 = \begin{bmatrix} (4,4) \circ 0 \\ (4,4) \circ 0 \\ (2,3) \circ X \end{bmatrix}.$$

Using another iteration, one gets

$$v = \begin{bmatrix} \phi & \phi & \phi \\ \phi & \phi & \phi \\ \phi & \phi & \phi \end{bmatrix} v + \begin{bmatrix} (2,2) \\ (2,2) \\ (2,2) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (4,4) \\ \phi \end{bmatrix} \circ 0 + \begin{bmatrix} (1,1) \\ (1,1) \\ (1,1) \end{bmatrix} \circ X.$$

The new solution is then

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (4,4) \circ 0 \\ (4,4) \circ 0 \\ (2,2) \circ 1 \end{bmatrix}$$

which can be confirmed to be the solution by applying another iteration.

Example 4: Consider the circuit shown in Fig. 8 where the weight of each transistor is shown on the figure, and the weights of all storage nodes are 1. Suppose A = B = C = D = 1, E = 0; and let the initial conditions at all the storage nodes be X. The network equations are give by

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi & (3,3) & \phi \\ (3,3) & \phi & (3,3) \\ \phi & (3,3) & \phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} (2,2) \\ (4,4) \\ \phi \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ \phi \\ \phi \end{bmatrix} \circ 0 + \begin{bmatrix} (1,1) \\ (1,1) \\ (1,1) \end{bmatrix} \circ X.$$

Forward elimination gives:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi \\ (4,4) \\ (3,3) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ \phi \\ \phi \end{bmatrix} \circ 0$$

while backward substitution gives

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (3,3) \\ (4,4) \\ (3,3) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ \phi \\ \phi \end{bmatrix} \circ 0 = \begin{bmatrix} (4,4) \circ 0 \\ (4,4) \circ 1 \\ (3,3) \circ 1 \end{bmatrix}.$$

Note that in this problem we could have collapsed the **b** vectors into one at the outset. Now suppose A = C = E = 1, B = D = X, then the equations become:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi & (0,3) & (3,3) \\ (0,3) & \phi & (0,3) \\ (3,3) & (0,3) & \phi \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} (2,2) \\ (4,4) \\ \phi \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ \phi \\ \phi \end{bmatrix} \circ 0 + \begin{bmatrix} (1,1) \\ (1,1) \\ (1,1) \end{bmatrix} \circ X.$$

Forward substitution gives

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \phi \\ (4,4) \\ (3,3) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ \phi \\ (3,3) \end{bmatrix} \circ 0$$

while backward substitution gives

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} (3,3) \\ (4,4) \\ (3,3) \end{bmatrix} \circ 1 + \begin{bmatrix} (4,4) \\ (0,3) \\ (3,3) \end{bmatrix} \circ 0 = \begin{bmatrix} (4,4) \circ 0 \\ (4,4) \circ 1 \\ (3,3) \circ X \end{bmatrix}.$$

VI. Implementation and Results

We have implemented the above algorithms as a part of a mixed-mode simulator which partitions MOS circuits into channel-connected subcircuits. The switch-level solution mode uses the same data structure and storage, and the same equation ordering as the circuit analysis part of the program. For switch-level simulation only the lower (or upper) part of the matrix A need to be formulated. Sparse matrix techniques are used. During the reordering phase, nodes with single or no off-diagonal row (or column) elements after symbolic elimination are chosen first (Markowitz criterion). These nodes do not require any operations during

the equation elimination process, Eq. (22). The simulator allows zero-delay, unit-delay, or multiple-delay simulation. Both large combinational and sequential circuits have been analyzed. The accuracy of the results has been confirmed by performing circuit simulation. Compared to COSMOS [9], our switch-level logic simulator is slower, especially for circuits containing small size subcircuits. In this case the compiled functions of each subcircuit is small. The advantage of the matrix approach, however, is in analyzing large channel-connected subcircuits, such as tally circuits and barrel shifters. In addition, in our current implementation, some speed degradation occurs in having to load the matrices at every iteration. The speed is expected to improve by updating the matrices rather than reloading them at every iteration.

An advantage of our approach compared to the compiled approach is in fault simulation. It allows more realistic circuit-related switch-level fault models to be inserted in the circuit for fault simulation than the compiled approach. We have implemented a concurrent fault simulation using the proposed algebra, where transistor faults, line faults and bridging faults are simply inserted into the appropriate places in the matrix formulation [10]. Both output voltage monitoring and current testing are easily applied for fault detection.

VII. Conclusions

In this paper, we have presented a new algebraic method for switch-level simulation. The method employs a matrix formulation similar in structure to the underlying circuit nodal equations. The algebra uses logical max and min operations, equivalent to addition and multiplication, in formulating and solving the equations. Two rules on the order in which operations have to be carried out are also derived. These rules need to be followed in some cases

in order to arrive at a minimal solution. Gaussian elimination and sparse-matrix techniques are used in solving the equations.

The new approach has been found particularly useful in fault simulation where realistic circuit related faults can be easily injected into the matrix formulation. In addition, faults with variable strengths can be easily specified. The approach can also be easily extended to include switch-level network models with partial ordering [11].

Acknowledgement: The author would like to thank his students, Terry Lee, Tom Messerges, and Farid Najm, for their contributions to this work and for their efforts in implementing and testing the algorithms.

Appendix A

Gauss elimination eventually uses the right-hand side of each equation k in (18) one at a time to eliminate the k-th variable from the remaining equations. If we define the original system in (18) by

$$\mathbf{v} = \mathbf{A}^{(0)}\,\mathbf{v} + \mathbf{b}^{(0)}. \tag{A.1}$$

Gauss elimination derives n new systems

$$\mathbf{v} = \mathbf{A}^{(k)}\,\mathbf{v} + \mathbf{b}^{(k)},\ k = 1,2,\ldots,n \tag{A.2}$$

where the final system $\mathbf{A}^{(n)}$ is strictly upper triangular, which can then be solved by backward substitution. What is needed is to show that the systems in (A.2) all have the same least solution. Suppose that at the (k-1) step in the Gauss elimination process, the system in (A.2) is partitioned as:

$$\mathbf{A}^{(k-1)} = \begin{bmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} & \mathbf{A}_{13}^{(k-1)} \\ \mathbf{\Phi} & a_{22}^{(k-1)} & \mathbf{A}_{23}^{(k-1)} \\ \mathbf{\Phi} & \mathbf{A}_{32}^{(k-1)} & \mathbf{A}_{33}^{(k-1)} \end{bmatrix},\ \mathbf{b}^{(k-1)} = \begin{bmatrix} \mathbf{b}_1^{(k-1)} \\ b_2^{(k-1)} \\ \mathbf{b}_3^{(k-1)} \end{bmatrix} \tag{A.3}$$

where $\mathbf{A}_{11}^{(k-1)}$, $a_{22}^{(k-1)}$, $\mathbf{A}_{33}^{(k-1)}$ are of order k-1, 1, and n-k, respectively. Define

$$\mathbf{A}^{(k-1)} = \mathbf{Q}^{(k)} + \mathbf{R}^{(k)}$$

$$\text{where } \mathbf{Q}^{(k)} = \begin{bmatrix} \mathbf{\Phi} & \mathbf{\Phi} & \mathbf{\Phi} \\ \mathbf{\Phi} & a_{22}^{(k-1)} & \mathbf{\Phi} \\ \mathbf{\Phi} & \mathbf{A}_{32}^{(k-1)} & \mathbf{\Phi} \end{bmatrix},\ \mathbf{R}^{(k)} = \begin{bmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} & \mathbf{A}_{13}^{(k-1)} \\ \mathbf{\Phi} & \phi & \mathbf{A}_{23}^{(k-1)} \\ \mathbf{\Phi} & \mathbf{\Phi} & \mathbf{A}_{33}^{(k-1)} \end{bmatrix}. \tag{A.4}$$

System k is obtained from (A.4) by eliminating the k-th variable:

$$\begin{aligned} \mathbf{A}^{(k)} &= \mathbf{Q}^{(k)*}\,\mathbf{R}^{(k)} \\ \mathbf{b}^{(k)} &= \mathbf{Q}^{(k)*}\,\mathbf{b}^{(k-1)} \end{aligned} \tag{A.5}$$

If **A** is absorptive, then $\mathbf{Q}^{(k)}$ and $\mathbf{A}^{(k)}$ can be shown to be also absorptive [7], and hence stable, at every step in the elimination process. Thus, the closure $a_{22}^{(k-1)*} = e$.

In particular,

$$\mathbf{Q}^{(k)*} = \begin{bmatrix} \mathbf{E} & \mathbf{\Phi} & \mathbf{\Phi} \\ \mathbf{\Phi} & e & \mathbf{\Phi} \\ \mathbf{\Phi} & \mathbf{A}_{32}^{(k-1)} & \mathbf{E} \end{bmatrix} \tag{A.6}$$

$$\begin{aligned} \mathbf{A}^{(k)} &= \mathbf{Q}^{(k)*}\,\mathbf{R}^{(k)} \\ &= \begin{bmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} & \mathbf{A}_{13}^{(k-1)} \\ \mathbf{\Phi} & \phi & \mathbf{A}_{23}^{(k-1)} \\ \mathbf{\Phi} & \mathbf{\Phi} & \mathbf{A}_{33}^{(k-1)} + \mathbf{A}_{32}^{(k-1)}\,\mathbf{A}_{23}^{(k-1)} \end{bmatrix} \end{aligned} \tag{A.7}$$

$$\begin{aligned} \mathbf{b}^{(k)} &= \mathbf{Q}^{(k)*}\,\mathbf{b}^{(k-1)} \\ &= \begin{bmatrix} \mathbf{b}_1^{(k-1)} \\ \mathbf{b}_2^{(k-1)} \\ \mathbf{b}_3^{(k-1)} + \mathbf{A}_{32}^{(k-1)}\,\mathbf{b}_2^{(k-1)} \end{bmatrix}. \end{aligned} \tag{A.8}$$

Note that all the elements on and below the diagonal in the first k columns of $\mathbf{A}^{(k)}$ are ϕ.

At step k, the solution is obtained from (A.5) by:

$$\begin{aligned} \mathbf{v} &= \mathbf{A}^{(k)*}\,\mathbf{b}^{(k)} \\ &= (\mathbf{Q}^{(k)*}\,\mathbf{R}^{(k)})^*\,\mathbf{Q}^{(k)*}\,\mathbf{b}^{(k-1)} \\ &= (\mathbf{Q}^{(k)} + \mathbf{R}^{(k)})^*\,\mathbf{b}^{(k-1)} \\ &= \mathbf{A}^{(k-1)*}\,\mathbf{b}^{(k-1)}, k = 1,2,\ldots,n \end{aligned} \tag{A.9}$$

where we have used the identity $(\mathbf{Q}+\mathbf{R})^* = (\mathbf{Q}^*\mathbf{R})^*\mathbf{Q}^*$ [7, p.91] and (A.4) to arrive at the final result in (A.6). Thus all the n systems in (A.1) have the same least solution.

References

[1] R. E. Bryant, "An Algorithm for MOS Logic Simulation," Lambda, 4th qtr. 1980, pp. 46-53.

[2] R. E. Bryant, "A Switch-Level model and Simulator for MOS Digital Systems," *IEEE Trans. Computers*, February 1984, pp. 160-177.

[3] J. P. Hayes, "A Unified Switching Theory with Applications to VLSI Design," *Proc. IEEE*, 1982, pp. 1140-1151.

[4] R. H. Byrd, G. D. Hachtel, M. R. Lightner, and M. H. Heydemann, "Switch-Level Simulation: Models, Theory, and Algorithms," *Computer-Aided Design of VLSI Circuits and Systems*, Vol. 1, JAI Press, Greenwich, CT, 1986.

[5] I. N. Hajj, "A Path Algebra for Switch-Level Simulation," *IEEE Int. Conf. on Computer-Aided Design*, Santa Clara, CA, pp. 153-155, November 1985.

[6] R. E. Bryant, "Algorithmic Aspects of Symbolic Switch Network Analysis," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-6, no. 4, pp. 618-633, July 1987.

[7] B. A. Carre´, *Graphs and Networks*. Clarenon Press, Oxford, England 1979.

[8] R. E. Bryant, "Switch-Level Algorithms," *IEEE Design & Test of Computers*, Vol. 4, no. 4, pp. 26-40, August 1987.

[9] R. E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffer, "COSMOS: A Compiled Simulator for MOS Circuits," *Proc. 24th Design Automation Conference*, pp. 9-16, June 1987.

[10] T. Lee and I. N. Hajj, "A Switch-Level Matrix Approach to Transistor-Level Fault Simulation," *International Conference on Computer-Aided Design, Santa Clara, CA, November 1991, (accepted).*

[11] P. Agrawal, S. H. Robinson, and T. G. Szymanski, "Automatic Modeling of Switch-Level Networks using Partial Orders," *IEEE Transactions on Computer-Aided Design*, Vol. 9, no 7, pp. 696-708, July 1990.

Figure Captions

Fig. 1(a): Parallel combination of two branches, where $a_3 < a_2$.

Fig. 1(b): Series combination of two branches, $a_3 < a_2$.

Fig. 2: Storage node v connected to an input node I by a branch of strength (a_1,a_2).

Fig. 3: Storage node v connected to two input nodes I_1 and I_2.

Fig. 4: Two storage nodes connected in series to an input node.

Fig. 5(a): NMOS NOR gate.

Fig. 5(b): Switch-level network model.

Fig. 6(a): NMOS circuit with pass transistors.

Fig. 6(b): Corresponding switch-level network model.

Fig. 7: XNOR circuit.

Fig. 8: NMOS circuit with pass transistors.

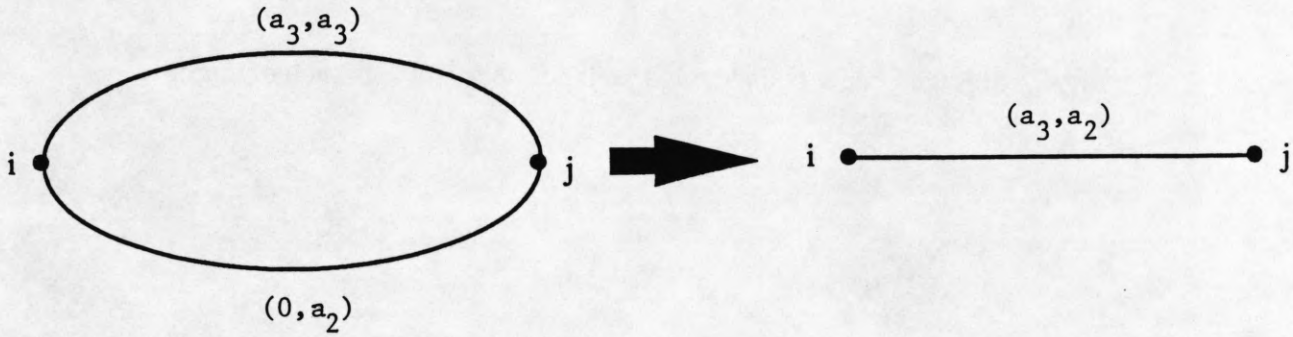


Fig. 1(a) : Parallel combination of two branches, where $a_3 < a_2$.

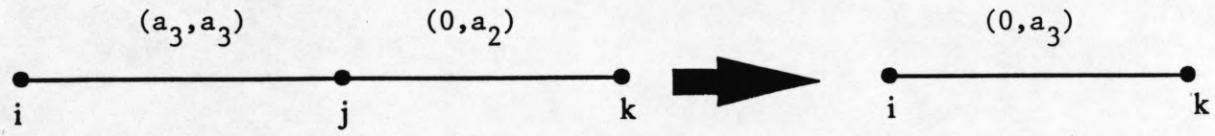


Fig. 1(b) : Series combination of two branches, $a_3 < a_2$.

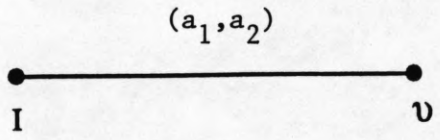


Fig. 2 : Storage node v connected to an input node I by a branch of strength (a_1, a_2).

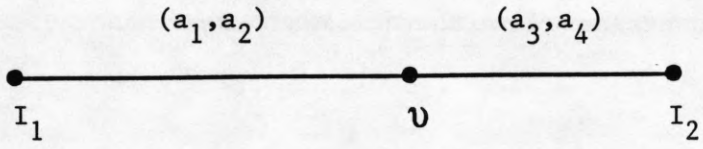


Fig. 3 : Storage node v connected to two input nodes I_1 and I_2.

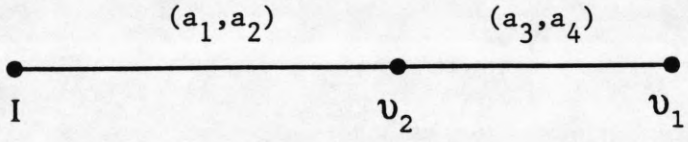


Fig. 4 : Two storage nodes connected in series to an input node.

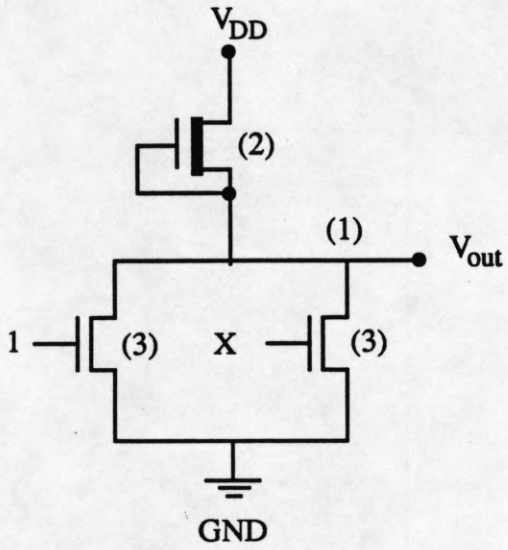
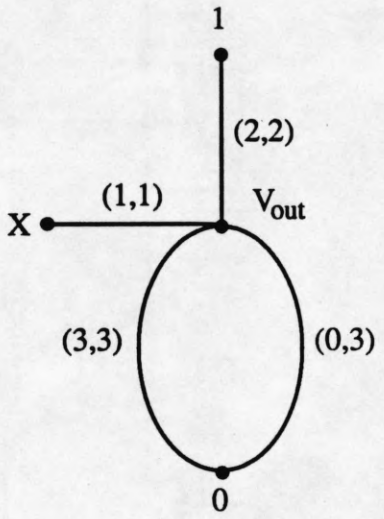


Fig. 5(a) : NMOS NOR gate.



(b) : Switch-level network model.

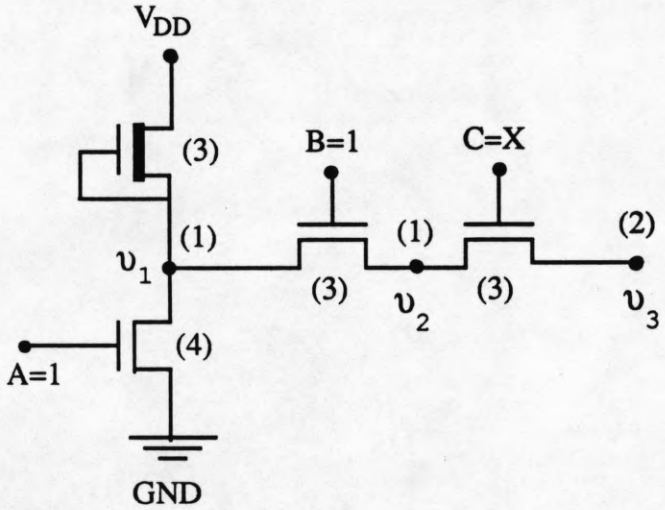
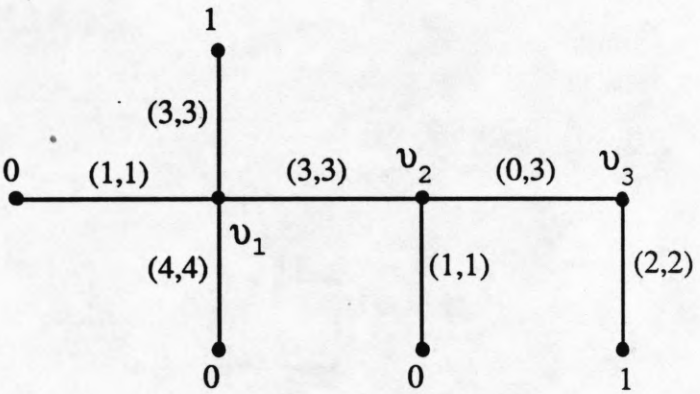


Fig. 6(a) : NMOS circuit with pass transistors.



(b) : Corresponding switch-level network model.

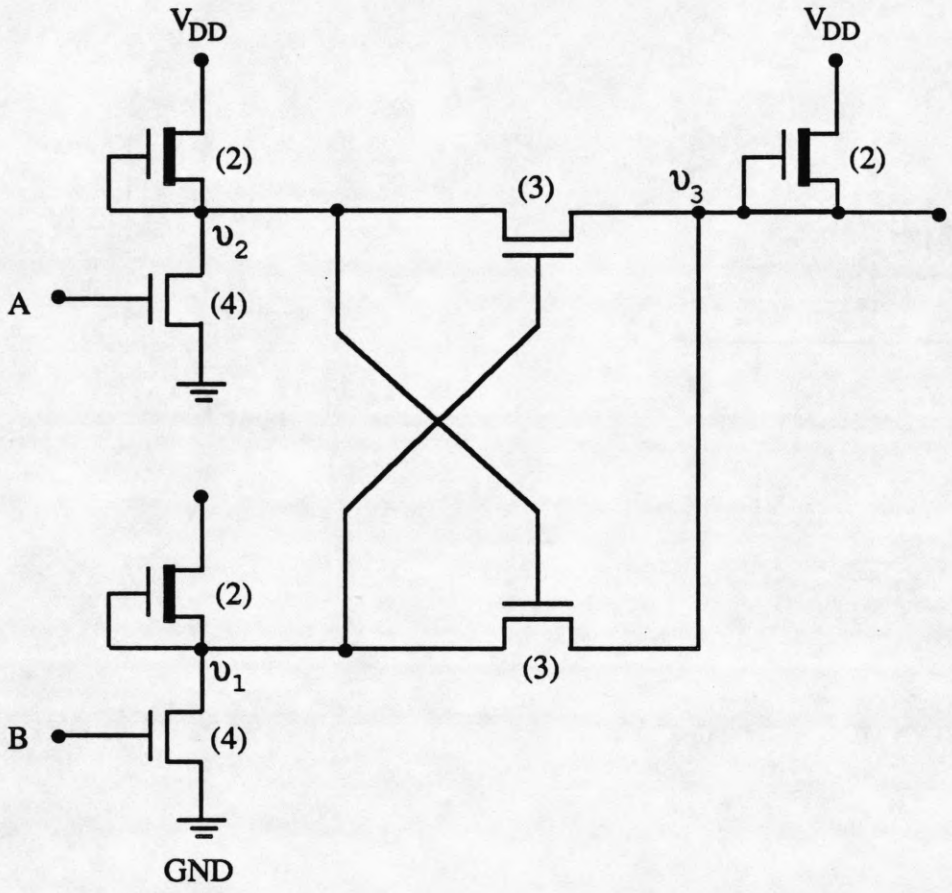


Fig. 7 : XNOR circuit.

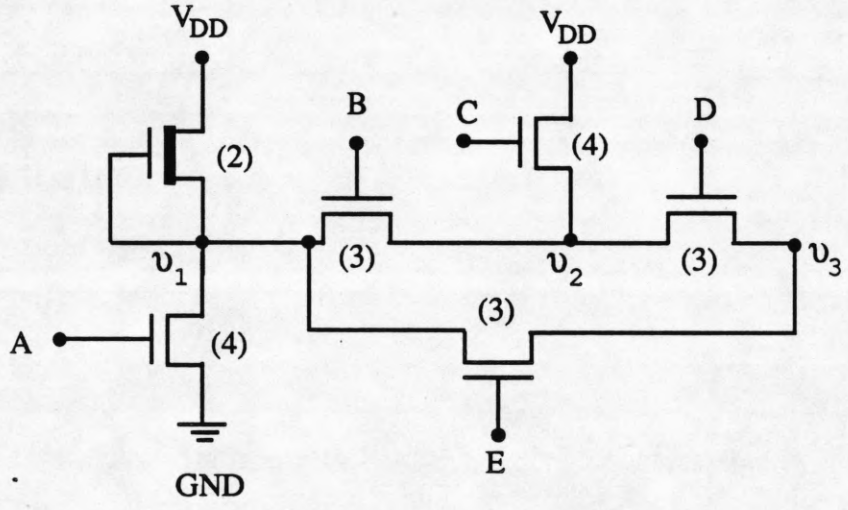


Fig. 8 : NMOS circuit with pass transistors.