Mixed Algebraic and Boolean Symbolic Analysis of Switched Linear Networks: Theory and Applications to Coupled Gate and Interconnect Delay Modeling

Sambuddha Bhattacharya and C.-J. Richard Shi, SENIOR MEMBER, IEEE

Mixed-Signal CAD Research Laboratory,
Department of Electrical Engineering, University of Washington
Seattle, WA 98195, USA
{sbb,cjshi}@ee.washington.edu

UWEE Technical Report
Number UWEETR-2003-0018
August 18, 2003

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Mixed Algebraic and Boolean Symbolic Analysis of Switched Linear Networks: Theory and Applications to Coupled Gate and Interconnect Delay Modeling*

Sambuddha Bhattacharya and C.-J. Richard Shi, SENIOR MEMBER, IEEE

Mixed-Signal CAD Research Laboratory,
Department of Electrical Engineering, University of Washington
Seattle, WA 98195, USA
{sbb,cjshi}@ee.washington.edu

ABSTRACT: Abstraction of digital MOS circuits as switch-level networks has proven very useful for VLSI automation. Symbolic analysis of switch-level networks by representing the signal at the gate of a MOS transistor as a Boolean variable has been shown to be efficient for verification in terms of functionality and timing. For this work, in addition to expressing the signal at the gate of MOS transistors as Boolean variables, we represent the design parameters of the transistors as algebraic symbols. In this paper, we present the theory for unified symbolic analysis in terms of both algebraic and Boolean symbols. Our formulation is general and is valid for extensions of conventional switched linear networks with various circuit elements like passive elements, current sources, etc.

The theory developed for mixed algebraic and Boolean symbolic analysis is applied for symbolic delay estimation in logic-stages with interconnects. Instead of conventional numeric delay estimation techniques, our method provides a single closed-form analytic delay expression that is symbolic in terms of both Boolean variables and the geometric design parameters of transistors and interconnects. Such expressions provide accurate estimations of signal delay over any input pattern and extensive variation in design variables like width and lengths of transistors and interconnects. This is implemented in a computer program and validated on modern VLSI technologies.

KEYWORDS: *Switch-level Networks, Symbolic Analysis, Multi-Terminal Determinant Decision Diagram, Logic stage, Interconnect, Moments, Delay Estimation.*

* This research was supported by U.S. Defense Advanced Research Projects Agency's (DARPA) NeoCAD program under Grant No. 66001-01-1-8920 and by the National Science Foundation's (NSF) CAREER award under Grant No. 9985507.

1. Introduction

The *Metal Oxide Semiconductor* (MOS) based digital circuits have been successfully modeled as linear switch-level networks [3]. These models capture various important aspects of digital MOS networks like bi-directionality, charge storage and the uniqueness of different logic families that are often impossible to model at the gate level. On the other hand, the device-level models capture too many details that may not be necessary at different stages of the design cycle. The appeal of the switch-level model, thus, stems from its attractive trade-off between resolution/accuracy and computational complexity with respect to the gate-level and electrical-level models.

Over the years, the area of switch-level network analysis has established its importance in the field of *Very Large Scale Integration* (VLSI) automations. The main thrust of research in this area has traditionally been in switch-level simulations [25], testing [26] and symbolic functional analysis [27]. Recently, symbolic functional analysis schemes that symbolically represent the input logic patterns of a switch-level network have been extended for the purpose of timing verification and analysis. A *Binary Decision Diagram* (BDD) [5] based method of estimating the signal delay in logic stages by replacing the transistors with their equivalent on-resistances and capacitances has been reported in [7]. A similar scheme of delay estimation based on the more general *Multi-Terminal Binary Decision Diagram* (MTBDD) [6] is presented in [10]. These methods in essence compute the RC-tree delay by modeling conducting transistors with their switch-resistor model and then construct decision diagrams with pattern dependent delay information.

The main advantage of symbolic analysis schemes for switch-level networks reported until now lies in pre-creating the logic decision diagrams for repeated functional evaluation. This pre-empts regressing to expensive graph-based switch-network search algorithms every time a new logic pattern is input to the circuit [27]. Unfortunately, there are some emerging issues that the current symbolic analysis schemes [7][10] fail to address. Firstly, the aggressive VLSI design styles with tighter delay bounds, lesser area and minimal power dissipation demand many iterations and optimizations. Specifically, various pre and post-layout transistor, gate and interconnect optimization schemes [28][29] are being successfully applied resulting in improved designs. A tighter coupling between the optimizations and timing analysis and verification schemes naturally would improve design/timing convergence. This would demand one further level of symbolic characterization of switched-networks: namely, symbolically expressing the

transistors and interconnects in terms of geometric and design parameters. Secondly, as the semiconductor processes migrate rapidly towards smaller feature sizes, a significant degree of process, voltage and temperature variations induce deviations from desired design performance. Consequently, designers need to execute long simulations of variational analysis to ensure correct functionality [30]. A symbolic switch-network analysis scheme with symbolic representation of variable parameters could address these issues early in the design cycle.

This paper presents a generalized theory for the symbolic analysis of switched linear networks. In addition to the input signals as Boolean variables, all circuit design/geometric parameters are also incorporated symbolically in our scheme. Thus, varying transistor sizes or resistor widths can be expressed symbolically unlike the methods presented in [7][10]. Starting directly from the basic circuit equations, this scheme develops a method for the symbolic analysis of arbitrary switched linear networks comprising of (but not restricted to) MOS transistors, resistors, capacitors and inductors. For this, we use a technique called *Multi-Terminal Determinant Decision Diagram* (MTDDD) [14], introduced recently in the context symbolic analysis of analog circuits. MTDDDs originally developed for efficient symbolic representation of algebraic expressions are extended for handling Boolean functions.

In this paper, we apply our theory for the specific case of delay analysis and estimation in switched linear networks. In our delay estimation scheme, we incorporate several ideas that are not addressed by the previous works in [7][10]. First, we include computation of interconnect delay along with the delay through the logic blocks. Second, unlike the largely inaccurate Elmore model [8] used in [7][10], we adopt a delay model based computation of higher order moments [15]. Third, the symbolic computation of moments directly from the circuit equations ensures accurate estimation of delays for loops of conducting transistors.

Accurate estimation of delay for the switch-resistor models of MOS transistors requires accurate modeling of the equivalent effective resistance and capacitance. We develop a modified scheme to obtain a pre-simulation based lookup table for effective resistance.

Some preliminary results of this paper were presented in [2]. The paper is organized as follows. Section 2 presents a brief introduction to switched linear networks and delay estimation schemes. Mixed numeric-symbolic analysis of circuit equations and moments with MTDDDs is explained in Section 3. Section 4 develops the theory of compact circuit and moment equations for switched linear networks. An enumeration scheme for Boolean sensitization conditions is presented in Section 5. The construction of moment MTDDDs is described in Section 6. Section 7 elaborates the effective resistance estimation for MOS transistors. Section 8 presents the experimental results.

2. Preliminary

2.1 Switched Linear Network Model

A MOS transistor is often abstracted as a switch. It conditionally forms a connection between the *drain* and the *source* nodes depending on the voltage at its *gate* node. Digital circuits consisting of MOS transistors have been successfully modeled and analyzed with switch-level networks [3][4]. Such switch-level network are formally defined as a set of nodes and MOS transistors, where some nodes are classified as inputs and some are classified as storage nodes. The input nodes represent connection to a signal source external to the chip or the part of the circuit under consideration. The storage nodes have characteristics of a capacitor – they retain their state in absence of inputs and may share the charge stored in them with other storage nodes. The MOS transistor is modeled by a resistive switch between the transistor's drain and source terminals, and two grounded capacitors at its drain, and source nodes.

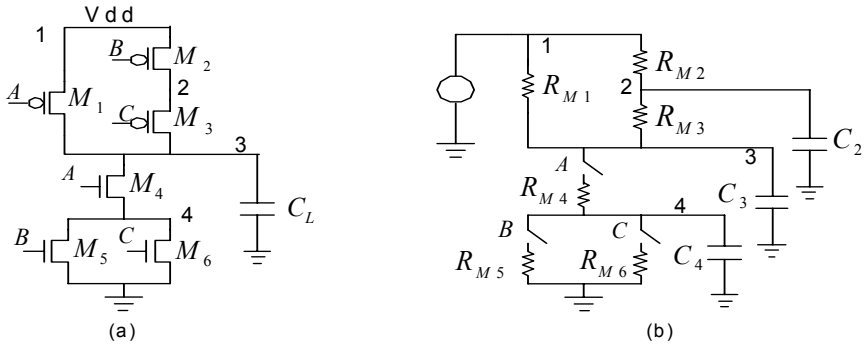


Figure 1: (a) A logic-cell that implements the function $\overline{A(B+C)}$ (b) A switched linear network model of the logic cell when inputs *A,B,C* are all logical *0*. In the model, each transistor is replaced by an equivalent resistor between the drain and the source, and two grounded capacitors at the drain and source terminals. The capacitor at node *2*, C_2 , is the sum of capacitive contributions due to transistors M_2 and M_3. The load capacitance models the capacitance at the gate of the next level being driven by this cell.

Figure 1 represents the switched linear network model for an OAI21 logic cell. Transistor M_i in the logic-cell is replaced by an equivalent resistor R_{Mi} in the model. The capacitance at each node in the equivalent circuit is due to the capacitive contributions of each transistor incident on the node. Thus, *node 2* has capacitive contributions due to the transistors M_2 and M_3, and *node 3* has capacitive contributions from the load capacitor C_L and transistors M_1, M_3 and M_4. Large circuits comprising of MOS transistors can be partitioned into smaller *Channel-Connected Regions* (CCR) that consists of conducting transistors connected to each

other through their drains and sources. Each CCR forms a logic-cell or a logic blocks and can be modeled as a switched linear circuit as illustrated in Figure 1.

Interconnections between logic-cells or CCRs are modeled as RC tree or mesh networks. A logic-stage in VLSI comprises of a logic-cell and an interconnect RC structure between that logic-cell and other logic-cells driven by it as shown in Figure 2(a). The corresponding logic-stage switched network model includes the interconnect RC tree as has been illustrated in Figure 2(b). In this paper, we deal with a more general form of switched linear networks that can comprise resistors, capacitors and inductors along with MOS transistors.

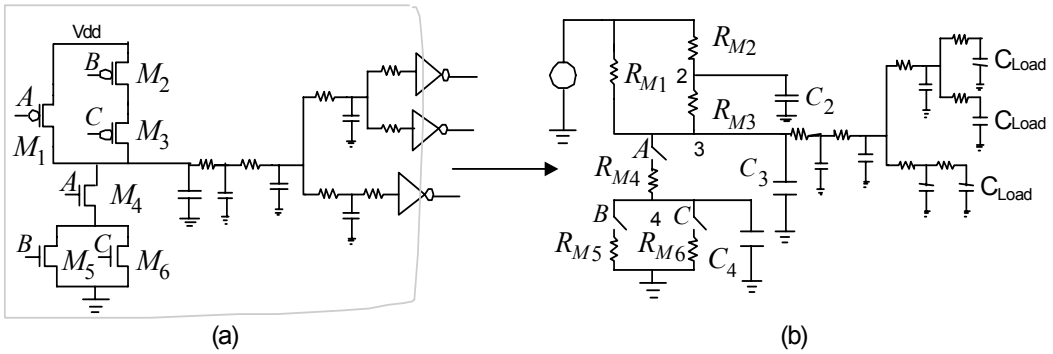


Figure 2: An example logic stage (a) Logic cell with interconnect modeled with RC tree consists of a logic stage (b) Switched Linear Network model for the logic stage. The network model includes the RC tree along with the switch-resistor models for the MOS transistors.

2.2 Signal Delay Estimation in Logic Stages

As illustrated in Figure 2, a large digital VLSI network can be reduced into a logic stages that comprises logic cells and interconnects. Until the last few steps in the design cycle, signal delay through large VLSI networks is estimated in terms of the macro-models for signal delay through individual stages [13]. The signal delay through a logic stage consists of delay components in the logic cell and the interconnect.

Traditionally, the total signal delay through a logic stage is estimated by one of the following schemes: (i) representing logic-cells by equivalent switch-resistors and analyzing the entire circuit as an RC structure [12] or (ii) estimating cell-delay by empirically deriving delay expressions in terms of load capacitance and input-signal transition time [11] and separately computing delay in interconnects. The advantage in the first method is that the entire logic-stage is modeled as a linearized RC network allowing direct delay computation. This paper adopts this estimation method and provides a symbolic framework for estimating delays in generalized switched linear networks.

2.3 Moments in Linear Networks

Delay estimation for RC trees based on a single dominant-pole Elmore model [8] was first popularized in [21]. Extension to arbitrary linear networks including grounded resistors was presented in [22]. The generalized theory of delay estimation from the circuit equations of linearized networks was developed in [15]. For a linearized network, the circuit equations in the Tableau form [20] can be written as

$$\mathbf{C}\dot{\mathbf{x}} = \mathbf{G}\mathbf{x} + \mathbf{b}\mathbf{u} \qquad \mathbf{x}|_{t=0} = \mathbf{x}_0 \tag{1}$$

where $\mathbf{x}$ is a vector composed of node voltages and necessary branch currents, $\mathbf{G}$ is the modified conductance matrix, $\mathbf{C}$ is the capacitance and inductance matrix and $\mathbf{u}$ is due to the system's inputs. For example, the normalized Tableau equations for the equivalent RC circuit of Figure 1(b), is given as:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & R_{M1} & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & R_{M2} & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & R_{M3} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v(1) \\ v(2) \\ v(3) \\ I_1 \\ I_2 \\ I_3 \\ I_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ C_2\dot{v}(2) \\ C_3\dot{v}(3) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

The voltage at node i is represented as $v(i)$. I_7 represents the current in the voltage source, and any other I_P represents the current in R_{MP}. The first three rows in Eq. (2) represent the KCL equations. The remaining four rows relate the voltages and currents in each branch. Here, we present Eq. (2) as we develop our theory of generalized mixed algebraic and Boolean symbolic analysis of switched linear networks from these basic network equations.

For our method, the conductance matrix **G** in Eq. (2) is a mixed numeric-symbolic matrix where the entries R_{MP} are *symbolic* and the rest of the entries are numeric. On the right-hand-side, the capacitors C_2 and C_3 are symbolic entries. In the *Laplace domain*, Eq. (1) can be expressed as

$$s\mathbf{C}\mathbf{X} - \mathbf{G}\mathbf{X} = \mathbf{b}\mathbf{U} \tag{3}$$

From Eq. (3), the transfer functions of the circuit can be written in the matrix-vector form as

$$\mathbf{H}(s) = (\mathbf{C}s - \mathbf{G})^{-1}\mathbf{b} \tag{4}$$

The *Taylor* series of $\mathbf{H}(s)$ about $s=0$ can be expressed in the form

$$\mathbf{H}(s) = \mathbf{m}_0 + \mathbf{m}_1 s^1 + \mathbf{m}_2 s^2 + ... + \mathbf{m}_n s^n + ... \tag{5}$$

where $\mathbf{m}_i$ is an n-dimensional vector composed of the i^{th} *moments* of the circuit. Equating the like powers of s in Eq. (4) and Eq. (5), and from Eq. (1) and Eq. (3), the following recursive equations for computing moments are obtained [15].

$$\mathbf{m}_0 = \mathbf{x}_h(0) \tag{6}$$

$$\mathbf{G}\mathbf{m}_{k+1} = \mathbf{C}\mathbf{m}_k \tag{7}$$

For our example circuit of Figure 1(b), the recursive moment equations based on Eq. (2) and Eq. (7) can be written as

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & R_{M1} & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & R_{M2} & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & R_{M3} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} m_k(1) \\ m_k(2) \\ m_k(3) \\ m_k(I_1) \\ m_k(I_2) \\ m_k(I_3) \\ m_k(I_7) \end{bmatrix} = \begin{bmatrix} 0 \\ C_2 m_{k-1}(2) \\ C_3 m_{k-1}(3) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

where, the k^{th} moment at the i^{th} node is represented as $m_k(i)$ and the k^{th} moment for current I_P is represented as $m_k(I_P)$. Similar to Eq. (2), some entries in Eq. (8) are numeric while some are symbolic.

2.4 Delay Estimation Based on Low Order Moments

Typically, accurate yet efficient delay estimation is accomplished by incorporating the effects of the first few moments of the circuit. For example, the transfer functions of Eq. (5) can be approximated to the 2nd order as shown below.

$$\hat{\mathbf{H}}(s) = \mathbf{m}_0 + \mathbf{m}_1 s^1 + \mathbf{m}_2 s^2 \tag{9}$$

For the transfer functions in Eq. (9), the propagation delay at any node of the circuit can be expressed in terms of the 1^{st} and 2^{nd} moments of the transfer function at that node. Several delay metrics based on the first few moments have been proposed in literature [1][23][24]. The delay metric we use in this work is adopted from [1] and is given as

$$t_{pd}(j) = \ln(2)\frac{m_1^2(j)}{\sqrt{m_2(j)}} \tag{10}$$

where $m_1(j)$ and $m_2(j)$ are the first and second moments at node j. We chose this metric for its high accuracy even for nodes relatively close to the driving source.

For our example circuit of Figure 1, the expressions for the 1st moment at node 2 and the 1st and 2nd moments at node 3 obtained from Eq. (8) are given as:

$$m_1(2) = \frac{R_{M1}R_{M2}C_2 + R_{M1}R_{M2}C_3 + R_{M2}R_{M3}C_2}{R_{M1} + R_{M2} + R_{M3}} \tag{11}$$

$$m_1(3) = \frac{R_{M1}R_{M2}C_2 + R_{M1}R_{M2}C_3 + R_{M1}R_{M3}C_3}{R_{M1} + R_{M2} + R_{M3}} \tag{12}$$

$$m_2(3) = \frac{R_{M1}R_{M2}C_2m_1(2) + R_{M1}R_{M2}C_3m_1(3) + R_{M1}R_{M3}C_3m_1(3)}{R_{M1} + R_{M2} + R_{M3}} \tag{13}$$

Under an accurate effective resistance model as described in Section 7, very precise estimations of signal delay can be obtained from Eq. (13). For example, the 50% delay obtained through HSPICE[9] simulations for the *OAI21* cell of Figure 1 in TSMC 0.18um technology (W_p=2.8um, W_n=1.4um, L_p=L_n=0.2um, C_{load}=40fF) driven by an inverter (W_p=8u, W_n=2.7um, L_p=L_n=0.3um) is 100.19ps. The estimated delay according to the Eq. (13) is 99.72ps.

3. Mixed Numeric and Symbolic Moment Computation with MTDDDs

Clearly, the symbolic computation of moments is a key task for efficient symbolic delay estimation. The circuit moments are computed from the mixed numeric-symbolic recursive equation set represented in Eq. (7). This section describes the symbolic computation of circuit moments using a compact graph structure called MTDDD.

The recursive set represented by Eq. (7) can be re-written in the following form

$$\mathbf{Tx} = \mathbf{w} \tag{14}$$

where **T** is a mixed numeric-symbolic matrix, **w** is a mixed numeric-symbolic vector and **x** is the vector of unknowns solved for. Solving for the moments from Eq. (7) is akin to repeatedly solving Eq. (14). Similar to symbolic circuit analysis, our procedure for symbolic computation of moments is based on Cramer's rule for solving sets of linear equations [31]. The i^{th} element of $\mathbf{x}$ is obtained as

$$x_i = \sum_j \mathbf{w}_j \det(\mathbf{T}_{ij}) / \det(\mathbf{T}) \tag{15}$$

where $\det(\mathbf{T})$ is the determinant of the matrix $\mathbf{T}$, and $\det(\mathbf{T}_{ij})$ is the determinant of the matrix $\mathbf{T}$ after removing row i and column j, or the cofactor of the matrix **T** with respect to the element at (i,j). Therefore, the key task is the representation of the determinants and the cofactors of a mixed numeric-symbolic matrix.

For this purpose, we utilize an efficient technique called Multi-Terminal Determinant Decision Diagram (MTDDD), introduced recently in the context of symbolic circuit analysis [14]. An MTDDD is an ordered, rooted, directed acyclic graph. As illustrated in Figure 3, it consists of some symbolic vertices and a set of terminal vertices, which can be the *0-terminal* vertex, the *1-terminal* vertex and some numeric terminal vertices with non-zero values. A symbolic vertex *V*, is characterized by a label *(V.label),* and two edges, namely *1-edge* (solid line) and *0-edge* (dotted line) pointing, respectively, to its *1-child (V.1-child)* and *0-child (V.0-child)*. Thus, a vertex *V* represents a *semi-symbolic expression V.expr* defined recursively as follows:

If (*V* is *0-terminal*), then *V.expr=0*

if (*V* is *1-terminal*), then *V.expr=1*

if (*V* is numeric terminal), then *V.expr=V.value*

if (*V* is a symbolic vertex), then *V.expr=V.label*(V.1-child).expr + (V.0-child).expr*

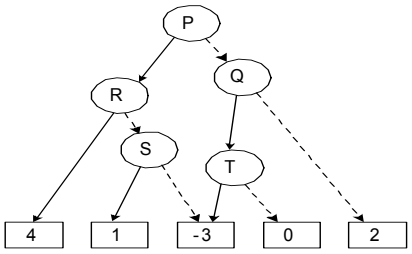


Figure 3: An MTDDD representing the algebraic expression *4PR + PS –3P –3QT + 2*. All terminal vertices are numeric and other vertices are symbolic. From any vertex, the *1-child* is reached along the solid-edge, and the *0-child* is reached along the dotted edge. A product term is obtained by traversing the 1-edges from the root to a terminal vertex.

In an MTDDD, a path from the root to the 1-terminal or a numeric-terminal represents a product term in the corresponding expression. Each such path contains exactly the same number

of 1-edges as the number of unique variables in the product term. The total number of such paths in an MTDDD structure is equal to the number of product terms in the expression. The entire expression represented by an MTDDD can be obtained by traversing all paths that correspond to product terms.

The algebraic expressions of the moments in Eq. (11) and Eq. (12) can be represented by MTDDD structures. Figure 4 comprises three MTDDD structures, one for the denominator and one each for the numerators, of the moment expressions in Eq. (11) and Eq. (12). Each symbolic vertex in the MTDDDs represents a variable element in the corresponding circuit, e.g., R_{M1}, C_2 etc. Multiple edges pointing to a vertex indicate sharing of the subgraph rooted at that vertex. Different types of subgraph sharing are observed for moment MTDDDs, viz., between two or more subgraphs of a moment MTDDD at a node, between subgraphs of the moment MTDDDs of the same order at different nodes, and between subgraphs of the moment MTDDDs of different order. In Figure 4, the subgraphs rooted at vertices R_{M2} and C_3 are shared by the two numerator MTDDDs.

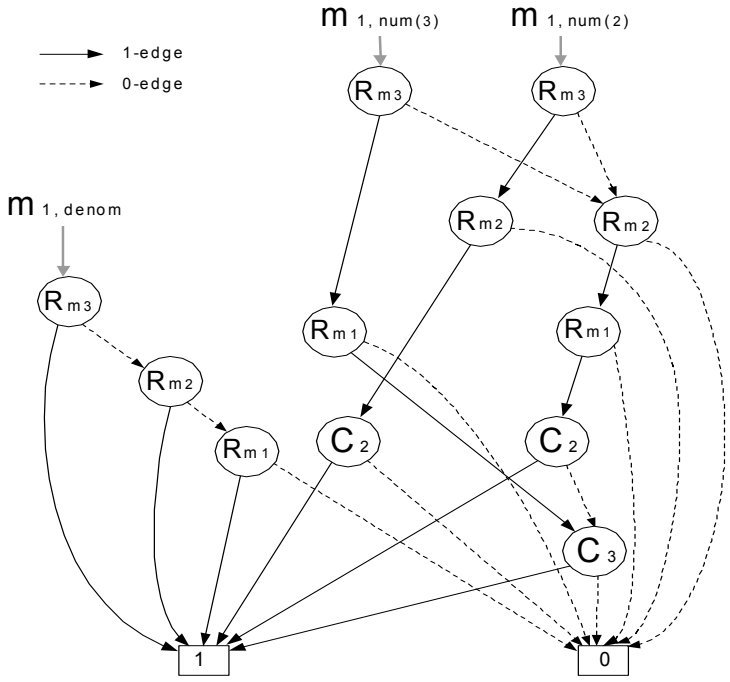


Figure 4: An MTDDD representing the *1st* moments at nodes *2* and *3* in the circuit of Figure 1(b). The lightly shaded arrows indicate the root of the MTDDDs for the denominator and numerators of the *1st* moment at nodes *2* and *3*. Multiple edges pointing to an MTDDD node indicates sharing of the subgraph rooted at that node (e.g., nodes R_{m2} , C_3).

To see how an MTDDD can be used to represent the determinant and cofactors of a circuit matrix, we consider the Laplace expansion of a matrix determinant with respect to a

particular element, $t_{i,j}$, at row i and column j. Then, the matrix determinant $\det(\mathbf{T})$ can be represented as follows:

$$\det(\mathbf{T}) = (-1)^{(i+j)} t_{i,j} \det(\mathbf{T}_{ij}) + \det(\mathbf{T}_{\overline{ij}}) \tag{16}$$

where, $\det(\mathbf{T}_{ij})$ is the cofactor of the matrix $\mathbf{T}$ with respect to the element $t_{i,j}$. The expression $\det(\mathbf{T}_{\overline{ij}})$ represents the remainder of the matrix $\mathbf{T}$ with respect to the element $t_{i,j}$, which is defined as the determinant of the matrix $\mathbf{T}$ after setting $t_{i,j}$ to 0. Clearly, if we can represent this expansion by a vertex with label $(-1)^{(i+j)} t_{i,j}$, the cofactor as the *1-child* and the remainder as the *0-child*, and then recursively expand the cofactor and the remainder, we can obtain an MTDDD.

Thus, for Eq. (7), the MTDDD for the moment of a given order can be constructed by recursively expanding the determinant and cofactors of the matrix associated with the equation set. The MTDDDs for the moments of the next higher order can be constructed next because of the recursive nature of Eq. (7).

4. Compact Moment Equations for Switched Linear Networks

This section describes the method of formulating generalized KCL and branch I-V equations and then arriving at compact moment equations for a switched linear network. Consider the example circuit of Figure 1. Clearly the circuit in Figure 1(b) represents the circuit of Figure 1(a) under only one input logic pattern. The moment equations for the circuit of Figure 1(b) are represented by Eq. (8). As is evident, the equivalent switch-resistor network of the cell of Figure 1(a) changes with each input pattern thereby resulting in a new set of moment equations for each pattern. Thus, for an ***n***-input logic cell, the 2^n different input patterns result in 2^n sets of moment equations like Eq. (8). Clearly, creating MTDDDs for each such moment equation set is inefficient. This motivates the need for a single set of moment equations valid for all input patterns.

For this, we first incorporate the input patterns as Boolean variables into Eq. (1), the circuit Tableau equations, by deriving a more general form KCL and branch equations. A single set of moment equations valid under all input logic patterns can then be obtained directly from these Boolean-ized Tableau equations.

The notations for the various Boolean operations are as follows: the operator $\wedge$ denotes the Boolean AND operation, the operator $\vee$ represents the Boolean OR operation. The

expression $\overline{A}$ indicates the Boolean NOT operation on the Boolean variable *A*. For clarity, the NOT operation on a Boolean function *F* is represented as $!F$.

4.1 Generalized Branch I-V Equations

Consider the circuit of Figure 1. We recognize that the current in a resistive branch is zero when the switch in the branch is open. The branch I-V equations can be modified to incorporate the switching effect of a transistor. For the switch-resistor model of the transistor in Figure 5, the modified branch equation can be written as

$$\overline{X}(V_1 - V_2) = IR_{M1} \tag{17}$$

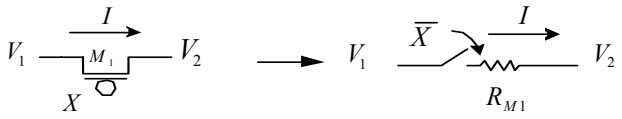


Figure 5: A MOS transistor and its switch-resistor form.

Thus, the new *I-V* equation for the branch corresponding to the transistor M_3 in Figure 1(b) can be written as

$$\overline{C}(V_2 - V_3) = I_3 R_{M3} \tag{18}$$

We modify every branch *I-V* equation of the circuit into this Boolean-ized form.

4.2 Generalized KCL Equations

Similar to the branch equations, the KCL equations need to be generalized for validity under changing input logic patterns. Consider the simple example in Figure 6, the node *1* is isolated if all of the switches A , B and C in the branches incident on it are open, i.e., if none of the currents i_1 , i_2 and i_3 exist. More generally, a node is isolated if there is no sensitizing path from either the voltage source or the ground leading to that node. In our example circuit of Figure 1(b), if node *2* becomes isolated because of opening the switches at $\overline{B}$ and $\overline{C}$, Eq. (2) is no longer valid. In other words, the matrix **G** in Eq. (2) is rendered singular by the isolated node.

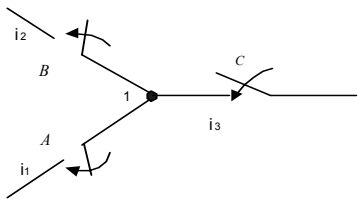


Figure 6: Isolation of node when all branches connecting to it have open switches.

Clearly, to obtain a single set of circuit equations for switched networks, the issue of isolation of a node needs to be addressed. We make the following observations:

Observation 1: *The presence of non-zero current in any branch connected to a node and the isolation of that node are mutually exclusive events.*

Observation 2: *An isolated node has no effect on any branch or any other node in a CCR.*

These observations indicate the close relationship between the isolation of a node and the KCL equation at that node. For the circuit of Figure 1(b), the Boolean conditions for the existence of the currents I_2 and I_3 are given by

$$F_{I_2} = \overline{B} \quad , \quad F_{I_3} = \overline{A} \wedge \overline{C} \ \vee \ \overline{B} \wedge \overline{C} \tag{19}$$

In this paper, we call each Boolean product term that activates a path to a given branch or node from the power or the ground node an *active Boolean path*. Thus the Boolean function represented in F_{I2} is an active Boolean path. Each Boolean product term in F_{I3} is an active Boolean path. In other words, the condition that a branch current exists is the union of all its active Boolean paths. Following Observation 1, the condition that a node is isolated is the negation of the union of all active Boolean paths of all branches connected to that node. Thus, the condition that node *2* in Figure 1(b) is isolated is given by

$$F_2 = \overline{(\overline{A} \wedge \overline{C} \ \vee \ \overline{B})} \tag{20}$$

We transform the KCL equation at node *2* in (2) to a Boolean-ized form as follows

$$F_2 V_2 - F_{I_2} I_2 + F_{I_3} I_3 = C_2 \dot{V}_2 \tag{21}$$

such that there is no singularity in matrix **G** due to the conditional isolation of node 2. In general, this Boolean-ized form of KCL can be written as

$$\bigvee_j F_{I_j} I_j + \left(\overline{\bigvee_j F_{I_j}} \right) V = C \dot{V} \tag{22}$$

where F_j is the active Boolean path for the j^{th} branch at a node. In other words, if there are no active Boolean paths leading to a node, the voltage at the node is set to an arbitrary value.

4.3 Modified Tableau Equations for Switched Linear Networks

With the transformations defined in Eq. (17) and Eq. (22), we devise a method of transforming the circuit Tableau equations such that a single set of equations govern the circuit behavior irrespective of input logic pattern. The necessary and sufficient set of transformations required for obtaining a single set of equations that govern the behavior of switched networks under any input condition is stated below.

Theorem 1: *Tableau equations for a stage circuit valid under any input condition can be obtained by*

(i) *Writing the equation set* $\mathbf{C}\dot{\mathbf{x}} = \mathbf{G}\mathbf{x} + \mathbf{b}\mathbf{u}$ *in Eq. (1) assuming every input to be 'on'.*

(ii) *Updating the branch equations with their Boolean-ized forms as shown in Eq. (17).*

(iii) *Updating the KCL equations in the original set with the Boolean conditions for isolation of nodes and presence of currents as illustrated in Eq. (22).*

Proof: Consider a switch-resistor network consisting of n nodes and b resistive branches and one voltage source. Under the condition that all switches are closed, the Tableau equations of the system consist of an $(n+b+1)$ dimension matrix system with each equation being linearly independent. The equation set consists of n KCL equations, b branch equations and one equation for the current in the voltage source. Suppose, due to a certain input pattern, k nodes are rendered isolated and p branches are switched open. Upon transformation of the branch equations to Boolean-ized forms, each of the p branch equations corresponding to the p open branches is of the form, $RI = 0$. Under our transformation scheme, each of the k KCL equations corresponding to the isolated nodes assumes the form $V = constant$. Therefore, the solution for $(p+k)$ out of $(n+b+1)$ variables is rather trivial. If the $(p+k)$ trivial variables and $(p+k)$ trivial equations are removed from the system, we still have a matrix equation set with $(n+b+1-p-k)$ independent equations for solving the remaining $(n+b+1-p-k)$ variables.

The proof of Theorem 1 is completed. □

For our example circuit of Figure 1(a), we first rewrite the Tableau equations of the equivalent switched network under the assumption that all switches are closed.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 & R_{M1} & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & R_{M2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & R_{M3} & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & R_{M4} & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & R_{M5} & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & R_{M6} & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ C_2 \dot{V}_2 \\ C_3 \dot{V}_3 \\ C_4 \dot{V}_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{23}$$

Then we update the matrix **G** in (23) according to the transformations suggested in Theorem 1. The complete set of changes introduced in **G** is shown in Table 1. The transformed Tableau equations can then be written as

$$\mathbf{C}'\dot{\mathbf{x}} = \mathbf{G}'\mathbf{x} + \mathbf{b}\mathbf{u} \qquad \mathbf{x}|_{t=0} = \mathbf{x}_0 \tag{24}$$

Table 1: Complete set of changes incorporated in matrix G of Eq. (23). For each location in the matrix, the table shows the original (Old) values and updated (New) values. Note that there are no changes for the locations (1,1), (3,3) and (5,5). This is because these nodes are never isolated under any input condition. For clarity and readability, we do not show the Boolean *AND* operator in the Boolean expressions.

Location	Old	New	Location	Old	New	Location	Old	New
(1,6)	1	$\overline{A}$	(1,7)	1	$\overline{B}$	(2,2)	0	$!(\overline{A}\overline{C} \vee \overline{B} \vee AB\overline{C})$
(2,7)	-1	$-\overline{B}$	(2,8)	1	$(\overline{A}\overline{C} \vee \overline{B}\overline{C} \vee AB\overline{C})$	(3,6)	-1	$-\overline{A}$
(3,8)	-1	$-(\overline{A}\overline{C} \vee \overline{B}\overline{C} \vee AB\overline{C})$	(3,9)	1	$(A\overline{B}\overline{C} \vee AB \vee AC)$	(4,4)	0	$!(A\overline{B}\overline{C} \vee B \vee C)$
(4,9)	-1	$-(A\overline{B}\overline{C} \vee AB \vee AC)$	(4,10)	1	B	(4,11)	1	C
(5,10)	-1	$-B$	(5,11)	-1	$-C$	(6,1)	-1	$-\overline{A}$
(6,3)	1	$\overline{A}$	(7,1)	-1	$-\overline{B}$	(7,2)	1	$\overline{B}$
(8,2)	-1	$-\overline{C}$	(8,3)	1	$\overline{C}$	(9,3)	-1	$-A$
(9,4)	1	A	(10,4)	-1	$-B$	(10,5)	1	B
(11,4)	-1	$-C$	(11,5)	1	C			

According to Eq. (7), and from the transformed Tableau equations in Eq. (24), we obtain a single set of recursive moment equations that are valid under any input logic pattern.

5. Enumeration of Sensitization Conditions

As is evident from Section 4, obtaining compact tableau equations that are valid under all input Boolean patterns requires identification of all active Boolean paths. This entails listing the sensitization condition at each node and branch of the circuit. The main complication in enumerating the sensitization conditions at every branch and node of a switched circuit arises from the bi-directionality of MOS transistors and the presence of parallel branches.

The problem of identifying sensitization conditions for switch-level networks has been addressed in different forms in the context of graph algorithms, switching theory and VLSI testing. One of the earlier works on symbolic switching theory state a method of enumerating all possible simple paths in a switched network [17] without explicitly dealing with the algorithmic and storage aspects. The solution of the Boolean equations corresponding to a switch-level network by Gaussian elimination was presented in [18]. It provides an efficient way of solving for the Boolean expressions at the nodes of the network. In the graph-theoretic context, efficient algorithms for enumerating all cutsets between a pair of nodes have been reported in literature [19]. Listing the cutsets between a node and the power/ground node is similar to identifying the conditions of isolation of the node.

As illustrated in Table 1, this work requires both the condition of isolation of the nodes and the active Boolean paths at the branches. For this, we implement an efficient iterative enumeration scheme that compactly represents the Boolean expressions as decision diagrams. We first briefly examine some basic graph-theoretic concepts and then describe our method of enumerating the sensitization conditions.

5.1 Preliminaries

An undirected graph is denoted by $\Gamma = (V, E)$, where V is a set of vertices and E is a set of edges, each represented by an unordered pair (v, w) of its end vertices. For an edge e, where $e = (v,w) \mid v,w \in V$, v and w are called its *adjacent* vertices and the edge is said to be *incident* to its adjacent vertices. Any switch-level circuit is an undirected graph, where the nodes of the circuits are the vertices and the branches correspond to the edges of the graph. Each edge e_i in a switch-level circuit has a Boolean variable e_{bi} associated with it.

A *path* between a pair of vertices v_0 and v_l in an undirected graph is a sequence of distinct edges $(v_0, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{l-1}, v_l)$. A path is called a *cycle* is if $v_0 = v_l$. For switch-level circuits, a path to a node i from the power or the ground node through a set of edges $e_0, e_1, e_2, \ldots, e_{i-1}$ forms an active Boolean path and corresponds to the Boolean product function $e_{b0} e_{b1} e_{b2} \ldots e_{bi-1}$. Clearly, each node can have multiple active Boolean paths associated with it. However, no active Boolean path can be completely contained in another. For example, if F_1 is an active Boolean path and F_2 represents a potential active Boolean path at a node v, and if $F_1 \subset F_2$ holds, then F_2 is not an active Boolean path.

A graph Γ is *connected* if there is a path between each pair of its vertices. A graph $\Gamma' = (V', E')$ is called a *subgraph* of Γ if $V' \subset V$ and $E' \subset E$. A connected graph without any cycle is called a *tree*. A tree T is said to be the *spanning tree* of a connected graph Γ if T is a subgraph of Γ and T contains all the vertices of Γ. Every edge of Γ that is not part of T is called a *link*.

5.2 Scheme for Enumerating the Sensitization Conditions

Let v_0 be the power/ground vertex for the connected undirected graph Γ corresponding to a switch-level circuit. Let T be a spanning tree of the graph Γ. Let every edge e_i has a Boolean variable e_{bi} associated with it. We make the following observation.

Observation 3: *Traversal through the spanning tree T starting from v_0 enumerates an active Boolean path for every edge e and every vertex v in the tree.*

Theorem 2: *If F_v is an active Boolean path at a vertex v, then the link $e_i = (v, w)$ contributes to the Boolean path at vertex w if w is not traversed already in enumerating the Boolean path F_v.*

Proof: Naturally, if F_v is an active Boolean path at v, then a potential active Boolean path at w is given by $F'_w = e_{bi} F_v$. Suppose the vertex w is already traversed in enumerating the active Boolean path F_v. Therefore, a cycle is formed by the edge (v , w). Clearly, there already exists an active Boolean path $F_w \subset F_v$ that sensitizes the node w. In other words, $F_w \subset F'_w$ holds and therefore, F'_w does not form an active Boolean path at node w.

The proof of Theorem 2 is completed. □

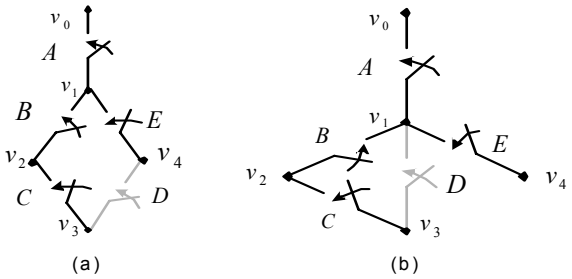


Figure 7: Switch level network graphs illustrating Theorem 2. The lightly shaded branches indicate the links while the rest form the spanning tree. (a) The edge (v_3 , v_4) adds active Boolean paths at nodes v_4, v_3 and v_2. No active Boolean path is added to node v1 (b) The edge (v_3 , v_1) adds active Boolean paths at nodes *v3* and *v2* but not at node *v1* due to the formation of a cycle.

The switch-level circuit graph examples in Figure 7 further illustrate Theorem 2. For the circuit graph in Figure 7(a), following Observation 3, the active Boolean paths at the various nodes obtained by traversing the spanning tree are: *A* (at node v_1), *AB* (at node v_2), *ABC* (at node v_3) and *AE* (at node v_4). According to Theorem 2, the link edge (v_3 , v_4) adds new active Boolean paths *ABCD* (at node v_4), *AED* (at node v_3) and *AEDC* (at node v_2). However, the potential active Boolean paths *ABCDE* and *AEDCB* are not added to node v_1 due to the formation of cycles. In the circuit graph of Figure 7(b), a traversal through the spanning tree creates the following active Boolean paths: *A* (at node v_1), *AB* (at node v_2), *ABC* (at node v_3) and *AE* (at node v_4). The edge (v_3 , v_1) adds an active Boolean path *AD* (at node v_3) and *ADC* (at node v_2). However, the potential active Boolean paths *ABCD* and *ADCB* at node v_1 are not added because of the formation of a cycle.

Based on Observation 3 and Theorem 2, we develop an algorithm to systematically enumerate the sensitization conditions at every vertex and edge of the graph corresponding to a switch-level circuit. This is described in the procedure *List_boolean_conditions* presented in Table 2.

First, a spanning tree is identified by a call to the procedure *create_spanning_tree*. Next, a walk along the spanning tree from the power/ground vertex enumerates the active Boolean paths at the vertices and the edges of the spanning tree. This is accomplished by calling the procedure *sensitization_along_tree*. The Boolean sensitization conditions at the edges and the vertices of the spanning tree are stored as decision diagrams [5][6]. For this, we extend the MTDDDs [14] to handle Boolean algebra apart from regular algebra.

Table 2: Algorithm for enumerating sensitization Conditions.

List_Boolean_Conditions (*Graph* $\Gamma(V, E)$)

begin

$T(V, E') =$ ***create_spanning_tree*** (Γ) // *identifies a spanning tree*

sensitization_along_tree (T) //*creates decision diagram at* (V, E') *during walk through T*

for (1)

$count = 0$

for ($e = (v, w) \mid e \in E - E'$) // *foreach link branch*

$F_v = v.boolean\_function$ // *Boolean function associated with vertex v*

$F_w = w.boolean\_function$

$update\_v =$ ***update_sensitization_along_tree*** ($v, e_b \wedge F_w, e$)

$update\_w =$ ***update_sensitization_along_tree*** ($w, e_b \wedge F_v, e$)

if ($update\_v \vee update\_w$) ***then***

$count = count + 1$

end if

end for

if ($count == 0$) ***then*** //*no update in one pass*

break

end if

end for

end procedure

update_sensitization_along_tree (*vertex v*, *Boolean Function F*, *edge e*)

begin

$found\_loop = new\_update = FALSE$

for ($e' = (v, w) \mid e' \neq e, e' \in E$) // *for all edges at v except e*

// *check_edge_function looks down the decision diagram of F to find a loop formed by* e'_b

$found\_loop = found\_loop \vee$ ***check_edge_function*** (e'_b , F)

if ($found\_loop == TRUE$) ***then***

return FALSE

end if

end for

$v.function = v.function \vee F$

for ($e' = (v, w) \mid e' \neq e, e' \in E'$) // *bfs through tree branches*

$new\_update = new\_update \vee$ ***update_sensitization_along_tree*** ($w, e' \wedge F, e'$)

end for

return new_update

end procedure

The rest of the algorithm follows an iterative scheme (outer loop). From Theorem 2, it is clear that new active Boolean paths at each edge or vertex can be enumerated due to the link edges in the circuit graph. Thus during each iteration of the outer loop, the algorithm enumerates new active Boolean paths triggered by the link edges. As illustrated in Theorem 2 and Figure 7, new active Boolean paths are obtained by *cross-propagating* the active Boolean paths at the vertices incident on the link edge. This is accomplished by calls to the procedure *update_sensitization_along_tree*. In accordance to Theorem 2, if the active Boolean path forms a cycle, it is discarded. Active Boolean paths enumerated in an iteration can trigger enumeration of newer active Boolean paths in the next iteration. The algorithm terminates if no new active Boolean path is enumeration during one iteration through the outer loop.

The recursive procedure *update_sensitization_along_tree* first checks if the active Boolean path under consideration forms a cycle. This is accomplished by a call to the procedure *check_edge_function*. This involves a traversal down the decision diagram of the active Boolean path searching the presence of the specified Boolean variable in the product term. The recursive routine *update_sensitization_along_tree* propagates the active Boolean paths along the spanning tree until a cycle or a power/ground vertex is reached.

The progression of the algorithm for the example circuit graph of Figure 8 is presented in Table 3. In Figure 8, the dark and lightly shaded lines represent the tree and link branches respectively. Each edge has a Boolean variable associated with it. The vertex v_1 is power/ground node. First, a traversal through the spanning tree from the vertex v_1 sets up the active Boolean paths at every vertex (Table 3, column 2). Next, the active Boolean paths due to the link C at vertices v_3 and v_4 and down edge B to vertex v_2 are added (Table 3, column 3). The active Boolean paths DCB, DC and ABC at the vertices v_2, v_3 and v_4 respectively trigger further enumeration. Next, due to the edge E, active Boolean paths DCE and AE are added at vertices v_2 and v_3 respectively (Table 3, column 4). In the next iteration, the active Boolean path AEC is added at vertex v_4 due to edge C. After this, the algorithm terminates as there are no valid new active Boolean paths at any of the vertices.

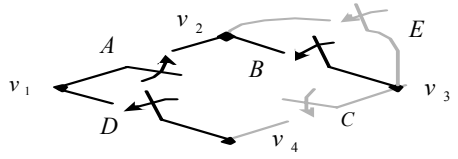


Figure 8: An example circuit graph for illustrating the progression of the algorithm for enumerating sensitization.

Table 3: Sensitization conditions at each node of the graph of Figure 8.

Node #	Tree Setup	Due to *C*	Due to *E*	Due to *C*
v_2	*A*	*DCB*	*DCE*	-
v_3	*AB*	*DC*	*AE*	-
v_4	*D*	*ABC*	-	*AEC*

6. MTDDDs for Boolean-ized Moments

The theory introduced in the previous two sections is applied in the construction of the MTDDDs for the moments of logic stages and interconnects driven by the stages. The procedure *Create_MTDDD* shown in Table 4 describes the basic steps involved in setting up the MTDDD structures.

Table 4: Algorithm for MTDDD construction

```
Create_MTDDD ( G , Max_moment_order )
begin
    // G is the moment matrix,  T(E,V) circuit graph
    List_Boolean_Conditions ( Graph T(E,V) )
    for ( i = 1 to Max_moment_order )
    begin
        moment [i,j ] = NULL
      for ( j : capacitive node)
       begin
          for ( k : capacitive node)
          begin
              P = cofactor( G - {j,k } )
            if ( G ( j,k ) is SYMBOLIC) then
                  Q = getvertex ( G ( j,k ) , P )
            else
                  Q = multiply ( G ( j,k ) , P )
            end if
          end for
          moment [i,j ] = union ( moment [i,j ] , Q )
       end for
    end for
end procedure
```

First, the call to the procedure *List_Boolean_Conditions,* described in Table 2, implicitly stamps the moment matrix with the transformations suggested in Theorem 1. In the MTDDD construction algorithm, the number of iterations through the outer loop is decided by the highest order of moment being computed. For the delay metric in Eq. (10), we calculate moments up to the 2nd order. During iteration through the intermediate loop, moments of the i^{th} order are computed at each node of interest. The MTDDD structures for moments are set up for all nodes

with capacitive elements. The construction of the moment MTDDD of the i^{th} order at the j^{th} node is complete after the entire set of iterations through the innermost loop.

The operation *getvertex(top,D_1,D_2)* generates an MTDDD vertex for the element *top* with the MTDDD sub-graphs rooted at D_1 and D_2 as its 1-child and 0-child respectively. The procedure *cofactor(G-{j,k})* returns an MTDDD vertex representing the cofactor of the matrix *G* with respect to the element at *{j,k}*. The operation *multiply(G{j,k},P)* returns an MTDDD vertex corresponding to the multiplication of the MTDDD *P* with the element *{j,k}* in matrix *G*. The operation *union(P,Q)* returns the algebraic sum of the two MTDDDs *P* and *Q*. All these basic MTDDD operations *getvertex, cofactor*, *multiply* and *union* are described in [14][16]. These have been modified to incorporate the simplification that may be required due to the presence of Boolean and numeric elements. In the MTDDDs constructed, the algebraic and Boolean symbols occupy places closer to the root while the numeric elements are pushed down to the leaf terminals.

The MTDDD for the first moments at *node 3* of our example circuit of Figure 1 is illustrated in Figure 9. Though our delay metric includes the first two moments, we show only the MTDDD graph of the 1^{st} moment for simplicity. Furthermore, for the same reason, a section of the MTDDD is represented in a black-box form. The MTDDD comprises both Boolean and algebraic symbols. One can visualize the algebraic sections hanging from the Boolean portion of the tree. The algebraic section hanging from a given Boolean portion of the tree indicates the moment expression associated with the Boolean function. Here, the expression $R_{m1}R_{m2}(C_2+C_3)+R_{m1}R_{m3}C_3$ is the numerator of the 1st moment at node 3 under the Boolean condition $\overline{A}\wedge\overline{B}\wedge\overline{C}$. Also shown are the algebraic expression $R_{m1}(C_2+C_3)+R_{m3}C_2$ and the corresponding Boolean function $\overline{A}\wedge B\wedge\overline{C}$. Even for this simple MTDDD, one can observe some sharing of subgraphs as indicated by multiple edges pointing to a node. The gray lines between the black-box and the rest of the MTDDD indicate such sharing. For the MTDDDs of the 2nd moments, there is more pronounced sharing of subgraphs.

So far, we have illustrated our symbolic analysis scheme and MTDDD construction for standard CMOS logic circuits. Here it needs to be pointed that the same analysis scheme is directly applicable to different MOS logic families like ratio-ed or dynamic circuits. Circuits in these logic families can be similarly expressed as a switched linear network. Once a circuit is converted into its switched linear network form, the formulation of compact circuit/moment equations and construction of moment/delay MTDDDs can be accomplished similarly.

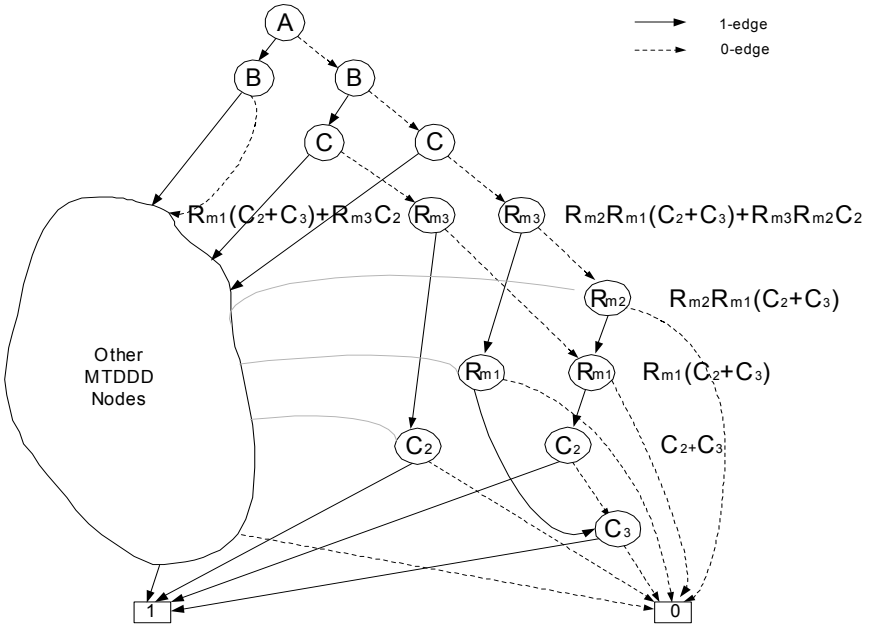


Figure 9: MTDDD for the numerators of the 1st moments at the node 3 for the example circuit of Figure 1. The path from *A* to *B* down to *C* along the *0-edge* indicate the Boolean function $\overline{A} \wedge \overline{B} \wedge \overline{C}$. The associated algebraic expression is $R_{m1}R_{m2}(C_2 + C_3) + R_{m1}R_{m3}C_3$. The algebraic expression $R_{m1}(C_2 + C_3) + R_{m3}C_2$ associated with the Boolean function $\overline{A} \wedge B \wedge \overline{C}$ is also shown. The gray lines indicate subgraph-sharing between the elaborated and the black-box parts of the MTDDD.

Numeric computation of moments involves simple traversals through the MTDDD trees [16]. The sharing of sub-graphs in the MTDDD trees enables efficient numeric computation of moments, as sub-graphs evaluated once need not be traversed again during the entire moment calculation process. Efficient hashing and caching schemes described in [16] enable the sharing of subgraphs.

At this point, we digress briefly to contrast our method of symbolic analysis with the scheme proposed in [10]. While the "symbols" in [10] comprise of only Boolean variables, the "symbols" in our method encompass the spectrum of real valued algebraic variables in addition to Boolean variables. The advantage of our scheme lies in the ability to generate accurate closed-form expressions useful in various modeling, optimization and verification methodologies. Generation of algebraic expressions for MTDDD structures, similar to their numeric computation, involves traversals down the MTDDD graph. As must be obvious from the various MTDDD examples in Figure 3, Figure 4 and Figure 9, the generation of expression for an MTDDD tree is rather trivial. The algorithm is explained in detail in [16].

7. Effective Resistance Calculation

The entire process of generation of closed-form delay expressions assumes an accurate estimation of effective resistances of MOS transistors. The accuracy and, therefore, the efficacy of our symbolic delay estimation method is strongly dependent on the precision of the effective resistance model. This section describes a new topology-dependent effective resistance calculation scheme.

Traditionally, the effective resistance computation schemes employ extensive simulation to build lookup tables [12]. Output transition delay is obtained by simulation for different transistor sizes, input signal slopes and load capacitances. Only one input of the logic-stage is assumed to be switching and the rest of the inputs are assumed to have fixed digital high or low value. A lookup table of equivalent resistance is obtained by equating the estimated RC delay to the transition delay obtained through simulation[12][13]. Unfortunately, these methods of effective resistance calculation ignore the impact of the position of the switching transistor in a channel-connected structure resulting in gross inaccuracies in delay estimation.

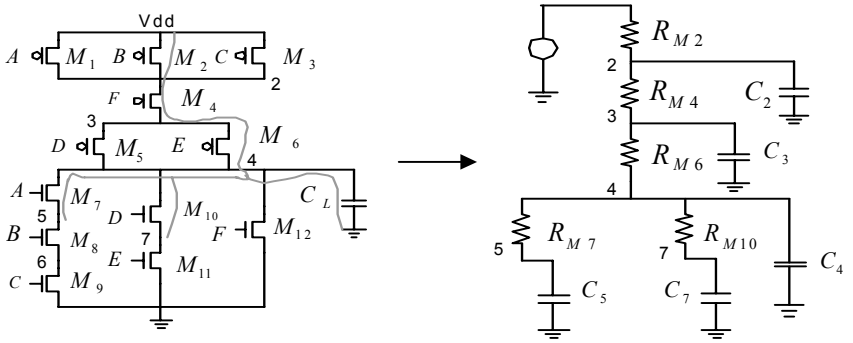


Figure 10: An AOI321 cell and its equivalent circuit under a specific sensitization. The sensitized part of the circuit is shown with the gray lines. Transistor M_4 is the switching transistor.

For this work, we adopt the original table-lookup method incorporating the dependence on the position of the switching transistor in the network. Consider the AOI321 circuit and its switch-resistor equivalent in Figure 10 with the sensitized path shown with the gray line. Even for the same sensitized path, different transition delay results from switching events at different transistors. Thus, the transition delay for *F* switching from *logic-1* to *logic-0* will be different from the delay if *E* switches from *logic-1* to *logic-0*. Eq. (24) provides the expression for transition delay at node *4* under the Elmore model [8] with no assumption about the switching transistor.

$$T_d = R_{M2}(C_2 + C_3 + C_4 + C_5 + C_7) + R_{M4}(C_3 + C_4 + C_5 + C_7) + R_{M6}(C_4 + C_5 + C_7) + R_{M7}C_5 + R_{M10}C_7 \quad (25)$$

Naturally, the resistance of the fully-on transistors are just a small fraction of the effective resistance of the switching transistors. If we assume identical width and length for all p-transistors (and n-transistors), and if M_4 is the switching transistor, the expression for transition delay is given as

$$T_d = R_{M4}(5C_P + 7C_N + C_L) + \alpha R_{M4}(11C_P + 18C_N + 2C_L) \quad (26)$$

where, each nodal capacitor is further expressed in terms of C_P and C_N, the drain/source capacitance of p and n transistor respectively, and $\alpha << 1$ is a heuristic factor. The factor α indicates that all transistors other than M_4 are fully-on and their resistance has a much smaller effect on the total delay. If M_6 is assumed to the switching transistor, the expression for transition delay is given as

$$T_d = R_{M6}(2C_P + 7C_N + C_L) + \alpha R_{M6}(14C_P + 18C_N + 2C_L) \quad (27)$$

The expressions in Eqs. (26) and (27) are equated to the transition delay obtained through simulation to obtain the effective resistance of the transistors M_4 and M_6 respectively. Effective resistance calculation by this method results in markedly good accuracy in our delay estimation method. However, this superior accuracy comes at the price of increased amount of pre-simulation.

Though the method is explained with the Elmore model, we actually use the delay metric of Eq. (10) for its superior accuracy. Curve-fitting methods are employed to express the effective resistances as posynomial functions of transistor widths, lengths, load and input slew rate. The parasitic drain/source capacitors are modeled as simple linear functions of the width and length of the transistors.

8. Experimental Results

The proposed symbolic delay estimation scheme has been implemented in a program called *SAMBA* (Symbolic Analysis Mixing Boolean and Algebra). The SAMBA framework is applied for generating analytic closed form expressions of delay valid under any input signal combination and parameter variation. The accuracy of the delay estimated from the analytic expressions is tested over a variety of circuits.

8.1 Delay Estimation for Single Logic Stage with Interconnects

We apply our method of delay computation on a TSMC 0.18micron standard cell library comprising of various *invertor, nand, nor, aoi, oai* logic stages with different number of inputs (up to six). In these tests, the logic cells drive large RC tree networks. Experiments are conducted with different p-transistor and n-transistor sizes, loads, input patterns and input signal slopes. For all the cases, the maximum error is less than 5% when the estimated delays are compared to delays obtained through HSPICE [9] simulations. Here, we present the results for a 6-input *aoi321* cell and an RC interconnect tree driven by it as shown in Figure 11. The estimated delay in SAMBA is compared with HSPICE under parameter variation. The results indicate excellent match with HSPICE delay values and are shown in Figure 12.

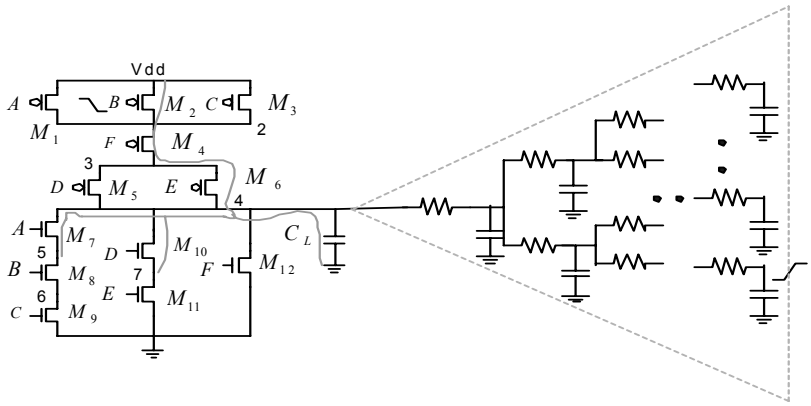


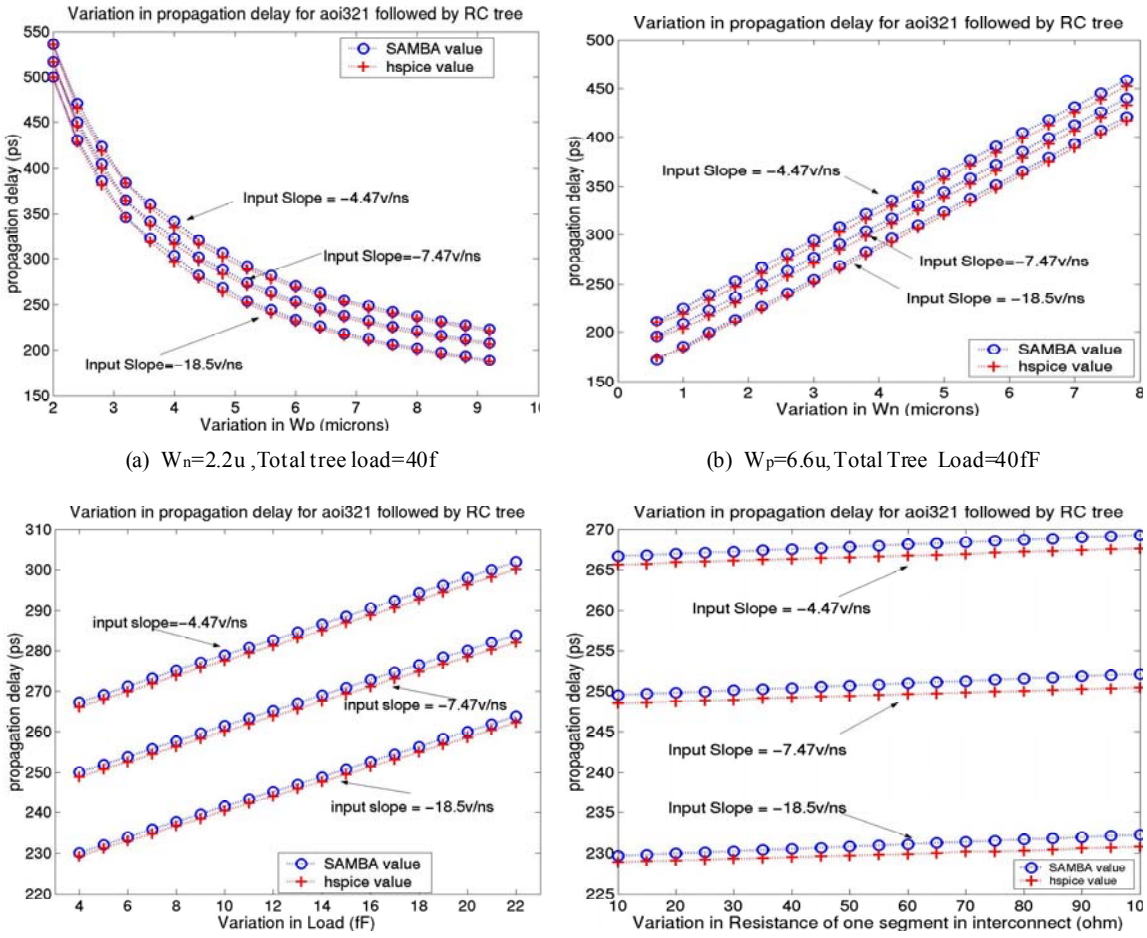
Figure 11: An aoi321 stage followed by a large RC tree. The input *B* at the gate of M_2 changes from *logic-1* to *logic-0*. The sensitized part of the logic stage is indicated by the gray line.

8.2 MTDDD Statistics

Table 5 presents the MTDDD statistics for standard cell circuits. For each standard cell, delay MTDDDs are constructed for the 1st moment at every node in the cell. The size of the MTDDDs for a given cell is the total number of nodes in the MTDDD graphs. Once the MTDDDs for the 1st moments at all the nodes are constructed, higher order moments can be computed by repeated traversal of the MTDDD graphs.

Table 5: Delay MTDDD sizes for Standard Cells.

Cell	# MTDDD Nodes	Cell	# MTDDD Nodes
INV	4	AOI22 / OAI22	148
NAND2 / NOR2	30	AOI221 / OAI221	260
NAND3 / NOR3	55	AOI32 / OAI32	285
NAND4 / NOR4	84	AOI321 / OAI321	325
AOI21 / OAI21	77	AOI33 / OAI33	383



(a) W_n=2.2u ,Total tree load=40f

(b) W_p=6.6u, Total Tree Load=40fF

(c) W_n=2.2u , W_p=6.6u

(d) W_n=2.2u, W_p=6.6u, Load=40fF

Figure 12: Rise delay variation in SAMBA and HSPICE for the circuit of Figure 11. The four cases are : (a) variation in the width of p-transistors, (b) variation in the widths of the n-transistors, (c) variation in the load capacitance at one interconnect sink, (d) variation in resistance in the first section of the interconnect. For each case, we use three different input signal slopes.

8.3 Delay Estimation for Series Connected Logic Stages

The symbolic timing macro-models of standard cell circuits obtained by SAMBA are applied for signal delay estimation in combinational logic blocks. For an industrial ASIC circuit in 0.18um TSMC technology, the signal delay estimated from the macro-models in the combinational blocks ranged within 10% of HSPICE computed delays. The comparison of the delay estimation at various nodes in one such block is presented in Figure 13. Given that the macro-models are generated for single-signal transition and the results are compared to that of transistor-level circuit simulation, the estimations present good accuracy and indicate the potential for incorporation of these macro-models in transistor sizing and timing analysis engines.

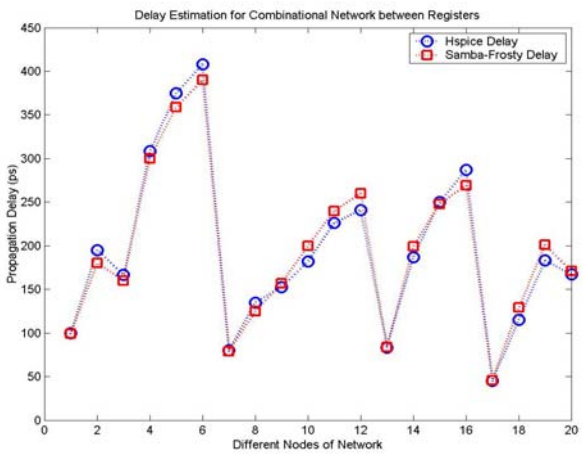


Figure 13: Comparison between SAMBA and HSPICE estimated signal delay at various nodes in a combinational logic block.

9. CONCLUSIONS

A new theory for unified symbolic analysis of switch-level networks is presented here. We treat the signals at the gates of transistors as Boolean variables and the design parameters of transistors and interconnects as algebraic variables. With this, we present the theory for mixed algebraic and Boolean symbolic analysis of switch level networks. We propose a series of transformations for compact representation of the circuit equations for switch level networks. The transformations proposed are general enough to allow extensions of conventional switch level networks to include other common circuit elements. We leverage MTDDDs to enable efficient and compact symbolic representation and solution of switch level network equations.

We apply this theory for symbolic estimation of signal delay in logic gates followed by interconnects. We obtain closed form analytic expressions for delay in terms of design parameters and input signal variables. The delay expressions are found to be very accurate for extensive parameter variation and under any input logic pattern. Our delay estimates in the worst case are within 5% of the HSPICE computed delay values for logic stages including interconnects. Even for series of logic stages in a combinational circuit, the delay estimates are found to be within 10% of HSPICE computed delays.

The symbolic timing models generated by analyzing switch-level logic stages has several applications in VLSI. Accurate analytic delay models are useful in concurrent transistor and interconnect sizing schemes. Fully symbolic models for delay can be directly applied for robust verification under process, voltage and temperature variations. Given the known pitfalls with variability and the importance of aggressive optimization in today's designs, behavioral

simulation and modeling, timing analysis engines can be extended to handle parametric delay models for faster design/timing convergence.

Acknowledgements: The authors thank Dr. Alicia Manthe for her assistance on transistor effective resistance extraction and Lei Yang for his assistance on the application of this theory to MOS digital circuit simulation.

REFERENCES

[1]. C. J. Alpert, A. Devgan and C. V. Kashyap, "RC delay metrics for performance optimization", *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 571-582, May 2001.

[2]. S. Bhattacharya and C-J. R. Shi, "Concurrent logic and interconnect delay estimation of MOS circuits by mixed algebraic and Boolean symbolic analysis", *Proc. IEEE Int. Symp. on Circuits and Systems*, May 2003, pp. 660-663.

[3]. C. J. Terman, *Simulation Tools for Digital LSI Design,* Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, Oct. 1983.

[4]. R. E. Bryant, "A switch-level model and simulator for MOS digital systems", *IEEE Trans. Computers*, vol. 33, pp. 160-177, Feb. 1984.

[5]. R. E. Bryant, "Graph-based algorithms for Boolean function manipulation", *IEEE Trans. Computers*, vol. 35, pp. 677-691, Aug. 1986.

[6]. R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo and F. Somenzi, "Algebraic decision diagrams and their applications", *Proc. IEEE/ACM Int. Conf. on Computer Aided Design,* Nov. 1993, pp. 188-191.

[7]. M. P. Desai and Y. T. Yen, "A systematic technique for verifying critical path delays in a 300MHz alpha CPU design using circuit simulation", *Proc. IEEE/ACM Design Automation Conf.*, Jun. 1993, pp. 125-130.

[8]. W. C. Elmore, "The transient response of damped linear networks", *Jour. Applied Physics*, vol. 19, pp. 55-63, Jan. 1948.

[9]. HSPICE User's Manual, Synopsys Inc., Aug. 2002.

[10]. C. B. McDonald and R. E. Bryant, "Computing logic-stage delays using circuit simulation and symbolic elmore analysis", *Proc. IEEE/ACM Design Automation Conf.,* Jun. 2001, pp. 283-288.

[11]. N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI design*, New York: Addison Wesley, 2nd Edition, 1993.

[12]. J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI", *IEEE Trans. Computer-Aided Design*, vol. 4, pp. 336-349, Jul. 1985.

[13]. L. M. Brocco, S. P. McCormick and J. Allen, "Macromodeling CMOS circuits for timing simulation", *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 1237-1249, Dec. 1988.

[14]. T. Pi and C.-J. R. Shi, "Multi-terminal determinant decision diagrams: A new approach to semi-symbolic analysis of analog integrated circuits", *Proc. IEEE/ACM Design Automation Conf.,* Jun. 2000, pp. 19-22.

[15]. L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis", *IEEE Trans. Computer-Aided Design,* vol. 9, pp. 352-366, Apr. 1990.

[16]. C.-J. R. Shi and X-D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams", *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1-18, Jan. 2000.

[17]. I. N. Hajj and D. Saab, "Symbolic logic simulation of MOS circuits", *Proc. IEEE Int. Symp. on Circuits and Systems*, May 1983, pp. 246-249.

[18]. R. E. Bryant, "Algorithmic aspects of symbolic switch network analysis", *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 618-633, Jul. 1987.

[19]. S. Tsukiyama, I. Shirakawa, H. Ozaki and H. Ariyoshi, "An algorithm to enumerate all cutsets of a graph in linear time per cutset", *Jour. Association for Computing Machinery*, vol. 27, pp. 619 – 632, Oct. 1980.

[20]. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, New York: Van Nostrand Reinhold, 1983.

[21]. J. Rubenstein, P. Penfield, Jr., and M. A. Horowitz, "Signal delay in RC tree networks", *IEEE Trans. Computer-Aided Design*, vol. 2, pp. 202-211, Jul. 1983.

[22]. C.-J. Shi and K. Zhang, "A robust approach to timing verification", *Proc. Int. Conf. on Computer-Aided Design,* Nov. 1987, pp. 56-59.

[23]. B. Tutuianu, F. Dartu and L. T. Pileggi, "An explicit RC-circuit delay approximation based on the first three moments of the impulse response", *Proc. IEEE/ACM Design Automation Conf.*, Jun. 1996, pp. 611-616.

[24]. T. Lin, E. Acar and L. T. Pileggi, "h-gamma: An RC delay metric based on a gamma distribution approximation to the homogeneous response", *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design,* Nov. 1998, pp. 19-25.

[25]. A. Salz and M. A. Horowitz, "IRSIM: An incremental MOS switch-level simulator", *Proc. IEEE/ACM Design Automation Conf.*, Jun. 1989, pp. 173-178.

[26]. K-J. Lee, C. A. Njinda and M. A. Breuer, "SWiTEST: A switch level test generation system for CMOS combinational circuits", *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 625-637, May 1994.

[27]. R. E. Bryant, "Boolean analysis of MOS circuits", *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 634-649, Jul. 1987.

[28]. V. Sundararajan, S. S. Sapatnekar and K. K. Parhi, "Fast and exact transistor sizing based on iterative relaxation", *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 568-581, May 2002.

[29]. N. Menezes, S. Pullela and L. T. Pileggi, "Simultaneous gate and interconnect sizing for circuit-level delay optimization", *Proc. IEEE/ACM Design Automation Conf.*, Jun. 1995, pp. 690-695.

[30]. S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi and V. De, "Parameter variation and impact on circuits and microarchitecture", *Proc. IEEE/ACM Design Automation Conf.*, Jun. 2003, pp. 338-342.

[31]. J. G. Broida and S. G. Williamson, *A comprehensive introduction to linear algebra*. Reading, MA: Addison Wesley, 1989.