Recursive Algorithms Based on MTB Tree for Estimating Delays in RC Tree Circuits with considering Charge Sharing Effect

Molin Chang, Shuih-Jong Yih and Wu-Shiung Feng

Department of Electrical Engineering, R244
National Taiwan University
Taipei, Taiwan, R.O.C. China

Abstract

Internal charges will affect the delay behavior of a CMOS gate. A Modified Threaded Binary (MTB) tree and two recursive algorithms, ECS and EER, are proposed to solve this problem. The charge sharing effect, which takes place among the internal nodes, is also considered in the algorithm ECS.

1. Introduction

The switch-level simulation is a compromised method between circuit-level and logic-level simulations. It has the advantage of fast speed and good adaptability for VLSI circuit. Some switch-level simulators have been implemented, such as MOSSIM, RSIM [1], [2]. BTS (Binary-tree Timing Simulator) is a switch-level timing simulator based on series-parallel circuit structure [3], which is two or three orders faster than SPICE and has more accurate waveform approximation during the transient period.

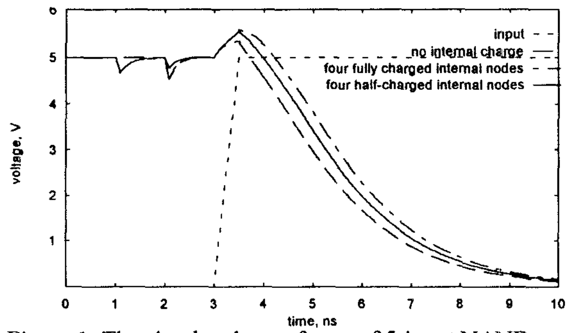

Figure 1: The simulated waveforms of 5-input NAND gate using SPICE.

The internal charges in a CMOS gate will evidently increase the delay. For example, there is a 25% increment of delay for a five-input NAND gate with four fully charged internal nodes [3]. Therefore, we can not neglect the delay affected by the charges stored in internal nodes. Futhermore, the charge sharing effect acting on the internal nodes will make the calculation work more difficult because the different amount of internal charge leads to different amount of delay. Fig. 1 shows the delay times of 5-input NAND gate due to the different amount of internal charges. Because of this, we propose two recursive algorithms based on MTB tree to solve this problem.

2. Series-Parallel Tree

If a gate is connected by series-parallel way such as fully complementary CMOS, Pseudo-NMOS, Dynamic CMOS and so on, it can be represented as a merged series-parallel tree, also called a *PN tree*. Fig. 2(b) illustrates the corresponding PN tree of the gate circuit shown in Fig. 2(a) which function is $Z=((a+((b+c)*(d+e)))+f)\#((a*((b*c)+(d*e)))*f)$, where $*$'s and $+$'s represent AND and OR operations, respectively. There are two logical expressions in the equation; the former is the function of P tree and the latter is of N tree. The OR nodes in the PN tree are virtual nodes because an OR node only parallels a subcircuit to another and does not create any real node in a gate circuit. For example, if the nM_d (the NMOS transistor that accepts the input d) is removed, the '+' is removed simultaneously but no internal node will be removed. Many troublesome problems, such as the evaluation of output state, equivalent RC time constant [3], [4], [5], [6] and internal charge effect, can be solved easily and efficiently by using a series-parallel tree to represent the circuits with the design styles as mentioned above.

3. MTB Tree

An MTB tree is also a series-parallel tree, but the threads of a leaf node are always pointed to AND nodes (denoted by $v*$). Because OR nodes (denoted by v^+) are virtual nodes, they must be excluded while establishing the threads of each leaf node. The idea of this new structure is modified from the threaded binary tree [8]; the difference is that the threads of any leaf node are always pointed to AND nodes. In the algorithm ECS that is proposed to solve the problem of charge sharing effect, we need to determine the attribute of each internal node, and probably need to merge two isolated groups or create a new isolated group while visiting the active (*turn-on*) leaf node in the PN tree. The MTB tree can let us find the adjacent internal nodes of any leaf node directly through its two threads, and thus largely decrease the search time. Fig. 2(b) shows an MTB tree corresponding to the circuit as shown in Fig. 2(a).

4. Delay Estimation

BTS is a waveform-based event-driven timing simulator and the primary goal is to obtain an accurate

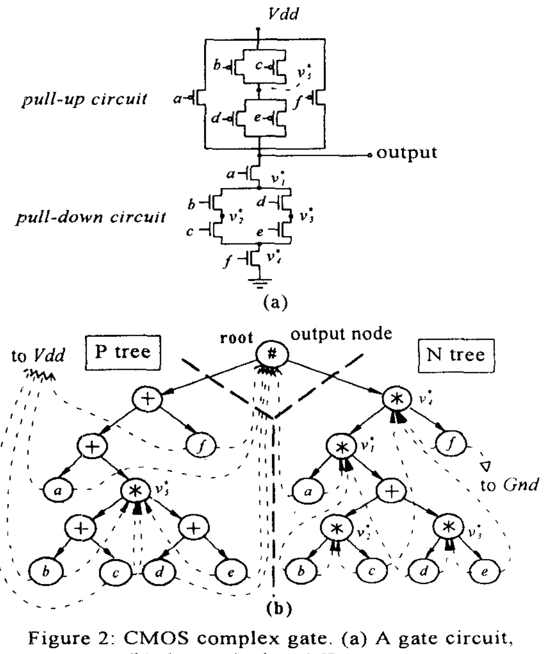

Figure 2: CMOS complex gate. (a) A gate circuit, (b) the equivalent MTB tree.

waveform [3]. In order to achieve this goal, we adopt the following equations to approximate the waveforms during the transient state [7], and consider the internal charges effect including the charge sharing effect while estimating the delay.

$$f = \begin{cases} \frac{0.2t}{T} & t < 3T \\ 1 - 0.4\exp(-\frac{t-3T}{2T}) & t \geq 3T \end{cases} \quad \textit{for a rising signal} \tag{1}$$

$$f = \begin{cases} 1 - \frac{0.2t}{T} & t < 3T \\ 0.4\exp(-\frac{t-3T}{2T}) & t \geq 3T \end{cases} \quad \textit{for a falling signal} \tag{2}$$

where T is half of the time spent by signal between 90% (for a falling signal) or 10% (for a rising signal) and 50% of the steady state. If the value of T can be obtained, the transient waveform will then be easily plotted. Hereafter, we only take the falling signal as example, that is, only the N tree is discussed because of the identical treatment of both.

The delay due to the internal charges can be calculated approximately as

$$\Delta t = \frac{Q}{\overline{I}} = \frac{Q}{V_s} 2R = \frac{\int C dV}{V_s} 2R \tag{3}$$

where $\overline{I}$ is the average current, V_s is the voltage swing, R is the effective resistance of the conducting path and Q is the charges stored in the internal nodes. More than one internal nodes may be going to charge or discharge in the PN tree, and these nodes must be taken into account when calculating the switching delay. Thus, Eq. 3 is rewritten as

$$D_i = \sum_i 2R_i \frac{Q_i}{V_s} \tag{4}$$

where R_i is the effective resistance of internal node v_i^* with respect to ground, Q_i is the charges stored in the internal node v_i^*. For example, if let $V_a=V_c=V_d=V_f=5V$, $V_b=0V$, and $V_e=0V$ to $5V$ for the circuit as shown in Fig. 2(a), then the internal nodes v_1^* and v_3^* are considered because only both nodes have the charges to be discharged. The total delay time caused by the internal charges is $\Delta t = \Delta t_{v_1^*} + \Delta t_{v_3^*}$, where

$$\Delta t_{v_1^*} = 2R_{v_1^*} \frac{Q_{v_1^*}}{V_s} = 2(R_d + R_e + R_f)\frac{Q_{v_1^*}}{V_s}$$

$$\Delta t_{v_3^*} = 2R_{v_3^*} \frac{Q_{v_3^*}}{V_s} = 2(R_e + R_f)\frac{Q_{v_3^*}}{V_s}$$

5. Charge Sharing Effect

The states of internal nodes are dynamically changed after a new event occurs, and the output probably changes at this time. No matter what the output state is, we should recalculate the voltage value of each internal node such that the next event can use them as initial values. We observe that the internal nodes in the tree can be classified into three groups, which are (1) ***C** group (nodes in charged state)*: nodes are connected to output node (the output of this cluster must be in high state), (2) ***D** group (nodes in discharged state)*: nodes are connected to ground (denoted by *Gnd*) node, and (3) ***I** group (nodes in isolated state)*: nodes are connected to neither output node nor *Gnd* node. Fig. 3 can illustrate this concept. These groups maybe exist in the tree simultaneously; it depends on the input pattern and the state of output.

To decide the attribute of each node in the tree, we need to traverse the tree by three passes. Each pass should leave a mark on v^* that is visited such that any node will be dealt with only once. These marks make the charge sharing problem easier to solve in pass3. Therefore, before presenting the algorithm, a set of symbols to mark the v^* will be introduced as follows. ***H*** denotes high state, and means that $v^* \in \boldsymbol{C}$. ***L*** : Output low state ($v^* \in \boldsymbol{D}$). ***G***: Ground state ($v^* \in \boldsymbol{D}$). $\boldsymbol{I_i}$: Isolated state ($v^* \in \boldsymbol{I}$). ***U***: Undetermined state. All symbols are replaced with an unique integer number in BTS except $\boldsymbol{I_i}$, which is changed depending on its group number. We use $M(v^*)$ to represent the mark of v^*, so $M(v^*) \in \{\boldsymbol{H,L,G,I_i,U}\}$. From experimental results, we assume that the delay of the output changed from HIGH to LOW state is affected by the stored charges and the charge sharing effect in the N tree [3]. On the contrary, we consider the P tree when output is changed from LOW to HIGH.

Algorithm ECS: (Estimate Charge Sharing effect)

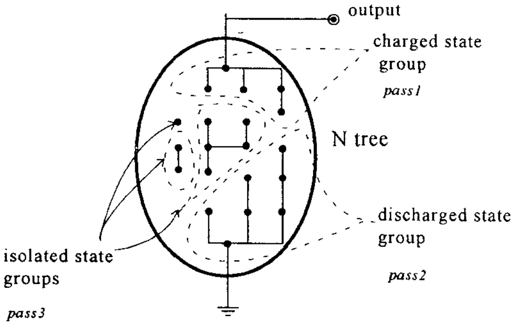

Figure 3 : Three states in the N tree

- •*Pass 1*: Traverse the N tree recursively by forward in-order visiting (i.e., traverse from left to right). In this pass, all nodes that can reach output node through drain-source channel of the turn-on MOS will be sought out and marked by ***H*** or ***L***, depending on the state of output. In other words, the v^*'s $\in \boldsymbol{C}$ can be almost picked out. This pass will be terminated when the state of any subtree is false.
- •*Pass 2*: Traverse the N tree recursively by backward in-order visiting (i.e., traverse from right to left). The purpose is to seek out the nodes that are connected to *Gnd* node, i.e., find out the set $\{v^* | v^* \in \boldsymbol{D}\}$, and also make a mark on them (marked by ***G***). This pass is also not a complete tree traversal, and the condition of termination is the same as pass 1.
- •*Pass 3*: Traverse the N tree recursively by forward pre-order visiting. After pass 1 and pass 2, the most v^*'s $\in \boldsymbol{C}$, ***D*** can be found out, but some special nodes (called *Hidden nodes*, which are the nodes that belong to ***C*** or ***D*** but cannot be sought out in pass 1 or pass 2.) are still not marked. In addition, the isolated nodes ($v^* \in \boldsymbol{I}$) are also left to be processed by this pass. In this pass, the whole tree is traversed but only the leaf nodes are visited. We can find two adjacent internal nodes of the visited leaf node immediately via two threads of the leaf node. If the leaf node is active, then we can determine the attributes of these two internal nodes by Rule 1 described as follows.

Rule 1: Assume V_i is the active leaf node we visited, and then the *lchild* and the *rchild* are used to obtain its two adjacent internal nodes, say v_{il}^* and v_{ir}^*, respectively.

(1) $M(v_{il}^*)=$ ***U*** and $M(v_{ir}^*)=$ ***H, L, G,*** then $M(v_{il}^*)=M(v_{ir}^*)$.

(2) $M(v_{il}^*)=$ ***U*** and $M(v_{ir}^*)=\boldsymbol{I_i}$, then $M(v_{il}^*)=M(v_{ir}^*)$.

(3) $M(v_{il}^*)=$ ***U*** and $M(v_{ir}^*)=$ ***U***, then $M(v_{il}^*)=M(v_{ir}^*)=\boldsymbol{I_i}+1$, i.e., to create a new isolated group that contains these two nodes.

(4) $M(v_{il}^*)= \boldsymbol{I_i}$ and $M(v_{ir}^*)= \boldsymbol{I_j}$, i < j, then $M(v_{ir}^*)=M(v_{il}^*)$ i.e., the isolated group $\boldsymbol{I_j}$ is merged into $\boldsymbol{I_i}$.

(5) $M(v_{il}^*)= \boldsymbol{L}$ and $M(v_{ir}^*)= \boldsymbol{G}$, then $M(v_{ir}^*)= \boldsymbol{L}$.

Note that v_{il}^* and v_{ir}^* can be exchanged each other as described in items (1), (2), and (4). #

We also use the circuit as shown in Fig. 2(a) as an example to demonstrate this algorithm. Three sequential events are applied to the circuit. Event 1: *Va=Vc=Vd=Vf =5V*, and *Vb=Ve=0V*. Pass 1: $M(v_1^*)=M(v_3^*)=\boldsymbol{H}$. Pass 2: $M(v_2^*)=M(v_4^*)=\boldsymbol{G}$. Pass 3: no v^* needs to be processed. Event 2: *Vc=Vd=Ve=5V*, and *Va=Vb=Vf=0V*. Pass 1: no $v^* \in \boldsymbol{C}$. Pass 2: no $v^* \in \boldsymbol{D}$. Pass 3: v_1^*, v_2^*, v_3^* and $v_4^* \in \boldsymbol{I_1}$, and charge sharing effect takes place. In theory, there are four 1/2-charged internal nodes in the N tree. Event 3: *Va* and *Vf* are changed from 0V to 5V. The output begins to discharge, and its delay should be added by the component resulting from the internal charges stored in nodes v_1^*, v_2^*, v_3^* and v_4^* .

6. Effective Resistance

In MTB tree, the nodes except leaf nodes are all operators (i.e., $*$'s and +'s), but only the v^* can become the focused node (means that the node attempts to discharge, denoted by v_f^*) while calculating the effective resistance (R_{eff}) about it. The calculation of R_{eff} with respect to v_f^* is a complicated problem because several paths conducting to *Gnd* node probably exists in the tree. For example, there are two discharging paths for the node v_2^* (in Fig. 2(a)), which are nM_b-nM_d-nM_e-nM_f and nM_c-nM_f.

6. 1 The Reduced-AOG tree

In the process of calculating effective resistance, we consider v_f^* one by one. Beginning from v_f^* and visiting its all ancestors until meeting the root, the visited nodes form an ascendant path, called *backbone* path. For example, $v_1^* \to v_2^* \to v_3^* \to v_1^+ \to v_2^+ \to v_4^* \to v_5^* \to v_6^* \to v_3^+ \to v_4^+$ is the backbone path in Fig. 4(a). It can be observed that the size of gathering group of v^* or v^+ is arbitrary, and v^* and v^+ groups are ordered by interlacing manner in the backbone path. In addition, a v^*-group combining with a v^+-group forms another kind of group naturally, called *AOG* (**AND-OR-G**roup), because v^* cannot merge with v^+. AOGs will exist repeatedly in the tree, and thus an AOG can be treated as a basic unit to be processed while reducing the PN tree.

Definition 1: An *AOG* in the PN tree has the form like $v_i^* \to \cdots \to v_n^* \to v_j^+ \to \cdots \to v_m^+$ in the ascendant order. If $v_i^* = v_f^*$, it's also called *primary-AOG* (*pAOG*); else called *extended-AOG* (*eAOG*). Then, the AOG tree is composed of one *pAOG* and several *eAOG*s.

Definition 2: A *Reduced-primary-AOG* (*RpAOG*) consists of two nodes which are one v^* and one v^+ in the sequence of $v_f^* \to v^+$. A *Reduced-extended-AOG* (*ReAOG*) consists of three internal nodes which are two v^*'s and one v^+ in the sequence of $v_{cr}^* \to v_{cl}^* \to v^+$ or $v_{cl}^* \to v_{cr}^* \to v^+$. Note that v_{cr}^* and v_{cl}^* are the nodes that are visited through their right and left child, respectively. Then, the Reduced-AOG tree is composed of one *RpAOG* and several *ReAOG*s.

Rule 2 : Sequential AND nodes in the MTB tree can be always reduced to *one-v*-subtree* or *two-v*-subtree*, according to the nodes in the *pAOG* or in the *eAOG*. From some focused node, say v_j^*, to visit its all ancestors, the left (right) R_{eff}'s of subtrees of v_{cr}^*'s (v_{cl}^*'s) and v_j^* should be summed up, and denoted by $R_u(R_d)$. After all, if all v^*'s are in the *pAOG*, they will be merged into one v^*. The rule can be summarized by the Eqs. 5 and 6 as follows.

$$R_u = \sum_i r_{t_l}(v_i^*) \cdot \bar{f}(v_i^*),\ \ i = j, j+1, \ldots, \max\ \textit{number of } v^* \textit{ in AOG}. \quad (5)$$

$$R_d = \sum_i r_{t_r}(v_i^*) \cdot f(v_i^*),\ \ i = j, j+1, \ldots, \max\ \textit{number of } v^* \textit{ in AOG}. \quad (6)$$

where $f(v_i^*) = \begin{cases} 0, & \textit{if } v_i^* = v_{cr}^* \\ 1, & \textit{if } v_i^* = v_{cl}^* \end{cases}$, and $r_{t_l}(v_i^*)$ and $r_{t_r}(v_i^*)$ are R_{eff}'s of the left and the right subtree of v_i^*, respectively

Rule 3: Sequential OR nodes in the MTB tree can be always reduced to an *one-v⁺-subtree*. From the lowest node, say v_j^+, to visit its all ancestors, the left (right) R_{eff}'s of subtrees of v_{cr}^+ (v_{cl}^+) and v_j^+ should be summed up, and denoted by $R_r(R_l)$. Note that v_{cr}^+ is the node that is visited through its right child. After all, all v^+'s will be merged into one v^+. The rule can be summarized by the Eq. 7 as below.

$$R_p = R_l \,||\, R_r = \frac{1}{\sum_i \frac{1}{r_t(v_i^+)}} ,\ \ i = j, j+1, \ldots, \max\ \textit{number of } v^+ \textit{ in AOG}. \quad (7)$$

where $r_t(v_i^+)$ is the R_{eff} of the subtree of v_i^+.

According to Rule 2 and Rule 3, a *Reduced-AOG tree* can be reduced from *AOG tree* when all subtrees in the backbone path are replaced with their R_{eff}'s An example to show the *AOG tree* is drawn in Fig. 4(a), the *Reduced-AOG tree* is presented in Fig. 4(b).

6. 2The Reduced-AOG circuit

The *Reduced-AOG circuit* is an equivalent topology circuit of the *Reduced-AOG tree*. The corresponding

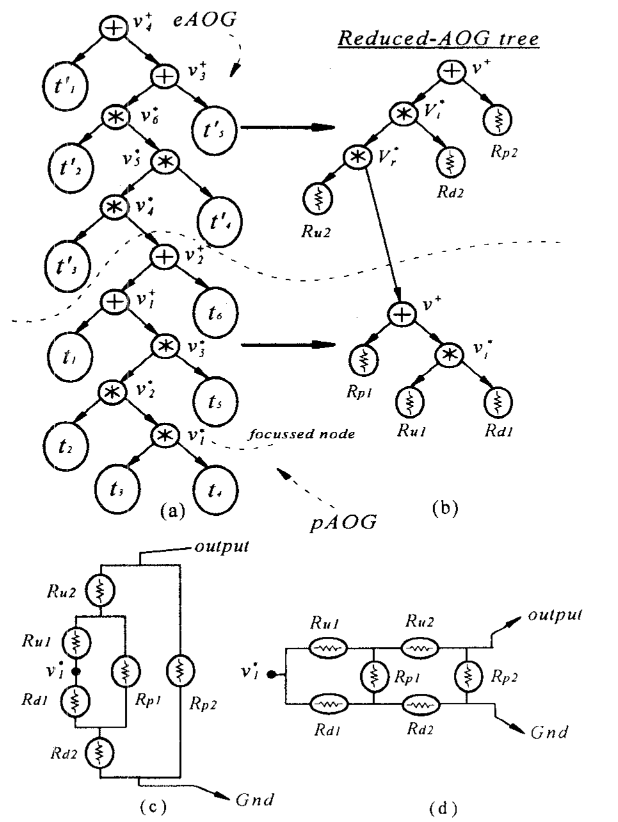

Figure 4: An example for illustrating the reduction of tree. (a) the original PN tree. (b) the Reduced-AOG tree. (c) the Reduecd-AOG circuit. (d) the R-mesh circuit.

Reduced-AOG circuit of Fig. 4(b) as shown in Fig. 4(c) is a topology connection that is expanded with regard to v_f^+. For each R_{ui}, R_{di} and R_{pi}, it could be open-circuit, short-circuit or a finite resistance, and these R_{eff}'s form a block. By arranging the components in the *Reduced-AOG circuit*, we can find that the *Reduced-AOG circuit* is exactly an *R-mesh* circuit. Therefore, translating the tree structure (*Reduced-AOG tree*) to the linked list using AOG as a basic unit (*R-mesh circuit*) can make the computation of R_{eff} easier. After translating the *Reduced-AOG circuit* to the *R-mesh* circuit, we can apply the $Y \rightarrow \Delta$ transformation to the R-mesh circuit and can calculate R_{eff} block by block.

Algorithm EER: (Estimate Effective Resistance)

An *AOG tree* (also a PN tree) can be reduced to a *Reduced-AOG tree* by the following steps.

- *Step 1*: Traverse the tree recursively by in-order manner to select the internal node with charge; of course, leaf nodes and v^+'s are excluded. Starting from the v_f^+, we can use *Rule2* and *Rule3* to reduce the *pAOG* to the *RpAOG*.
- *Step 2*: When visiting to an *eAOG*, also use *Rule2* and *Rule3* to reduce the *eAOG* to the *ReAOG*. We seek out the first v_{cr}^+ and v_{cl}^+ we meet, denoted by V_{cr}^+ and V_{cl}^+ respectively, and then (1) merge all v_{cr}^+ into V_{cr}^+ and all v_{cl}^+ into V_{cl}^+, (2) add all R_{eff}'s of the left subtrees of v_{cr}^+'s to R_{eff} of the left subtree of V_{cr}^+, and then can obtain R_{ui}. The same action is also applied to the node V_{cl}^+ to obtain R_{di}. (3) sum up all R_{eff}'s of v^+'s to obtain R_{pi}.
- *Step 3*: Repeating step2 until meet the root node, we establish a *Reduced-AOG tree.*
- *Step 4*: Translate the *Reduced-AOG tree* to the corresponding *R-mesh circuit*, and then apply the $Y \rightarrow \Delta$ transformation to the *R-mesh circuit*. Therefore, we can calculate R_{eff} block by block.
- *Step 5*: Calculate the delay value. Then find the next v_f^+ and repeat the above steps again. The algorithm is terminated while the whole tree is completely traversed. #

To demonstrate *Algorithm EER*, we use the PN tree that is composed of two *AOG*s as shown in Fig. 4(a), and its *Reduced-AOG tree* is shown in Fig. 4(b). Next, the *Reduced-AOG circuit* and its translated *R-mesh circuit* are shown in Fig. 4(c) and (d), respectively.

7. Simulation results and conclusion

This algorithm has been implemented in BTS. The simulation results of 5-input NAND gate with respect to different amount of internal charges are shown in Fig. 5. Another simulation results of the circuit as sketched in Fig. 2(a) are shown in Fig. 6, and there is an about 70% improvement of the delay time. The complexity of the algorithms grow linearly (*O(n)*) with the size of the cluster, or precisely, with the amount of the internal nodes. The CPU time comparisons are summarized in Table 1, and the speed ratio shows the advantage of BTS, especially for a large-scale circuit. To obtain the more accurate estimation, futher study on the relationship among the delay, the distribution and amount of the internal charges is necessary. This is the future work on BTS development.

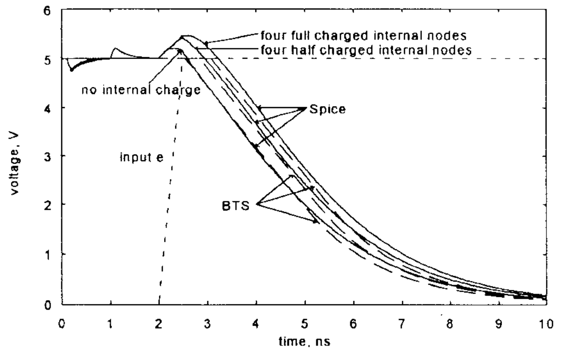

Figure 5: The simulated waveforms of 5-input NAND gate

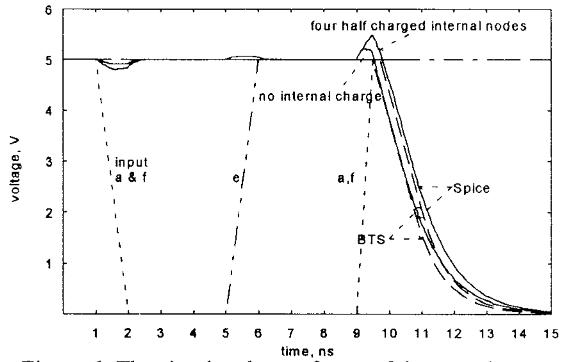

Figure 6: The simulated waveforms of the complex gate of Fig.2(a).

Table 1 : Comparisons between BTS and Spice

Circuit	MOS no.	CPU time on PC (DX4-100), secs BTS	Pspice	Speed ratio	Primary input event no.
complex gate (Fig. 1(a))	12	0.11	2.53	0.043	2
inverter chain (100 stages)	200	0.28	241.18	0.0012	1
74138	88	0.33	102.22	0.0032	11
7483	258	0.99	774.64	0.0013	13
74381	584	1.10	1670.98	0.00066	14

References

[1] R. E. Bryant, "A Switch level model and simulator for MOS digital systems," IEEE Computers, vol. C-33, pp. 160-177, Feb. 1894.

[2] C. J. Terman, "RSIM - A Logic-Level Timing Simulator," Proceedings of the IEEE International Conference on Computer Design, New York, pp. 437-440, November 1983.

[3] J. H. Wang, Molin Chang, and W. S. Feng, "Binary-tree timing simulation with consideration of internal charges," IEE Proceedings-E, vol. 140, No.4, pp.211-219, July 1993.

[4] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," IEEE Trans. on Computer-Aided Design, vol. CAD-2, NO. 3, pp.202-211, July 1983.

[5] T. M. Lin, and C. A. Mead, "Signal delay in general RC networks," IEEE Trans. on Computer-Aided Design, vol. CAD-3, No.4, pp.331-349, October 1984.

[6] J. -P. Caisso, E. Cerny, and N. C. Rumin, "A recursive technique for computing delays in series-parallel MOS transistor circuits," IEEE Trans. on Computer-Aided Design, vol. 10, No.5, pp.589-595, May 1991.

[7] F.C.Chang, C.F.Chen, and P.Subramaniam, "An accurate and efficient gate level delay calculator for MOS circuits," Proceedings of 25th ACM/IEEE conference on Design automation, Anaheim, CA, USA, pp.282-287, June 1988.

[8] T.Horowitz, and S.Sahni, "Fundamentals of Data Structures in Pascal". Computer Science Press, 2nd Edition, Chapter 5, pp.203-271, 1987.