Binary-tree timing simulation with consideration of internal charges

J.J.H. Wang
M. Chang
W.S. Feng

Indexing terms: Logic-level simulation, MOS circuits, Series–parallel RC networks

Abstract: An accurate and efficient block-level timing simulator is described. The high accuracy is attributed to a sophisticated delay model, which includes an accurate representation of the waveform, a consistent and meaningful definition of delay, a consideration of waveform slope effects at both input and output of a gate, a consideration of the multiple charging/discharging paths in the circuit, and a consideration of the various fan-out effect and various cell-size effects. Efficient delay calculation is accomplished through a logic-level simulator instead of using a transistor-level simulator. To represent the waveform accurately, the switching delay and slope are defined and calculated with consideration of the internal charges. To consider the internal charges when computing the waveform, a merged PN tree is used to represent a CMOS gate. The characteristics of the PN tree are described and the methods used to evaluate the conducting paths proposed. The relationship between the *RC* time constant and the slope waveform is investigated. After the conducting paths are obtained, a recursive algorithm can be applied to compute the *RC* time constant in series-parallel *RC* networks, followed by switching delay and slope. The results are satisfactory when compared with Spice.

1 Introduction

A frequently used model for rapidly computing the approximate delay in MOS circuit is the linear resistance–capacitance (*RC*) network with grounded capacitances [1, 2]. Penfield *et al.* [3] presented bounds with a fixed level of accuracy for the delay in *RC* trees. Wyatt [4] showed that these same bounds are also valid for *RC* meshes. The Elmore delay [5], in *RC* trees in particular, can be computed extremely efficiently, as was shown by Lin and Mead with their algorithm Tree [6], although this is only a reasonably good approximation to the true delay. For more general networks, in which the resistors form reconvergent paths, many methods were proposed in the past. Lin and Mead [6] proposed an iterative algorithm which was based on converting the nontree-like *RC* networks to an *RC* tree using node splitting. Chan and Karplus [7] partition the network into a spanning tree and linking branches.

Caisso *et al.* [9] proposed an efficient algorithm for computing the Elmore delay in *RC* networks in which the resistors are interconnected in a series–parallel manner. The algorithm is recursive and has a computational complexity of only $O(ms + ns)$, where $n + 1$ is the number of nodes, m is the number of links, and the delay is computed at s nodes. The algorithm is useful because many MOS VLSI digital circuits are based on design styles, such as fully complementary MOS, in which the transistor channels form series–parallel networks.

Considerable improvement in accuracy can be realised if the effects of the grounded capacitances are considered carefully. The initial values of charges stored in the grounded capacitors in a *RC* network play a very important role when determining the delay and the voltage waveform in a *RC* network. We define 'switching delay' as the difference between the time that output signal begins to change the time that input signal begins to change. We also define 'slope' as the changing rate after the output begins to change. The switching delay consists of the time needed to form the conducting path, and the time needed to charge/discharge the internal capacitors connected to the conducting paths before the output voltage begins to change. So the switching delay of a gate is a function of

(*a*) input signal slope
(*b*) the threshold voltage V_T of MOS
(*c*) charging or discharging paths in the gate
(*d*) the MOS aspect ratio and
(*e*) the internal capacitors.

We want to know not only whether the logic gate changes state or not, but also when the output voltage begins to change and how fast it will change. But most switch-level algorithms emphasised how to calculate the time constant of charging/discharging the load capacitance more accurately. In this paper we present an algorithm for computing the Elmore delay and the voltage waveform in series–parallel networks considering the charges stored in the parasitic capacitors.

2 Effect of internal charge

Each MOSFET in switch-level timing simulations is modelled with a linear resistor between its drain and source terminal, and with a grounded capacitor as its gate terminal. Integrated wires are distributed *RC* lines that are modelled with lumped elements. To embrace the capacitive effects found in MOSFETs, two grounded

Paper 9644E (C2, E3), received 18th January 1993

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, 10764, Republic of China

capacitors are connected to the source and drain terminal, respectively. Charges can be stored in these capacitors. The state transition of a CMOS gate, in fact, is the transference of these charges. We call the nodes connected to V_{DD}/ground/C_{load} as the internal nodes, and the capacitors used to model the drain/source capacitances of MOS transistors as the internal capacitors, and the charges stored in internal capacitors as internal charges. On the other hand, the charge stored in the load capacitor is called an external charge. If the external charges transfer from the load capacitor to ground output voltage decreases. If the charges transfer from V_{DD} to the load capacitor, output voltage increases. But the transference of external charge is not the only factor that affects the switching delay and the waveform slope of a gate; the transference of internal charges also does. Fig. 1 shows the relationship between the high-to-low switching delay of a five-input NAND gate and the number of the internal nodes with nonzero internal charges in the gate. There will be 25% error if the internal charges are neglected when calculating the switching delay.

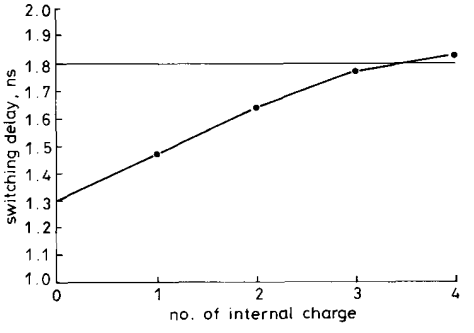

Fig. 1 *Switching delay of a five-input NAND gate against number of internal charges stored in gate (internal charges are stored at capacitors of series-connected NMOS transistors)*

The roles that the internal charges play on computing the switching delay and the waveform slope of a gate can be classified into three cases. Consider the six-input CMOS complex gate as shown in Fig. 2. The initial state

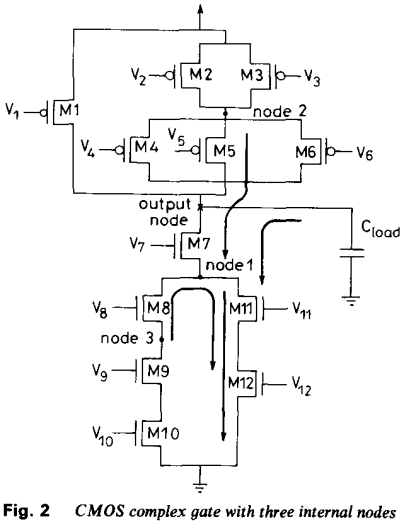

Fig. 2 *CMOS complex gate with three internal nodes*
$V_1 = V_2 = V_4 = V_6 = V_7 = V_8 = V_{10} = V_{12} = 5$ V
$V_5 = V_9 = 0$ V
$V_3 = V_{11} = 0 -> 5$ V

of this gate is high if the initial values of the inputs are as those listed in Fig. 2. Initially, V_3 and V_{11} are 0 V, so the initial state of the output node is high. Meanwhile, paths exist between V_{DD} and node 1, 2 and 3 shown in Fig. 2, so nodes 1, 2 and 3 are also charged to high. When the inputs V_3 and V_{11} increase from 0 to 5 V, V_{DD} is isolated from the output node and there is a discharging path formed from the output node to ground, M7–M11–M12. We denote this path as P_0. Meanwhile, there are three paths formed at the same time that connect nodes 1, 2 and 3 to ground, respectively. We denote these three paths as P_1, P_2 and P_3

P_0: M7–M11–M12

P_1: M11–M12

P_2: M5–M7–M11–M12

P_3: M8–M11–M12

where M_i is the MOSFET that constitutes the path and '–' means that the charges will transfer from the left side of '–' to the right side. There are three conditions when comparing these four paths

(i) $P_1 \subset P_0$: Because node 1 belongs to the conducting path P_0, the internal charge stored in the capacitors connected to node 1 must be discharged firstly before the external charge begins to decrease. So the internal charge in this case affects the switching delay. Fig. 3 shows the comparison of the waveforms obtained by using Spice. Let $V_2 = V_3 = V_4 = V_5 = V_6 = V_9 = V_{10} = V_{12} = 5$ V and $V_8 = 0$ V. If $V_{11} = 5$ V before V_1 and V_7 change to 5 V, the internal node capacitances were discharged before the pull-down path is formed. So there are no internal charges stored at node 1. The waveform of the output voltage is the solid line shown in Fig. 3*a*. But if $V_7 = 5$ V before V_1 and V_{11} change to 5 V, the internal node capacitances are changed to high before the pull-down path is formed. So there are internal charges stored at node 1. The waveform of the output voltage begins to decrease later than no-internal-charge case because these internal charges must be discharged firstly after the conducting path formed. The output voltage waveform is the dashed line shown in Fig. 3*a*. As shown in this Figure, the switching delays of these two cases are different, while the slopes of these two curves are almost the same.

(ii) $P_0 \subset P_2$: Because the output node belongs to P_2 so the charges stored at node 2 will flow through the output node. This means that the output voltage begins to decrease firstly and then the internal charges stored at node 2. These internal charges can be viewed as a charge supplier which can supplement the charge loss at the load capacitor, so the internal charge in this case affects the slope of the output voltage waveform. Let $V_2 = V_4 = V_6 = V_9 = V_{10} = V_{11} = V_{12} = 5$ V and $V_8 = 0$ V. If $V_5 = 5$ V before V_1, V_3 and V_7 change to 5 V, node 2 is isolated from the pull-down path. So there are no internal charges that will affect the transition. The waveform of the output voltage is the solid line shown in Fig. 3*b*. But if $V_5 = 0$ V before V_1, V_3 and V_7 change to 5 V, node 2 is connected to ground through a pull-down path. So there are more charges to be discharged through the conducting path than that of the no-internal-charge case. The waveform of the output voltage is the dashed line in Fig. 3*b*. The switching delays of these two circuits are almost the same, while the slopes are different. So the internal charges in this case will decrease the slope of the output waveform.

(iii) $P_0 \cap P_3 \neq 0$, but $P_0 \not\subset P_3$ and $P_3 \not\subset P_0$: There are

three possible conditions: P_0 is longer than P_3, P_3 is longer than P_0, or the lengths of these two paths are equal. The effect of this type of internal node is more

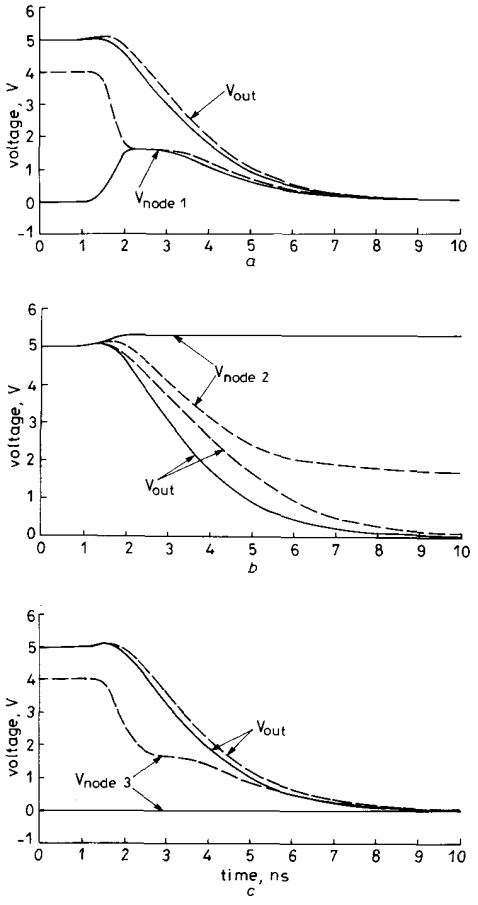

Fig. 3 *Comparison of output waveforms*
a case 1
b case 2
c case 3
——— No internal change
- - - - with internal change

complex than the two previous cases. But we determine that it affects the switching delay more than the slope. At the price of less accuracy, we process this type of internal nodes only when calculating the switching delay. Let $V_2 = V_3 = V_4 = V_5 = V_6 = V_7 = V_9 = V_{12} = 5$ V and $V_8 = V_{10} = 0$ V. If $V_8 = 0$ V before V_1 and V_{11} changes to 5 V, node 3 is isolated from the pull-down path. So there are no internal charges that will affect the transition. The waveform of the output voltage is the solid line in Fig. 3*c*. But if $V_8 = 5$ V before V_1, and V_{11} changes to 5 V, node 3 is connected to ground. The waveform of the output voltage is the dashed line in Fig. 3*c*. As shown, this type of internal charges affects the switching delay.

So the internal charges of case 1 and 3 should be considered when computing the switching delay and only the internal charges of case 2 taken into account when calculating the slope.

3 Series–parallel tree

The primary goal of a waveform-based event-driven timing simulator is to calculate the time the next event takes place and the waveform of this new event. To obtain an accurate waveform, one must know how many conducting paths exist in a gate when a new event happens. We use the series–parallel tree [9] to represent a logic gate, so we can obtain all conducting paths easily. If a logical gate is implemented based on some restricted design styles, such as full complementary CMOS pseudo-NMOS, dynamic CMOS, and so on, it can be represented as a merged series–parallel tree. This merged tree consists of two series–parallel trees representing the pull-up and pull-down subcircuits of this gate, respectively. We call this merged tree a PN tree, and the left and right childs of the root of the PN tree are the P tree and N tree, respectively. Fig. 4(*b*) illustrates the corresponding PN tree of the gate shown in Fig. 4*a* whose function is $Z = \overline{A(B(C + D) + (E + F)G)}$. The first item of the vertex of the tree is the type of node (parallel, series, leaf or output). S1, S2, S3, S4, S5 and S6 of the PN tree represent the internal nodes 1, 2, 3, 4, 5 and 6, as shown in Fig. 4*a*. The second item of each vertex is the gate signal if the vertex is a leaf vertex or is (H, L) if the vertex represents an internal node. The asterisk inside the second item of a leaf vertex means this leaf vertex is connected to the reference source, such as V_{DD} or ground. If the internal node is charged to high, the second item of the series vertex is H. Otherwise the second item is L. For example, if $A = B = G = 1$ and $C = D = E = F = 0$, the output voltage is high. Meanwhile, node 1, 2, 4, 5 and 6 are charged to high, so the second items of S1, S2, S4, S5 and S6 are H and the second item of S_3 is L, as shown in Fig. 4*b*.

Because the internal charges in the P tree can affect high-to-low transition and the internal charges in the N tree can affect the low-to-high transition, so we merge the P tree and N tree into a single tree, the PN tree. The PN tree can be used to calculate the transition delay of some special gates in which the pull-up and pull-down parts are turned on simultaneously, for example the gate implemented based on the pseudo-NMOS design style. The PN tree has the characteristics:

(*a*) The PN tree can represent the real netlist connections of the circuit. It also illustrates the relative position of an MOS transistor inside the circuit. The left child of a series vertex is nearer to the output node than the right child of this vertex. For example, the left child C of S_5 is nearer to the output node than the right child D. That means the PMOS transistor with gate signal C is nearer to the load capacitor than the PMOS with gate signal D, as shown in Fig. 4*a*.

(*b*) The reference source V_{DD}/ground propagates from the right-lowest vertex of P/N tree to the root of the PN tree. The signal propagates from the right child of a series vertex, through this series vertex, and then the left child of this vertex, and finally to the parent vertex of this series vertex. While the signal can propagate from the left or right child of a parallel vertex to the parent vertex of this vertex. For example, signal V_{DD} may propagate along F-S6-E, then to S4, or along G to S4.

According to the two characteristics, we develop an algorithm to trace all possible conducting paths from the output node to the references V_{DD}/ground. The algorithm is

```
EvaluateConductingPath()
{
  for each block {
    path_array_in_N = TracePathInSubTree(N_tree of this block, & num_of_path_in_N)
    path_array_in_P = TracePathInSubTree(P_tree of this block, & num_of_path_in_P)
  }
}
TracePathInSubTree(VERTEX *vertex, int *number)
{
  if vertex is a leaf vertex {
    allocate a single-node path;
    value of this node = value of vertex;
    *number = 1;
    return the head of this path;
  }
  left_path_array = TracePathInSubTree(left subtree of vertex, & num_of_path_in_left);
  Right_path_array = TracePathInSubTree(right subtree of vertex, & num_of_path_in_right);
  if vertex is a series vertex {
    *number = num_of_path_in_left * num_of_path_in_right;
    allocate *number heads;
    for every left path (i) {
      for every right path (j) {
        k = i * num_of_path_in_right + j
        append left path (i) to head k;
        append right path (j) to head k;
      }
    }
    return head;
  }
  else{
    *number = num_of_path_in_left + num_of_path_in_right;
    allocate *number heads;
    for every left path (i)
      append left path (i) to head i;
    for every right path (j)
      append right path (j) to head (i + j);
    return head;
  }
}
```

This algorithm is intrinsically recursive. Let L be the number of the leaf vertex, S be the number of the series vertex, and P be the parallel vertex. This recursive subroutine TracePathInSubTree will be called $(L + S + P)$ times. But the CPU time spent in each call depends on the kind of vertex, and how deep this subroutine is called recursively. We find that the most expensive operation is 'append a path to a head'. If we assume

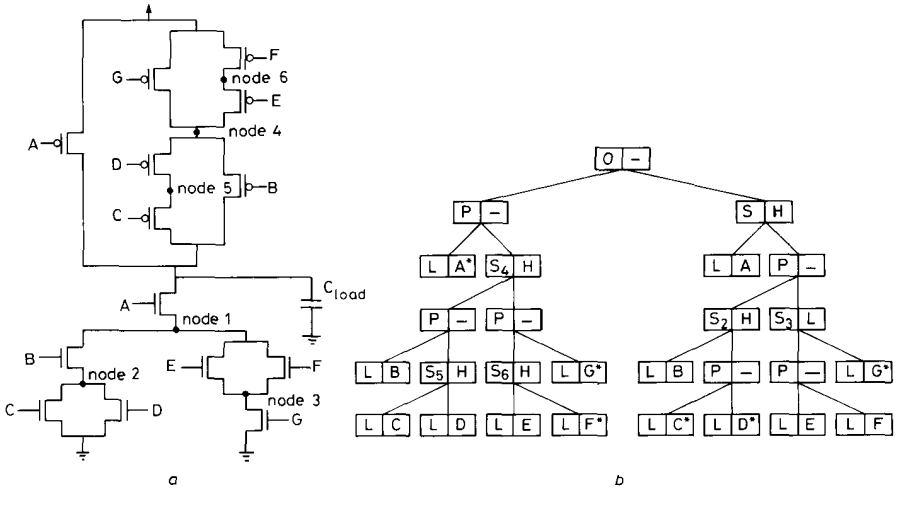

Fig. 4 *CMOS complex gate*
a Circuit diagram b equivalent merged series–parallel tree

this operation as unit time, although the time needed depends on how long the path is, the total time unit needed for a block is

$$\sum_{i=1}^{S} 2r_i l_i + \sum_{i=1}^{P} (r_i + l_i) \tag{1}$$

where r_i is the number of the paths in the right subtree of vertex i, and l_i is the number of the paths in the left subtree of vertex i.

After assigning the on/off state to the leaf vertex according to the gate signal, it is very easy to evaluate the conducting paths from the output node to V_{DD}/ground directly from the PN tree. For example, there are two conducting paths G-S4-B and G-S4-D-S5 C from V_{DD} to the output node when $A = E = F = 1$ and $B = C = D = G = 0$. But there are two cases to be considered when searching the conducting paths in a PN tree from a series vertex to V_{DD}/ground:

(i) Searching paths from series nodes in N(P) tree to ground(V_{DD}): The path from a series vertex consists the right child of the vertex. If there is no leaf vertex connected to the reference source in the right child, ascend to the parent of this series vertex. If the parent is a parallel vertex, ascend further. If this series vertex is the right child of its parent, ascend further. Otherwise, the path includes the right child. Continue the process until to the reference source. The path through a series vertex is through the left child firstly and then the right child of this vertex. While there are two paths through a parallel vertex, one through the left child and the other through the right child.

(ii) Searching paths from series nodes in P(N) tree to ground(V_{DD}): Because the path will cross the root of the PN tree, so the path consists of two parts, one from the series vertex to the root and the other from the root to ground/V_{DD}. The second part can be obtained as previously. The path from a series node to the root consists of the paths in the left child which can be obtained as follows: the paths from a series vertex to the root are the paths in the left child of this vertex. If there is no leaf vertex connected to the output node in the left child, ascend to the parent of this series vertex. If the parent is a parallel vertex, ascend further. If this series vertex is the left child of its parent, ascend further. Otherwise, the path includes the left child. Continue the process until to the output vertex. The paths through a series vertex consists of the right child firstly, then the series vertex, and finally the left child, while there are two paths through a parallel vertex.

For example, S1, S3, S4 and S5 are charged to high when $A = E = F = 1$ and $B = C = D = G = 0$; S2 and S6 are not changed. When G changes from low to high, then two pull-down paths G-S3-E-S1-A and G-S3-F-S1-A are formed, and the output will be discharged to low. S1 and S3 are considered as case 1. There are two paths G-E and G-F from S1 to ground, and there is one path G from S3 to ground. S4 and S5 are considered as case 2. There is one path C from S5 to the root and there are two paths B and D-S5-C from S4 to the root.

4 Calculation of delays

4.1 Definition of delay

To develop a delay model, a consistent and meaningful definition of delay is necessary. The most popular delay definition, provided by Elmore [5], can be inconsistent in MOS circuits because it occasionally gives rise to negative delays. The following definition [8] is used to alleviate this problem. Delay is measured as the difference between the time the output signal crosses a certain threshold voltage and the time the input signal crosses that same threshold voltage. Although this definition is reasonable, it is not suitable for a waveform-based timing simulator because the definition is strongly coupled with the choice of threshold voltage. So we define two terms that can be used in timing simulators: 'switching delay' as the difference between the time the output signal begins to change and the time the input signal begins to change and 'slope' as the changing rate after the output begins to change. These two terms constitute the event used in waveform-based timing simulations. The sign of the 'slope' can represent the rise or fall of the signal. If we want to know the 50%–50% delay for a path this can be calculated as the summation of the switching delay of each block in the path, that is to say, add the time the primary output of this path rises/falls to 50% and subtract the time the primary input of this path rises/falls to 50%.

4.2 Representation of voltage waveform

The delay is measured as a function of the output capacitance and the input waveform slope. The waveform slope can be represented by a single parameter T if we approximate the waveform by a linear segment followed by an exponential tail [8], as shown in eqn. 2 and eqn. 3 for a rising and falling signal, respectively.

$$f = \begin{cases} \dfrac{0.2t}{T} & t < 3T \\ 1 - 0.4 \exp\left(-\dfrac{t - 3T}{2T}\right) & t \geqslant 3T \end{cases} \tag{2}$$

$$f = \begin{cases} 1 - \dfrac{0.2t}{T} & t < 3T \\ 0.4 \exp\left(-\dfrac{t - 3T}{2T}\right) & t \geqslant 3T \end{cases} \tag{3}$$

where T is the time spent by the signal between 10% and 50% of the steady state. The rising waveform of a simple RC circuit can be represented as

$$V_0 = V_{DD}\left(1 - \exp\left(-\frac{t}{RC}\right)\right) \tag{4}$$

So T can be evaluated as

$$T = \frac{T_{10\%-50\%}}{0.4 V_{DD}} \tag{5}$$

where $T_{10\%-50\%} = 0.588RC$. Consider a CMOS inverter with various loads. The effective resistance of this gate is 3080 Ω and the parasitic capacitance is 0.016 pF. The results are shown in Fig. 5. The dashed lines are obtained using the preceding equations. So the waveform can be obtained if one calculates the time constant RC accurately. This method is simple and the results are accurate when compared to that of Spice. For simplicity, we use 'slope' to represent the time constant in the following.

4.3 Effective resistance

A transistor is modelled as an open switch if the gate is connected to a node with a logic of L (low). If the gate is H (high), the transistor is modelled as a resistor. The value of this effective resistance depends on the physical parameters, such as L and W, and the slope of the signal

at the gate, and is a nonlinear function. We don't try to model the transistor as having a linear resistance to achieve greater accuracy, because solving a general linear

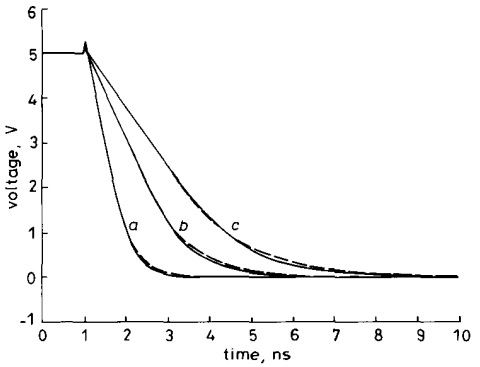

Fig. 5 *Output waveforms of CMOS inverter with various load capacitances*

a 0.2 pF slope 0.21 ns/V
b 0.5 pF slope 0.482 ns/V
c 0.8 pF slope 0.753 ns/V

resistance network is prohibitive expensive. So the discrete switch model is used in our program. The value of the effective resistance may be one of the three cases

$$R_{eff} = \begin{cases} \infty & \text{if the gate is L} \\ R_{on} & \text{if the gate is H} \\ R_t & \text{if the gate changes from L to H} \end{cases}$$

The values of R_{on} and R_t depend on the physical parameters and the load capacitance, and R_t depends also on the slope of the signal at the gate. These two values can be obtained from the simulated results of a CMOS inverter using Spice. R_{on} is obtained with an impulse at the input, and R_t is obtained with a ramp signal as input. So we maintain two tables in our program, one two-dimensional and the other three-dimensional.

$$R_{on} = f(W/L, C_{load}) \quad (6)$$

$$R_t = f(W/L, C_{load}, slope) \quad (7)$$

Of course, it is very expensive to maintain a three-dimensional table in the program if we try to model all possible transition cases. But because the load capacitance of a gate is the summation of the input capacitances of the fanout of this gate, so C_{load} will be discrete times of the input capacitance of a CMOS inverter with the minimum size (W/L). From the simulated results of Spice, the relationship between the effective resistance and the slope of the gate signal is smoothly linear, so only a few slopes need be recorded, and the actual value can be obtained using interpolation.

Whether calculating the slope or the switching delay, we need to calculate the total effective resistance from a node to the reference sources V_{DD}/ground. If this node is the output node of this gate, the recursive algorithm [9] can be applied to obtain the total effective resistance R_{tol}. But if this node is an internal node, this algorithm is no more applicable. We develop a new algorithm that can be used to calculate R_{tol} from an internal node to the reference sources. For each internal node to be charged/discharged, the circuit configuration can be simplified to that shown in Fig. 6*a*, where Ui and Di are the up and down series-connected effective resistances and Pi is the parallel-connected effective resistance. For each Ui, Di and Pi, it could be open-circuit, short-circuit or one finite resistance. For example, Fig. 6*b* shows the equivalent circuit diagram from node 5 in Fig. 4 to V_{DD}, where U2 is short circuit; Fig. 6*c* shows the equivalent circuit diagram from node 2 to ground, where D2 is short circuit and P2 is open circuit.

We use the following algorithm to convert the PN tree onto the equivalent mesh from an internal node to the reference source V_{DD}/ground.

```
ConstructMesh(VERTEX *vertex)
{
  while (vertex is not the root of the tree) {
    if (vertex → parent is a series vertex) {
      while (vertex → parent is a series vertex) {
        if vertex is the left child of its parent
          add the effective resistance of vertex → parent → right
              to the d_i;
        else
          add the effective resistance of vertex → parent → left
              to the u_i;
        vertex = vertex → parent;
      }
    }
    else {/* vertex is the right child of its parent */
      while (vertex → parent is a parallel vertex) {
        if vertex is the left child of its parent
          add the effective resistance of vertex → parent → right
              to the p_i;
        else
          add the effective resistance of vertex → parent → left
              to the p_i;
        vertex = vertex → parent;
        if (vertex is the root) break;
      }
      append u_i, d_i, and p_i to the mesh;
    }
  }
}
```

4.4 Calculation of slope

The first objective is to determine the slope associated with a signal propagating from a reference source to the output node. Because some internal charge will affect the

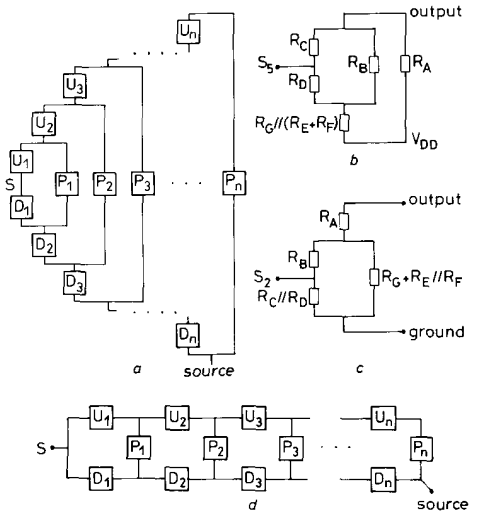

Fig. 6 *Mesh structure used to calculate total effective resistance from internal node S to source*

a Simplified circuit configuration
b Equivalent circuit from node 5 to V_{DD}
c Equivalent circuit from node 2 to ground
d Translated circuit

slope, we must calculate the time constant RC due to external charges and internal charges, separately. And then use the largest RC to calculate the slope of the output voltage. Two theorems proven in Reference 9 can be applied to calculate the slope due to the external charges. The algorithm for computing the slope consists of a recursive post-order traversal of the series–parallel tree by applying these two theorems to the parallel and serial nodes, separately. The slope calculated for the leaf nodes are the product of the effective resistance and the parasitic capacitance.

Since the conducting path for the internal node of case 2 contains the output node, so the slope of internal charges will be greater than that of the external charge. The slope is determined by the internal charges farthest away from the output node in most cases. The slope of the internal charges can be expressed as

$$S_i = S_o + \sum (R_o + R_k)C_k \tag{8}$$

where S_o is the slope of the external charge and R_o is the total effective resistance of the conducting path of output node. R_k is the total resistance from node k to the output node and C_k is the parasitic capacitance at node k. For example, if G in Fig. 4*a* changes to high at some time, the output voltage will decrease to low. The slope of the output voltage due to this event can be obtained as

$$\begin{aligned}\text{slope} = {} & R_{Gn}C_{S_3} + (R_{Gn} + R_{En}//R_{Fn})C_{S_1} \\ & + (R_{An} + R_{Gn} + R_{En}//R_{Fn})C_{load} \\ & + (R_{Cp} + R_{An} + R_{Gn} + R_{En}//R_{Fn})C_{S_5} \\ & + (R_{Dp} + R_{Cp} + R_{An} + R_{Gn} + R_{En}//R_{Fn})C_{S_4}\end{aligned}$$

4.5 Calculation of switching delay

The switching delay consists of two parts: the time needed to form a conducting path and the charging/ discharging time of the internal capacitances. The first term is determined by the threshold voltage V_T of a MOS transistor and the slope of the input signal. It can be obtained easily if the input waveform is expressed as shown in eqns. 1 and 2. The charging/discharging time is calculated approximately as

$$\Delta t = \frac{Q}{\bar{I}} \simeq \frac{\int C\,dV}{V_s} 2R \tag{9}$$

where $\bar{I}$ is the average current, V_s is the voltage swing and R is the effective resistance of the conducting path. Because the capacitance C is a function of voltage, so the internal charge can be easily obtained from the integration. But since there are many possible conducting paths that the internal charges can flow, the calculation of R is not so easy. For example, internal charges at node 2 can flow through C, D, or through B-E-G.

For each internal node to be charged/ discharged, the configuration can be reduced to that shown in Fig. 6*a*. For each Ui, Di and Pi it could be open-circuit, short-circuit or a finite resistance. The configuration can be translated to the circuit shown in Fig. 6*d*. The algorithms are discussed in the next Section. Then the problem is to find the equivalent resistance from node S to the source. As a result, the total resistance R can be obtained recursively from this diagram.

In the example, S3 and S1 must be taken into account when calculating the switching delay. So there are two paths from S1 to ground, E-G and F-G, and there is one path from S3 to ground, which is G. So the discharge time can be obtained as

$$t_{discharge} = \Delta t_{S_1} + \Delta t_{S_3} \tag{10}$$

where

$$\Delta t_{S_3} = 2R_{Gn}\frac{Q_{S_3}}{V_s}$$

$$\Delta t_{S_1} = 2(R_{Gn} + R_{En}//R_{Fn})\frac{Q_{S_1}}{V_s}$$

So the output voltage waveform can be obtained after calculating the switching delay and the slope.

5 Simulation algorithm and results

The algorithm has been implemented in our event-driven switch-level timing simulator. The simulator reads in the circuit description file and partitions the circuit into small gates. If there is no feedback path existing in the circuit, a waveform-evaluation technique is used to obtain the whole waveform of each gate, one by one from the primary input to the primary output of the circuit. However, if a feedback path exists in the circuit, the conventional event-driven technique is used, event evaluation, event propagation and event insertion and deletion. The event consists of the time the event takes place and the slope of the event, positive for rising and negative for falling. The data structure of the node contains an event list that records all events happening at this node. After all events are processes, the whole voltage waveform at each node can be obtained from the event list.

The simulation algorithm is summarised as follows

```
Simulation ( ) {
  schedule input vectors into an event pool;
  while (event pool is nonempty) {
    pick event e from top of the event queue;
    for each state transient s affected by e {
      t = time needed to form a conducting path;
      if s is high-to-low {
        for each internal node with status high {
          trace the PN tree to obtain the path list connected to ground
          if this node belonged to case1 and case3 {
            t = t + Δt;
            assign low to this node
          }
          else
            calculate slope;
        }
      }
      else {
        for each internal node with status low {
          trace the PN tree to obtain the path list connected to V_DD
          if this node belonged to case1 and case3 {
            t = t + Δt;
            assign high to this node
          }
          else
            calculate slope;
          }
        }
        s.time = e.time + t;
        calculate slope of output recursively;
        s.slope = the maximum value of all slope
        schedule s into the event pool;
      }
    }
  }
}
```

The simulator has been tested extensively for basic modules such as counter, decoders, adders and ALUs. The CPU time comparisons are summarised in Table 1.

Table 1: Comparisons between BTS and Spice3

Circuit	MOD no.	CPU time on Sun4	
		BTS	Spice3
		(s)	(s)
74381	584	1.183	1478.68
7483	258	0.433	282.37
74147	146	0.183	121.55
100 stage inverter chain	200	0.216	100.7

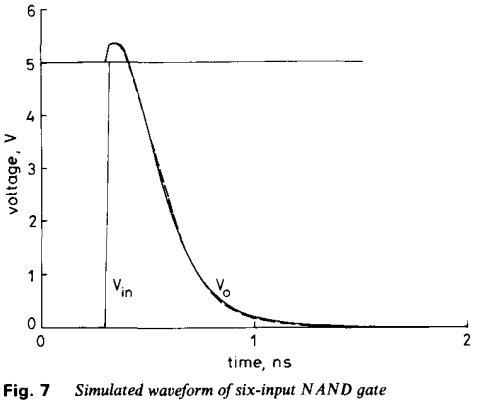

Fig. 7 *Simulated waveform of six-input NAND gate*
---- Our delay model
——— Spice result

Since BTS considers the effects of the internal charges, the simulator is not as fast as other timing simulators, but the waveforms it derives are more accurate. A six-input NAND gate is simulated by using Spice and our timing simulator. The results are compared in Fig. 7. The dashed line is the result obtained from our simulator. This waveform is described by using eqn. 3 and the slope is calculated as stated in Section 4. The result is accurate when compared with the result from Spice. An one-bit full adder is implemented based on the complementary cMOS design style. There are many internal nodes that must be taken into considerations when calculating the switching delay and the slope. This adder is simulated by using our timing simulator and by using Spice. The results are shown in Fig. 8. Fig. 8*a* shows the input waveforms of V_A, V_B and V_C. Fig. 8*b* and *c* show the voltage waveforms at terminals Carry and Sum. The results are satisfactory. Fig. 9 and Fig. 10 show the voltage waveforms of a decoder 74S138 and a four-bit ALU 74S381, respectively. The results are good agreement.

6 Conclusion

An accurate delay calculation method that computes the switching delays and slopes in series–parallel RC networks with the considerations of the effects of the internal charges. We have proposed a PN tree to represent the structural connection of the gate and described a method to enumerate all conducting paths in the gate. We have also investigated the relation between the slope of waveform and the *RC* time constant. After classifying the internal node and investigating the effects of different

internal charges, a simple but reasonable method calculates the switching delay and the slope. The accuracy is satisfactory when compared to results of Spice simulation.

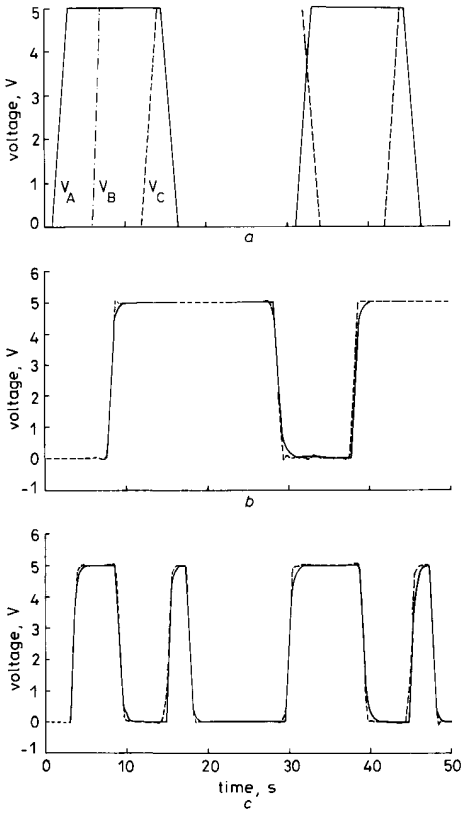

Fig. 8 *Simulated waveforms of one-bit full adder*
a input waveforms
b Carry waveform
c Sum waveform
------- Spice
——— Our model

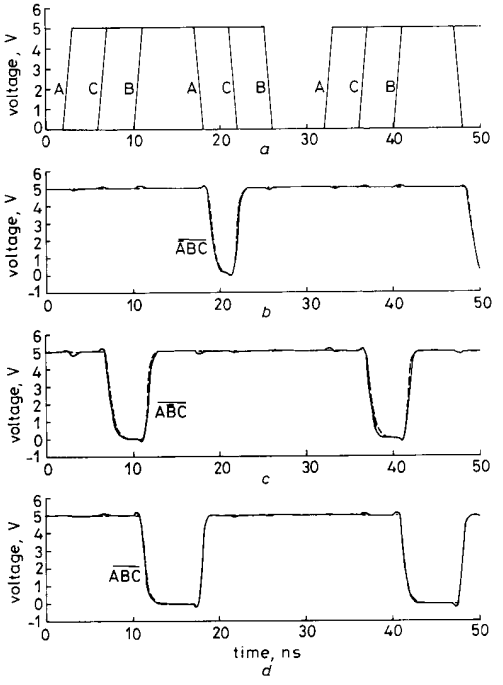

Fig. 9 *Simulated waveforms of decoder 74S138*

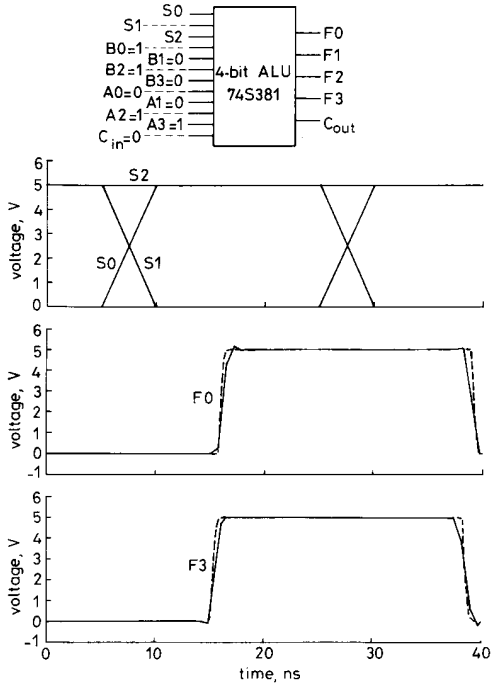

Fig. 10 *Simulated waveforms of four-but ALU 74S381*

7 References

1 TERMAN, C.J.: 'RSIM — A logic-level timing simulator'. Proceedings of International Conference on *Computer design*, 1983, pp. 437–440
2 OUSTERHOUT, J.K.: 'Switch-delay models for digital MOS VLSI'. Proceedings of 21st Conference on *Design automation*, 1984, pp. 542–548
3 RUBINSTEIN, J., PENFIELD, P., and HOROWITZ, M.: 'Signal delay in RC tree networks', *IEEE Trans.*, 1983, **CAD-2,** pp. 202–211
4 WYATT, J.L.: 'Signal delay in RC mesh networks', *IEEE Trans.*, 1985, **CAS-32,** pp. 507–510
5 ELMORE, W.C.: 'The transient response of damped linear networks with particular regards to wide-band amplifiers', *J. Appl. Phys.*, 1948, **19,** pp. 55–63
6 LIN, T.M., and MEAD, C.A.: 'Signal delay in general RC-networks', *IEEE Trans.*, 1984, **CAD-3,** pp. 331–349
7 CHAN, P.K., and KARPLUS, K.: 'Computing signal delay in general RC networks by tree/link partitioning'. Proceedings of 26th ACM/IEEE conference on *Design automation*, Las Vegas, CA, USA, June 1989, pp. 485–490
8 CHANG, F.C., CHEN, C.F., and SUBRAMANIAM, P.: 'An accurate and efficient gate level delay calculator for MOS circuits'. Proceedings of 25th ACM/IEEE conference on *Design automation*, Anaheim, CA, USA, June 1988, pp. 282–287
9 CAISSO, J.-P., CERNY, E., and RUMIN, N.C.: 'A recursive technique for computing delays in series-parallel MOS transistor circuits', *IEEE Trans.*, 1991, **CAD-10,** pp. 589–595